Петин В. А., Биняковский А. А.

ПРАКТИЧЕСКАЯ ЭНЦИКЛОПЕДИЯ ARDUINO

Издание второе, дополненное

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО
Москва, 2020

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515
П29

Петин В. А., Биняковский А. А.

П29 Практическая энциклопедия Arduino. 2-е изд., доп. – М.: ДМК Пресс, 2020. – 166 с.

ISBN 978-5-97060-798-5

В книге обобщаются данные по основным компонентам конструкций на основе платформы Arduino, которую представляет самая массовая на сегодняшний день версия ArduinoUNO или аналогичные ей многочисленные клоны. Книга представляет собой набор из 33 глав-экспериментов. В каждом эксперименте рассмотрена работа платы Arduino с определенным электронным компонентом или модулем, начиная с самых простых и заканчивая сложными, представляющими собой самостоятельные специализированные устройства. В каждой главе представлен список деталей, необходимых для практического проведения эксперимента. Для каждого эксперимента приведена визуальная схема соединения деталей в формате интегрированной среды разработки Fritzing. Она дает наглядное и точное представление – как должна выглядеть собранная схема. Далее даются теоретические сведения об используемом компоненте или модуле. Каждая глава содержит код скетча (программы) на встроенном языке Arduino с комментариями. В конце каждой главы содержатся ссылки для скачивания скетчей с сайта http://arduino-kit.ru, дополнительных программ, а также на видеоурок данного эксперимента.

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515

ISBN 978-5-97060-798-5

СОДЕРЖАНИЕ

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.

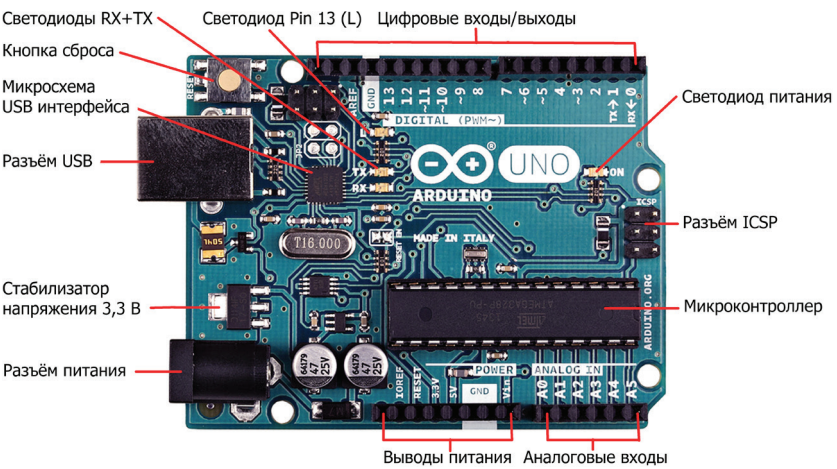

Рис. 1. Плата Arduino UNO

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;

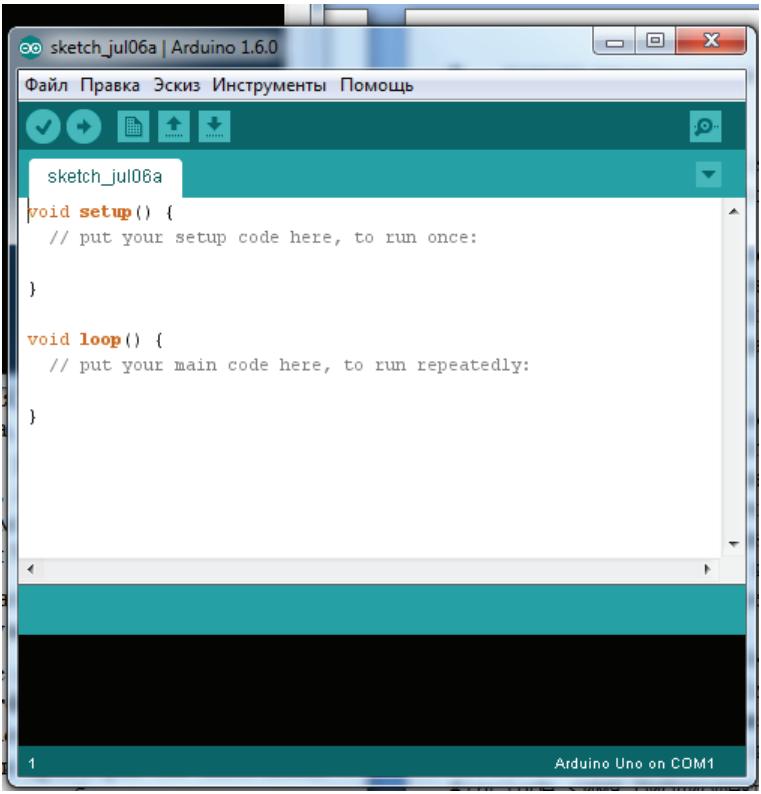

Рис. 2. Среда Arduino IDE

- 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.

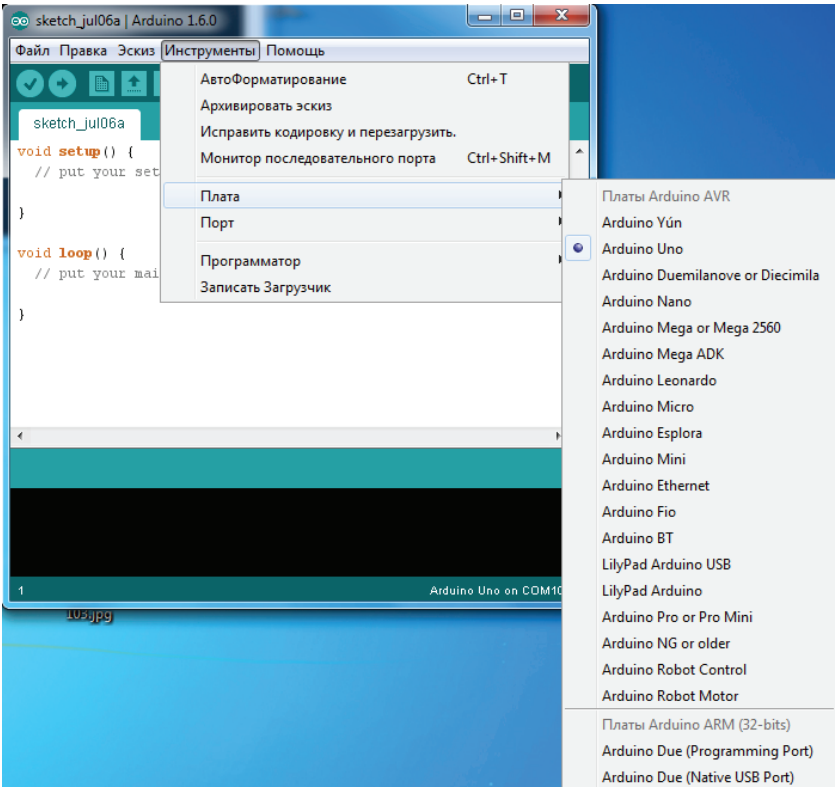

Рис. 3. Выбор Arduino платы

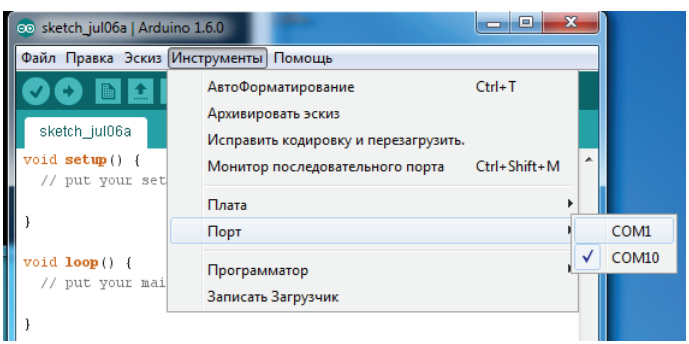

Рис. 4. Выбор порта подключения платы Arduino

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.

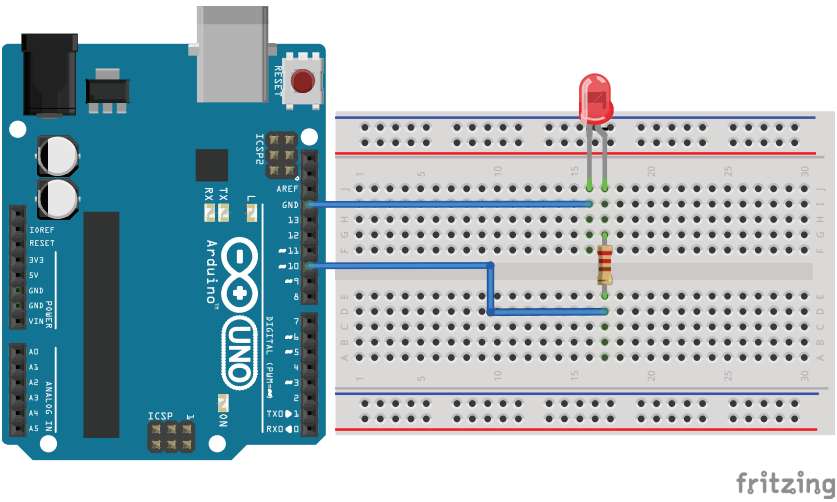

Рис. 1.1. Схема подключения светодиода

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

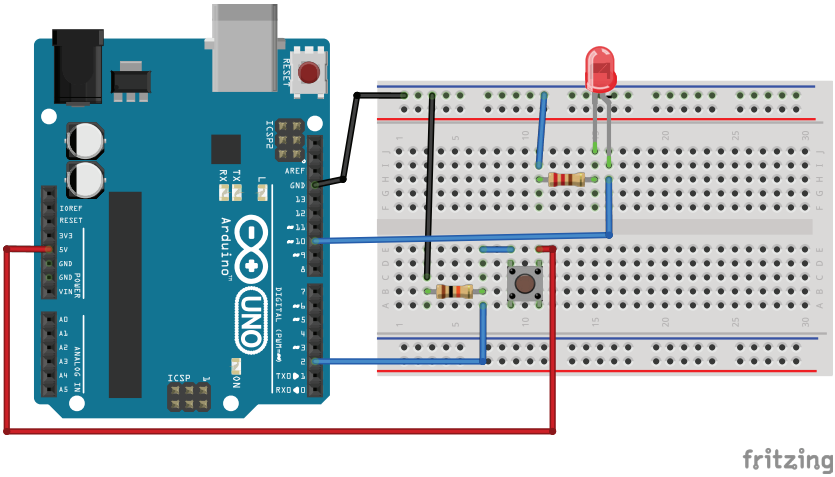

Рис. 2.1. Схема подключения кнопки и светодиода

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.

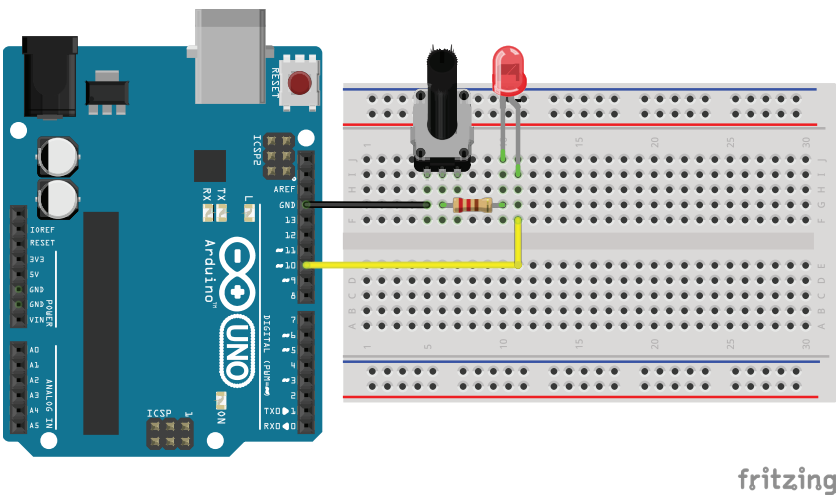

Рис. 3.1. Схема подключения потенциометра и светодиода

Скетч эксперимента приведен в листинге 3.1. Он совсем простой – нам необходимо просто включить светодиод, подключенный к цифровому выводу D10 Arduino.

Листинг 3.1

```
const int LED=10;    // вывод для подключения светодиода 10 (D10)
void setup()
   {
   // Конфигурируем вывод подключения светодиода как выход (OUTPUT)
   pinMode(10, OUTPUT);
   // включаем светодиод, подавая на вывод 1 (HIGH)
   digitalWrite(LED,HIGH);
   }
void loop()
 {;}
```

Порядок подключения:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) – к одной из ног резистора 220 Ом (см. рис. 3.1).
2. Свободную ногу резистора 220 Ом подключаем к средней ножке потенциометра, вторую (любую крайнюю) подсоединяем к GND (см. рис. 3.1).
3. Загружаем в плату Arduino скетч из листинга 3.1.
4. Поворачиваем ручку потенциометра и наблюдаем изменение яркости светодиода от полного выключения до почти полной яркости.

На странице http://arduino-kit.ru/0003 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 3.1.

4 Светодиодная шкала 10 сегментов. Крутим потенциометр, меняем количество светящихся светодиодов

В этом эксперименте мы рассмотрим работу аналоговых входов Arduino, работу потенциометра в качестве аналогового датчика и будем демонстрировать показания аналогового датчика с помощью светодиодной шкалы.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- потенциометр 2 кОм;
- светодиодная шкала 10 сегментов;
- резистор 220 Ом – 10 штук;
- провода папа-папа.

В предыдущих экспериментах мы рассматривали работу с цифровыми выводами Arduino, они имеют только два возможных состояния: включено или выключено, HIGH или LOW, 1 или 0. Но для получения информации об окружающем мире необходимо работать с аналоговыми данными, имеющими бесконечное число возможных значений в данном диапазоне. Для получения аналоговых данных Arduino имеет аналоговые входы, оснащенные 10-разрядным аналого-цифровым преобразователем для аналоговых преобразований. Точность АЦП определена разрешением. 10-разрядный означает, что АЦП может разделить аналоговый сигнал на 2^{10} различных значений. Следовательно, Arduino может присвоить $2^{10} = 1024$ аналоговых значения, от 0 до 1023. Опорное напряжение определяет максимальное напряжение, его значение соответствует значению 1023 АЦП. При напряжении 0 В на контакте АЦП возвращает значение 0, опорное напряжение возвращает значение 1023. Несмотря на то что можно изменить опорное напряжение, мы будем использовать опорное напряжение 5 В.

Рассмотрим, как использовать потенциометр в качестве аналогового датчика. Рисунок 4.1 показывает, как правильно подключить

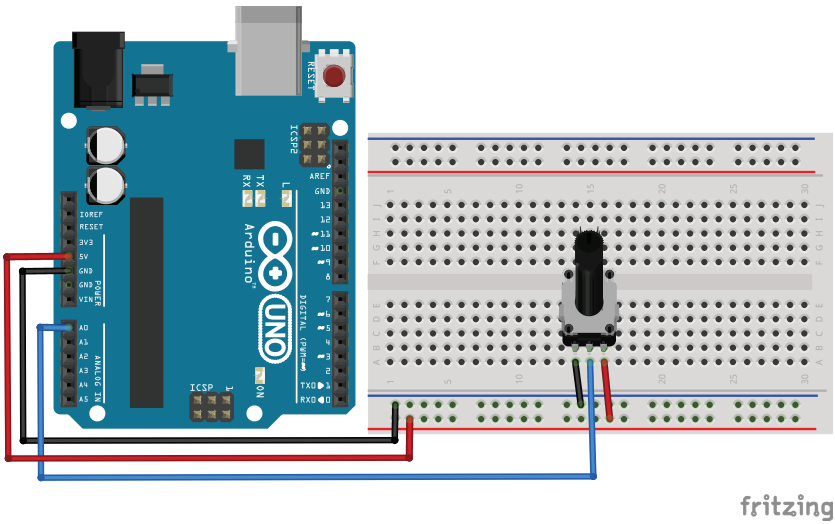

Рис. 4.1. Схема подключения потенциометра в качестве аналогового датчика

ваш потенциометр к Arduino в качестве аналогового датчика. Мы подключаем один из крайних выводов на землю, другой крайний вывод – к +5 В. Средний вывод потенциометра подключаем к аналоговому входу A0 платы Arduino. Для считывания данных с аналогового порта в Arduino есть функция `analogRead()`.

Загружаем на плату Arduino скетч из листинга 4.1 для считывания значений из аналогового порта и вывода их в монитор последовательного порта Arduino.

Листинг 4.1

```
const int POT=0;      // Аналоговый вход A0 для подключения потенциометра
int valpot = 0;       // переменная для хранения значения потенциометра
void setup()
   {
   Serial.begin(9600);
   }
void loop()
   {
   valpot = analogRead(POT);     // чтение данных потенциометра
   Serial.println(valpot);       // вывод значений в последовательный порт
   delay(500);                   // задержка 0.5 сек
  }
```

Порядок подключения:

1. Подключаем потенциометр по схеме на рис. 4.1.
2. Загружаем в плату Arduino скетч из листинга 4.1.
3. Запускаем в Arduino IDE монитор последовательного порта.
4. Поворачиваем ручку потенциометра и наблюдаем вывод аналоговых значений потенциометра в монитор последовательного порта (см. рис. 4.2).

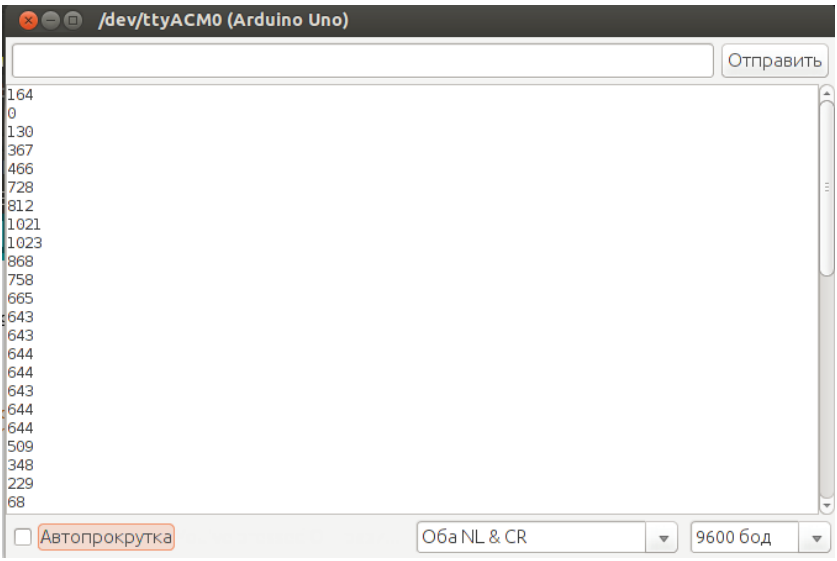

Рис. 4.2. Вывод аналоговых значений потенциометра в монитор последовательного порта

Теперь визуализируем аналоговые данные потенциометра с помощью 10-разрядной линейной светодиодной шкалы. Шкала представляет собой сборку из 10 независимых светодиодов с катодами со стороны надписи на корпусе. Для подключения шкалы к Arduino будем использовать 10 цифровых выводов D3–D12. Схема соединений показана на рис. 4.3. Каждый из светодиодов шкалы выводом анода соединен с цифровым выводом Arduino, а катодом на землю через последовательно соединенный ограничивающий резистор 220 Ом.

Аналоговые данные потенциометра (0–1023) масштабируем в данные шкалы (0–10) с помощью функции map() и зажигаем соответствующее количество светодиодов. Скетч приведен в листинге 4.2.

Листинг 4.2

```
const int POT=0;      // Аналоговый вход A0 для подключения потенциометра
int valpot = 0;       // переменная для хранения значения потенциометра
// список контактов подключения светодиодной шкалы
const int pinsled[10]={3,4,5,6,7,8,9,10,11,12};
int countleds = 0;   // переменная для хранения значения шкалы
void setup()
   {
   for(int i=0;i<10;i++)
      {
       // Сконфигурировать контакты подсоединения шкалы как выходы
       pinMode(pinsled[i],OUTPUT);
       digitalWrite(pinsled[i],LOW);
      {
   }
void loop()
   {
   valpot = analogRead(POT);            // чтение данных потенциометра
   // масштабируем значение к интервалу 0-10
   countled=map(valpot,0,1023,0,10);
   // зажигаем количество полосок на шкале, равное countled
   for(int i=0;i<10;i++)
      {
      if(i<countleds)                   // зажигаем светодиод шкалы
          digitalWrite(pinsled[i],HIGH);
      else                              // гасим светодиод шкалы
          digitalWrite(pinsled[i],LOW);
      }
}
```

Порядок подключения:

1. Подключаем потенциометр по схеме на рис. 4.1.
2. Подключаем выводы светодиодной шкалы контактами анодов через ограничительные резисторы номиналом 220 Ом к выводам Arduino D3–D12, контактами катодов – на землю (см. рис. 4.3).
3. Загружаем в плату Arduino скетч из листинга 4.2.
4. Поворачиваем ручку потенциометра и наблюдаем на светодиодной шкале уровень значения потенциометра от максимального номинала.

На странице http://arduino-kit.ru/0004 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 4.1, 4.2.

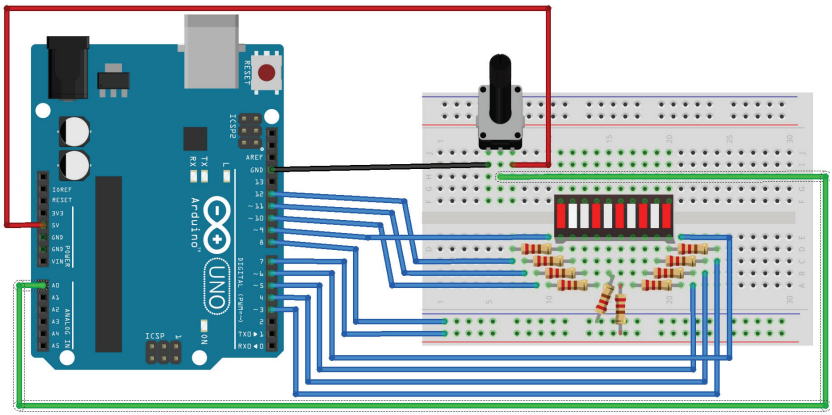

Рис. 4.3. Схема подключения линейной светодиодной шкалы

5 RGB-светодиод. Широтно-импульсная модуляция. Переливаемся цветами радуги

В этом эксперименте мы рассмотрим широтно-импульсную модуляцию, которая позволяет Arduino выводить аналоговые данные на цифровые выводы, и применим эти знания для создания прозвольных цветов свечения с помощью RGB-светодиода.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- RGB-светодиод;
- резистор 220 Ом – 3 штуки;
- провода папа-папа.

Arduino не может на цифровой вывод выдавать произвольное напряжение. Выдается либо +5 В (HIGH), либо 0 В (LOW). Но уровнем напряжения управляется многое: например, яркость светодиода или скорость вращения мотора. Для симуляции неполного напряжения используется ШИМ (широтно-импульсная модуляция, или PWM). ШИМ – это операция получения изменяющегося аналогового значения посредством цифровых сигналов. Цифровой сигнал на выходе постоянно переключается между максимальным и минимальным значениями. Переключение имеет частоту в тысячи герц. Глаз не замечает мерцания более 50 Гц, поэтому нам кажется, что светодиод не мерцает, а горит в неполную силу. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса (см. рис. 5.1).

Arduino-функция `analogWrite()` выдает ШИМ-сигнал на цифровой вывод Arduino. После вызова `analogWrite()` на выходе будет генерироваться постоянная прямоугольная волна с заданной шириной импульса до следующего вызова `analogWrite()`, частота выдаваемого ШИМ-сигнала равна 490 Гц.

На платах Arduino Nano и UNO ШИМ поддерживают выводы 3, 5, 6, 9, 10 и 11, на плате Mega – выводы 2–13. Данные выводы отмечены знаком тильды ~.

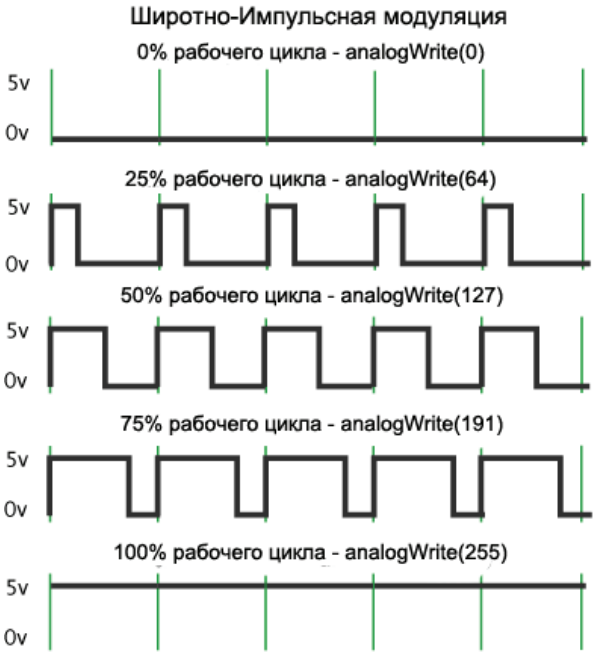

Рис. 5.1. Зависимость значений ШИМ от ширины импульса

В данном эксперименте мы используем RGB-светодиод. RGB расшифровывается как аббревиатура Red, Green, Blue, при помощи этих цветов можно получить любой цвет путем смешения. Светодиод RGB отличается от обычного тем, что содержит 3 небольших кристалла R, G, B, которые смогут синтезировать любой цвет или оттенок. RGB-светодиод имеет 4 вывода (см. рис. 5.2).

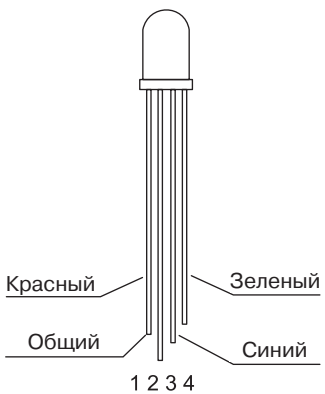

Рис. 5.2. Выводы RGB-светодиода

Подключим RGB-светодиод к плате Arduino и заставим переливаться его цветами радуги. На рис. 5.3 показана схема подключения RGB-светодиода к плате Arduino.

Теперь перейдем к написанию скетча. На самом деле радуга имеет множество

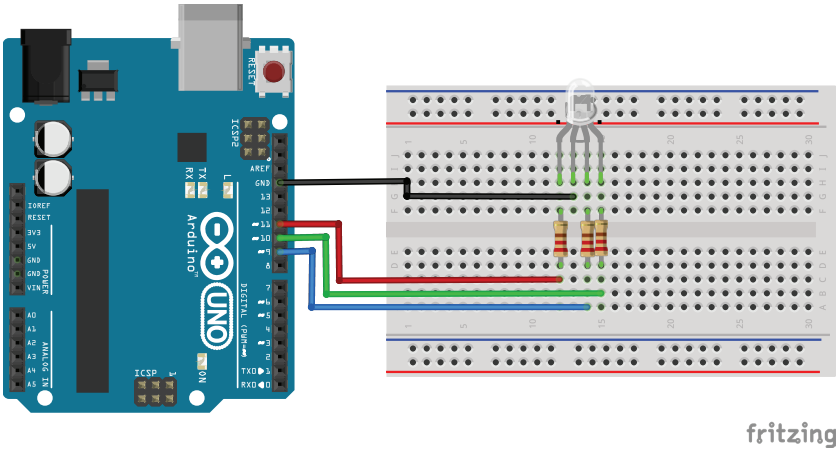

Рис. 5.3. Схема подключения RGB-светодиода

цветов, а 7 цветов были придуманы только потому, что эти цвета наиболее устойчиво воспринимаются и определяются глазом и мы их можем назвать и вспомнить поговоркой «Каждый Охотник Желает Знать, Где Сидит Фазан». Список этих 7 основных цветов радуги с разложением по компонентам R, G и B представлен в табл. 5.1.

Таблица 5.1

Цвет	R	G	B
Красный	255	0	0
Оранжевый	255	125	0
Желтый	255	255	0
Зеленый	0	255	0
Голубой	0	255	255
Синий	0	0	255
Фиолетовый	255	0	255

Наш светодиод должен переливаться от красного до фиолетового, проходя через все 7 основных цветов. Алгоритм вычисления любого промежуточного цвета радуги следующий:

1. Примем за начальную точку отсчета красный цвет (255, 0, 0).
2. Будем постепенно увеличивать значение зеленой составляющей G, пока не достигнем значения оранжевого (255, 125, 0), а затем и желтого цвета (255, 255, 0).

3. Постепенно уменьшим значение красной составляющей R до значения зеленого цвета (0, 255, 0).
4. Постепенно увеличим значение синей составляющей B до значения голубого цвета (0, 255, 255).
5. Постепенно уменьшим количество зеленой составляющей G до значения синего цвета (0, 0, 255).
6. Постепенно увеличим количество красной составляющей R до значения фиолетового цвета (255, 0, 255).
7. Выдерживаем небольшую паузу и переходим к шагу 1.

Содержимое скетча показано в листинге 5.1.

Листинг 5.1

```
const int RED=11;        // вывод красной ноги RGB-светодиода
const int GREEN=10;      // вывод зеленой ноги RGB-светодиода
const int BLUE=9;        // вывод синей ноги RGB-светодиода
int red;      // переменная для хранения R-составляющей цвета
int green;    // переменная для хранения G-составляющей цвета
int blue;     // переменная для хранения B-составляющей цвета
void setup()
   {
    pinMode(RED,OUTPUT);
    pinMode(GREEN,OUTPUT);
    pinMode(BLUE,OUTPUT);
   }
void loop()
   {
   // от красного к желтому
   red=255;green=0;blue=0;
   for(green=0;green<=255;green++)
        setRGB(red,green,blue);
   // от желтому к зеленому
   for(red=255;red>=0;red--)
        setRGB(red,green,blue);
   // от зеленого к голубому
   for(blue=0;blue<=255;blue++)
        setRGB(red,green,blue);
   // от голубого к синему
   for(green=255;green>=0;green--)
        setRGB(red,green,blue);
   // от синего к фиолетовому
```

```
   for(red=0;red<=255;red++)
         setRGB(red,green,blue);
   delay(2000);
   }
// функция установки цвета RGB-светодиода
void setRGB(int r,int g,int b)
   {
    analogWrite(RED,r);
    analogWrite(GREEN,g);
    analogWrite(BLUE,b);
    delay(10);
   }
```

Порядок подключения:

1. Чтобы видеть смешивание трех компонент R, G, B, а не отдельные составляющие, необходимо сделать поверхность светодиода шероховатой небольшой обработкой напильником или накрыть RGB-светодиод матовой пластиной.
2. Подключаем RGB-светодиод по схеме на рис. 5.3.
3. Загружаем в плату Arduino скетч из листинга 5.1.
4. Наблюдаем свечение светодиода переливающимися цветами радуги.

На странице http://arduino-kit.ru/0005 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 5.1.

6 Семисегментный индикатор одноразрядный. Выводим цифры

В этом эксперименте мы рассмотрим работу с семисегментным светодиодным индикатором, которая позволяет Arduino визуализировать цифры.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- одноразрядный семисегментный индикатор;
- резистор 510 Ом – 7 штук;
- провода папа-папа.

Светодиодный семисегментный индикатор представляет собой группу светодиодов, расположенных в определенном порядке и объединенных конструктивно. Светодиодные контакты промаркированы метками от a до g (и дополнительно dp – для отображения десятичной точки), и один общий вывод, который определяет тип подключения индикатора (схема с общим анодом ОА, или общим катодом ОК). Зажигая одновременно несколько светодиодов, можно формировать на индикаторе символы цифр. Схема одноразрядного семисегментного индикатора показана на рис. 6.1.

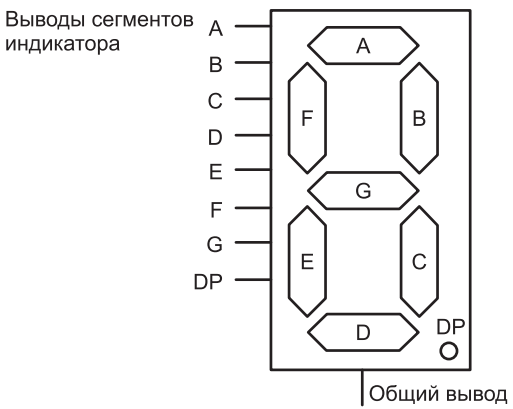

Рис. 6.1. Схема одноразрядного семисегментного индикатора

Для подключения одноразрядного светодиодного индикатора к Arduino будем задействовать 7 цифровых выводов, каждый из контактов a–g индикатора подключается к выводу Arduino через ограничительный резистор 470 Ом. В нашем эксперименте мы используем семисегментный индикатор с общим катодом ОК, общий провод подсоединяем к земле. На рис. 6.2 показана схема подключения одноразрядного семисегментного индикатора к плате Arduino.

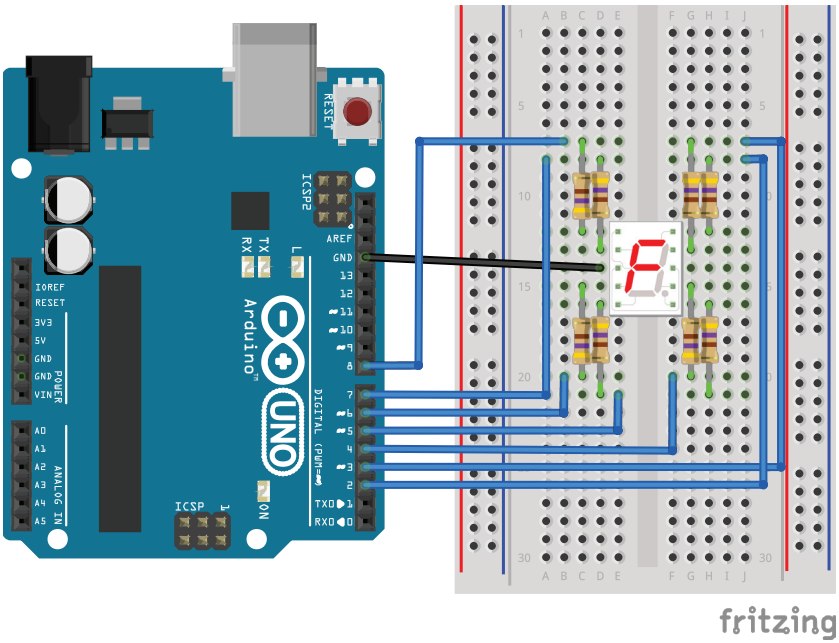

Рис. 6.2. Схема подключения семисегментного индикатора к Arduino

Приступим к написанию скетча. Мы будем на семисегментный индикатор в цикле выводить цифры от 0 до 9 с паузой 1 секунда. Сформируем массив значений для цифр 0–9, где старший разряд байта соответствует метке сегмента a индикатора, а младший – сегменту g.

```
byte numbers[10] = { B11111100, B01100000, B11011010, B11110010, B01100110,
B10110110, B10111110, B11100000, B11111110, B11110110};
```

Для преобразования значения цифры в данные для вывода значения на выводы Arduino будем использовать битовые операции языка Arduino:

```
bitRead(x,n);    // получение значения n разряда байта x
```

Скетч эксперимента представлен в листинге 6.1.

Листинг 6.1

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[7]={2,3,4,5,6,7,8};
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100, B01100000, B11011010, B11110010, B01100110,
B10110110, B10111110, B11100000, B11111110, B11100110};
// переменная для хранения значения текущей цифры
int number=0;
void setup()
   {
   // Сконфигурировать контакты как выходы
   for(int i=0;i<7;i++)
      pinMode(pins[i],OUTPUT);
   }
void loop()
   {
   showNumber(number);
   delay(1000);
   number=(number+1)%10;
   }
// функция вывода цифры на семисегментный индикатор
void showNumber(int num)
   {
   for(int i=0;i<7;i++)
       {
       if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
          digitalWrite(pins[i],HIGH);
       else    // потушить сегмент
          digitalWrite(pins[i],LOW);
       }
   }
```

Порядок подключения:

1. Подключаем семисегментный индикатор по схеме на рис. 6.2.
2. Загружаем в плату Arduino скетч из листинга 6.1.
3. Наблюдаем вывод цифр на экран семисегментного индикатора.

На странице http://arduino-kit.ru/0006 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 6.1.

7 Матрица 4-разрядная из 7-сегментных индикаторов. Делаем динамическую индикацию

В этом эксперименте мы рассмотрим работу Arduino с 4-разрядной семисегментной матрицей. Получим представление о динамической индикации, позволяющей использовать одни выводы Arduino при выводе информации на несколько семисегментных индикаторов.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- 4-разрядная семисегментная матрица;
- резистор 510 Ом – 8 штук;
- кнопка;
- резистор 10 кОм;
- провода папа-папа.

Матрица 4-разрядная из семисегментных индикаторов состоит из четырех семисегментных индикаторов и предназначена для одновременного вывода на матрицу 4 цифр, также есть возможность вывода десятичной точки. Схема 4-разрядной матрицы на 7-сегментных индикаторах показана на рис. 7.1.

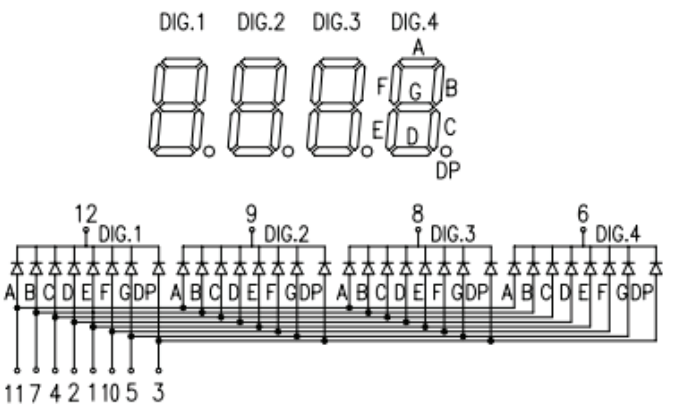

Рис. 7.1. Схема 4-разрядной матрицы на 7-сегментных индикаторах

Для вывода цифры необходимо зажечь нужные светодиоды на контактах A–G и DP и выбрать нужную матрицу подачей LOW на вывод 6, 8, 9 или12.

Подключим контакты матрицы к плате Arduino и будем выводить цифры на различные разряды матрицы. Для подключения нам понадобятся 12 выводов Arduino. Схема соединений для подключения 4-разрядной матрицы к плате Arduino показана на рис. 7.2. При подключении контактов используются ограничительные резисторы 510 Ом.

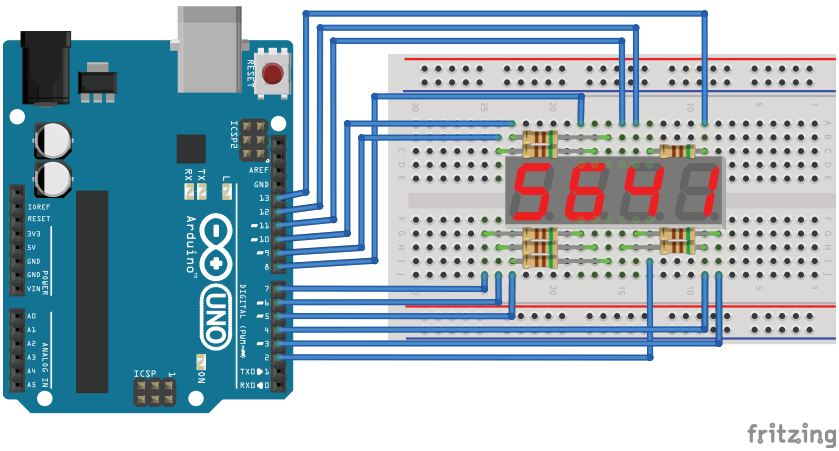

Рис. 7.2. Схема подключения 4-разрядной матрицы к Arduino

Напишем скетч последовательного вывода цифр (0–9) на произвольный регистр матрицы. Для выбора случайного значения из диапазона будем использовать функцию `random()`. В массиве `numbers[]` хранятся значения, соответствующие данным для отображения цифр 0–9 (старший разряд байта соответствует метке сегмента A индикатора, а младший – сегменту G), в массиве `pins[]` – значения контактов для сегментов A–G и DP, в массиве `pindigits[]` – значения контактов для выбора разряда матрицы. Содержимое скетча показано в листинге 7.1.

Листинг 7.1

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[8]={9,13,4,6,7,10,3,5};
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100, B01100000, B11011010,
```

```
B11110010, B01100110, B10110110,
B10111110, B11100000, B11111110,
B11110110};
// переменная для хранения значения текущей цифры
int number=0;
// семисегментного индикатора
int pindigits[4]={2,8,11,12};
// переменная для хранения текущего разряда
int digit=0;

void setup()
   {
   // Сконфигурировать контакты как выходы
   for(int i=0;i<8;i++)
      pinMode(pins[i],OUTPUT);
   for(int i=0;i<4;i++)
      {pinMode(pindigits[i],OUTPUT);
      digitalWrite(pindigits[i],HIGH);
      }
   }
void loop()
   {
   number=(number+1)%10;
   showNumber(number);   // DS
   for(int i=0;i<4;i++)
      digitalWrite(pindigits[i],HIGH);
   digit=random(0,4);
   digitalWrite(pindigits[digit],LOW);
   delay(3000);
   }
// функция вывода цифры на семисегментный индикатор
void showNumber(int num)
   {
   for(int i=0;i<7;i++)
       {
       if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
          digitalWrite(pins[i],HIGH);
       else    // потушить сегмент
          digitalWrite(pins[i],LOW);
       }
   }
```

Порядок подключения:

1. Подключаем семисегментный индикатор по схеме на рис. 7.2.
2. Загружаем в плату Arduino скетч из листинга 7.1.
3. Наблюдаем вывод цифр на экран семисегментного индикатора.

Возникает вопрос: как нам выводить цифры одновременно на все разряды матрицы? Если одновременно делать выбор всех разрядов

(одновременная подача LOW на выводы 6, 8, 9, 12), то на всех разрядах будет одна цифра. Но нам на каждый разряд одновременно необходимо выводить разные цифры. Эта проблема решается с помощью динамической индикации. Динамическая индикация подразумевает поочередное зажигание разрядов индикатора с частотой, не воспринимаемой человеческим глазом. Человеческий глаз обладает инерционностью, и если заставить индикаторы отображать информацию поочередно с достаточно большой скоростью, то человеку будет казаться, что все индикаторы отображают свою информацию непрерывно. В результате можно по одним и тем же проводникам поочередно передавать отображаемую информацию. Обычно достаточно частоты обновления информации 50 Гц, но лучше увеличить эту частоту до 100 Гц. Напишем скетч секундомера на нашей матрице. Для подсчета времени используем Arduino-функцию `millis()`, возвращающую количество миллисекунд, прошедшее с начала работы скетча. Добавим в схему кнопку и создадим секундомер 0–999 сек с точностью 0.1 сек, который будет запускаться с обнулением при нажатии и останавливаться с индикацией прошедшего времени при повторном нажатии кнопки.

Скетч для секундомера показан в листинге 7.2.

Листинг 7.2

```
// список выводов Arduino для подключения к разрядам a-g
// семисегментного индикатора
int pins[8]={9,13,4,6,7,10,3,5};
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100, B01100000, B11011010,
B11110010, B01100110, B10110110,
B10111110, B11100000, B11111110,
B11110110};
// переменная для хранения и обработки текущего значения
int number=0;
int number1=0;
int number2=0;
// семисегментного индикатора
int pindigits[4]={2,8,11,12};
// переменная для хранения текущего разряда
int digit=0;
// для отмеривания 100 мс
unsigned long millis1=0;
// режим 1 - секундомер работает
mode=0;
const int BUTTON=14;     // Контакт 14(A0) 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 кнопки как вход
   pinMode (BUTTON, INPUT);
   // Сконфигурировать контакты как выходы
   for(int i=0;i<8;i++)
   pinMode(pins[i],OUTPUT);
   for(int i=0;i<4;i++)
    {pinMode(pindigits[i],OUTPUT);
    digitalWrite(pindigits[i],HIGH);
    }
   }
void loop()
   {
   tekButton = debounce(prevButton);
   if (prevButton == LOW && tekButton == HIGH) // если нажатие...
      {
      mode=1-mode; // изменение режима
      if(mode==1)
          number=0;
      }
   if(millis()-millis1>=100 && mode==1)
      {millis1=millis1+100;
       number=number+1;
       if(number==10000)
          number=0;
      }
   number1=number;
   for(int i=0;i<4;i++)
      {
      number2=number1%10;
      number1=number1/10;
      showNumber(number2,i);
      for(int j=0;j<4;j++)
         digitalWrite(pindigits[j],HIGH);
      digitalWrite(pindigits[i],LOW);
      delay(1);
      }
   }
// функция вывода цифры на семисегментный индикатор
void showNumber(int num,int dig)
   {
   for(int i=0;i<8;i++)
      {
      if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
```

```
        digitalWrite(pins[i],HIGH);
      else // потушить сегмент
        digitalWrite(pins[i],LOW);
      }
      if(dig==1)    // десятичная точка для второго разряда
        digitalWrite(pins[7],HIGH);
    }
// 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
    }
  }
```

Порядок подключения:

1. Подключаем семисегментный индикатор по схеме на рис. 7.3.
2. Загружаем в плату Arduino скетч из листинга 7.2.
3. Нажатием кнопки запускаем или останавливаем секундомер.

На странице http://arduino-kit.ru/0007 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 7.1, 7.2.

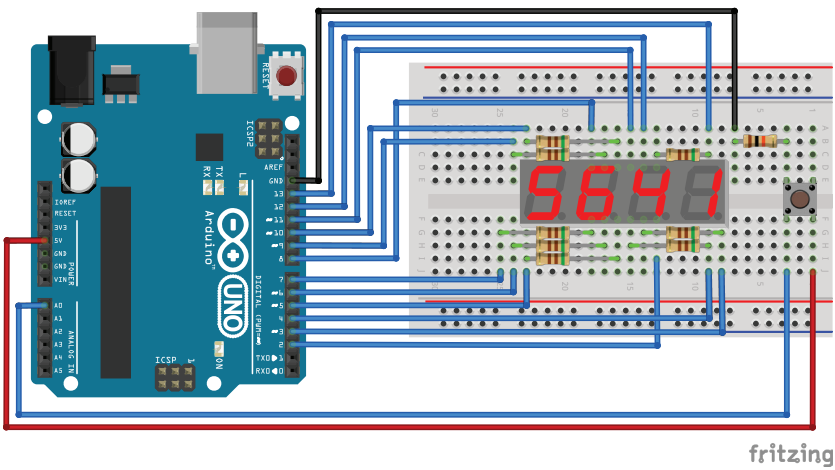

Рис. 7.3. Схема подключения для секундомера

8 Микросхема сдвигового регистра 74HC595. Управляем матрицей из 4 разрядов, экономим выводы Ардуино

В этом эксперименте мы рассмотрим работу Arduino с микросхемой 74HC595 – расширителем выходов, позволяющей уменьшить количество выводов Arduino для управления 4-разрядной семисегментной матрицей.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- 4-разрядная семисегментная матрица;
- микросхема 74HC595;
- резистор 510 Ом – 7 штук;
- провода папа-папа.

Цифровых выводов Arduino Nano и UNO, а иногда даже и Arduino Mega может не хватить, если требуется управлять большим количеством выводов. В этом случае можно использовать микросхему 74HC595. Микросхема 74HC595 – восьмиразрядный сдвиговый регистр с последовательным вводом, последовательным или параллельным выводом информации, с триггером-защелкой и тремя состояниями на выходе. Назначение контактов микросхемы 74HC595 показано на рис. 8.1.

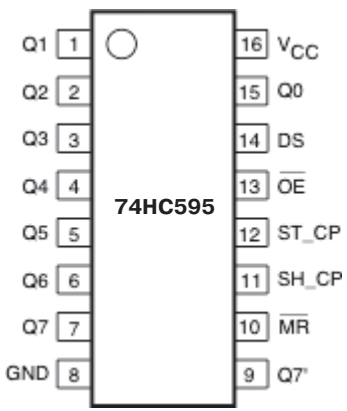

Рис. 8.1. Микросхема 74HC595

Для управления нам вполне достаточно всего лишь трех выводов: SH_CP, ST_CP и DS. Когда на тактовом входе SH_CP появляется логическая единица, регистр считывает бит со входа данных DS и записывает его в самый младший разряд. При поступлении на тактовый вход следующего импульса все повторяется, только бит, записан-

ный ранее, сдвигается на один разряд, а его место занимает вновь пришедший бит. Когда все восемь битов заполнились, и приходит девятый тактовый импульс, регистр снова начинает заполняться с младшего разряда, и все повторяется вновь. Чтобы данные появились на выходах Q0…Q7, нужно их «защелкнуть». Для этого необходимо подать логическую единицу на вход ST_CP. Что бы мы ни делали с регистром, данные на выходах не изменятся, пока мы вновь не «защелкнем» их.

Вывод Q7" предназначен для последовательного (каскадного) соединения сдвиговых регистров. При таком подключении биты из первого регистра будут проталкиваться в следующий в каскаде регистр, из него – в следующий и т. д. Таким образом, каскад из двух 8-битных регистров будет работать как один 16-битный. Можно соединить сколько угодно микросхем.

В следующем эксперименте со светодиодной матрицей мы рассмотрим каскадное подключение микросхем 74HC595, в данном эксперименте мы используем только одну микросхему – восемь выводов микросхемы 74HC595 подключены к восьми сегментам матрицы, четыре вывода матрицы для выбора разряда подключены к четырем выводам платы Arduino. Схема подключения представлена на рис. 8.2.

Приступаем к написанию скетча запуска и останова секундомера 0–999 сек с точностью 0.1 сек. Используем библиотеку Arduino SPI. Поскольку при использовании библиотеки SPI применяются Arduino

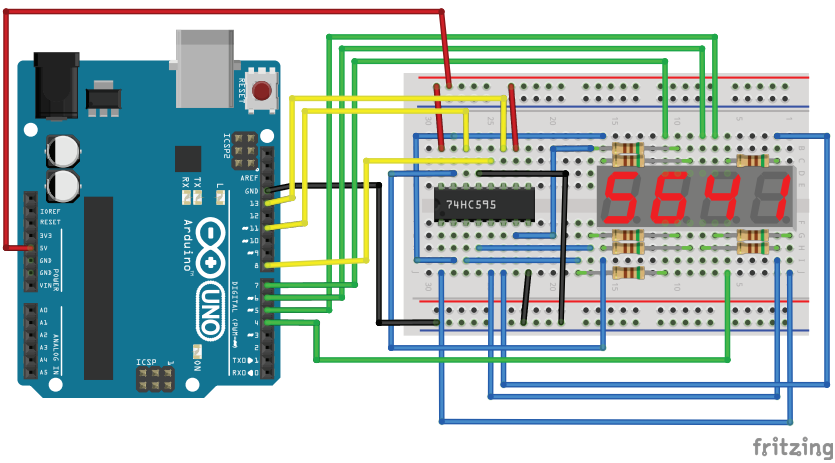

Рис. 8.2. Схема подключения 4-разрядного семисегментного индикатора с использованием сдвиговых регистров 74HC595

выводы 11 и 13, для выбора регистров матрицы используем выводы Arduino 4, 5, 6, 7. Содержимое скетча показано в листинге 8.1.

Листинг 8.1

```
// Подключение библиотеки SPI
#include <SPI.h>
// пин SS
int pin_spi_ss=8;
// значения для вывода цифр 0-9
byte numbers[10] = { B11111100, B01100000, B11011010,
B11110010, B01100110, B10110110,B10111110, B11100000, B11111110,B11110110};
// переменная для хранения значения текущей цифры
int number=0;
int number1=0;
int number2=0;
// семисегментного индикатора
int pindigits[4]={4,5,6,7};
// переменная для хранения текущего разряда
int digit=0;
//
unsigned long millis1=0;

void setup()
   {
   SPI.begin();
   // Сконфигурировать контакты как выходы
   pinMode(pin_spi_ss,OUTPUT);
   for(int i=0;i<4;i++)
      {pinMode(pindigits[i],OUTPUT);
      digitalWrite(pindigits[i],HIGH);
      }
   }

void loop()
   {
   if(millis()-millis1>=100)
      {millis1=millis1+100;
      number=number+1;
      if(number==10000)
         number=0;
      }
   number1=number;
   for(int i=0;i<4;i++)
      {
```

```
      number2=number1%10;
      number1=number1/10;
      showNumber(number2,i);
      for(int j=0;j<4;j++)
         digitalWrite(pindigits[j],HIGH);
      digitalWrite(pindigits[i],LOW);
      delay(1);
      }
   }
// функция вывода цифры на семисегментный индикатор
void showNumber(int num,int dig)
   {
   byte maska;

   digitalWrite(pin_spi_ss,LOW);
   if(dig==1) maska=1;
   else maska=0;
   SPI.transfer(numbers[num]+maska);
   digitalWrite(pin_spi_ss,HIGH);
   }
```

Порядок подключения:

1. Подключаем семисегментный индикатор по схеме на рис. 8.2.
2. Загружаем в плату Arduino скетч из листинга 8.1.
3. Нажатием кнопки запускаем или останавливаем секундомер.

На странице http://arduino-kit.ru/0008 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 8.1.

9 Матрица светодиодная 8x8

В этом эксперименте мы рассмотрим каскадное подключение нескольких микросхем 74HC595, что позволит, используя 3 вывода Arduino, управлять множеством контактов, что будет продемонстрировано в примере вывода фигур на экран светодиодной матрицы 8×8.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- светодиодная матрица 8×8;
- микросхема 74HC595 – 2 штуки;
- провода папа-папа;
- провода папа-мама.

В светодиодных матрицах светодиоды расположены в определенном порядке, а выводы расположены в удобном для монтажа порядке. Светодиодные матрицы бывают одноцветными, двухцветными и RGB. В эксперименте будем использовать одноцветную светодиодную матрицу FYM-23881BUG-11, которая представляет собой набор из 64 светодиодов зеленого цвета, собранных в матрицу 8×8. Расположение выводов матрицы показано на рис. 9.1.

Для подключения светодиодной матрицы к Arduino будем использовать каскадное подключение 2 микросхем 74HC595. При таком подключении биты из первого регистра будут проталкиваться в следующий в каскаде регистр. Нужно подсоединить вывод QH' первого регистра к пину DS (MOSI). Схема соединений показана на рис. 9.2.

Для формирования изображения матрицы будем использовать динамическую индикацию для каждого столбца. Каждые 3 секунды будем менять фигуру для матрицы. Данные фигур хранятся в массиве `figure[]`. Содержимое скетча показано в листинге 9.1.

Листинг 9.1

```
// подключение библиотеки SPI
#include <SPI.h>
int ss_pin=8;             // пин SS
int pos=0;                //
int offfigure=0;          // текущая фигура для отображения
unsigned long millis1=0;
```

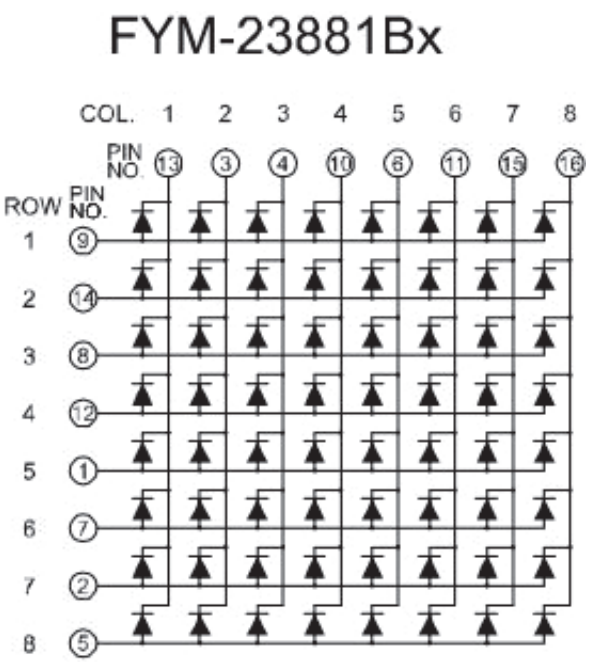

Рис. 9.1. Расположение выводов матрицы FYM-23881BUG-11

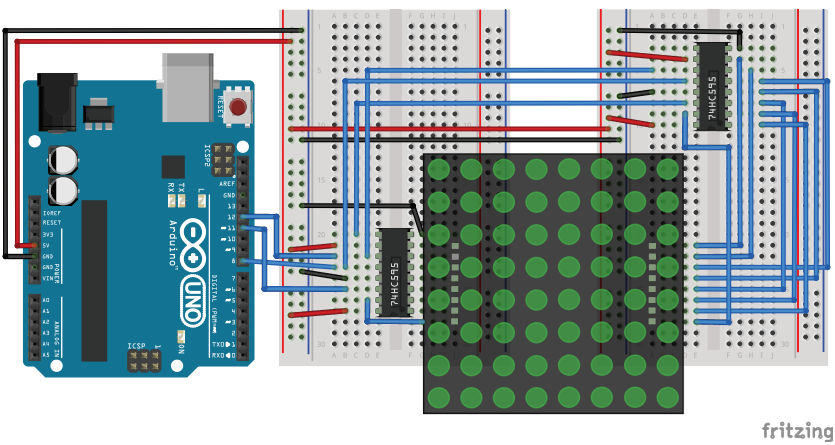

Рис. 9.2. Схема подключения светодиодной матрицы

```
// массив с данными фигур для отображения
byte figure[2][8]={
{B10011001,B10011001,B10011001,B10000001,B10000001,B10011001,B10011001,B10011001},
{B10101010,B10101010,B10101010,B10101010,B10101010,B10101010,B10101010,B10101010}
};
void setup()
   {
   SPI.begin();
   // Сконфигурировать контакт SS как выход
   pinMode(ss_pin, OUTPUT);
   }
void loop()
   {
   digitalWrite(ss_pin, LOW);
   // столбцы
   SPI.transfer(B00000001<<pos);
   // строки
   SPI.transfer(figure[offfigure][pos]);
   digitalWrite(ss_pin,HIGH);     // вывести данные на выводы 74HC595
   delay(1);
   pos=(pos+1)%8;
   if(millis()-millis1>3000)      // через 3 секунды - новая фигура
    {
    offfigure=(offfigure+1)%2;
    millis1=millis();
    }
}
```

Порядок подключения:

1. Подключаем матрицу по схеме на рис. 9.2.
2. Загружаем в плату Arduino скетч из листинга 9.1.
3. Наблюдаем процесс попеременного выводы фигур на экране матрицы.

На странице http://arduino-kit.ru/0009 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 9.1.

10 Пьезоизлучатель. Управляем пьезоизлучателем: меняем тон, длительность, играем Имперский марш

В этом эксперименте мы произведем генерацию звуков на Arduino с помощью пьзоизлучателя.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- пьезоизлучатель;
- резистор 100 Ом;
- провода папа-папа.

Самым простым вариантом генерации звука является использование пьезоизлучателя. Пьезокерамические излучатели (пьезоизлучатели) – электроакустические устройства воспроизведения звука, использующие обратный пьезоэлектрический эффект – возникновение механических деформаций под действием электрического поля. Пьезоизлучатели бывают двух типов – со встроенным генератором и без. Пьезоизлучатели со встроенным генератором излучают фиксированный тональный сигнал сразу после подачи на них номинального напряжения. Они не могут воспроизводить произвольного сигнала. Их обычно используют для простого звукового оповещения. Если требуется проиграть мелодию, то используют пьезоизлучатели без встроенного генератора и генерируют сигнал отдельно. В эксперименте мы используем пьезоизлучатель без встроенного генератора. Схема подключения пьезоизлучателя показана на рис. 10.1.

Приступим к написанию скетча. Для воспроизведения мелодии необходимо подавать последовательно звуки определенной частоты и длительности. Для генерации звуков определенной частоты и длительности будем использовать Arduino-функцию `tone()`:

```
tone(pin,frequency,duration);
```

Функция `tone()` генерирует на выводе прямоугольный сигнал заданной частоты (с коэффициентом заполнения 50%). Функция

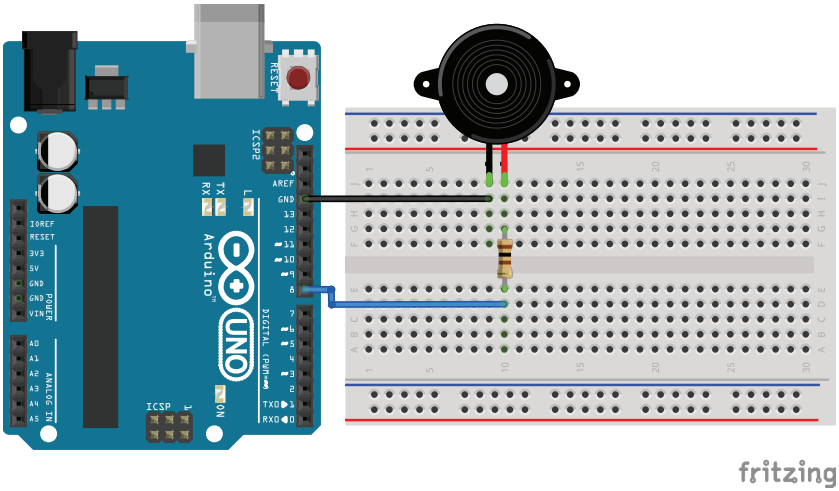

Рис. 10.1. Схема подключения пьезоизлучателя к Arduino

также позволяет задавать длительность сигнала. Если длительность сигнала не указана, он будет генерироваться до тех пор, пока не будет вызвана функция `noTone()`. Значения частот для нот первой и второй октав представлены в табл. 10.1.

Таблица 10.1

1 октава	Обозначение	Частота, Гц	2 октава	Обозначение	Частота, Гц
до	C	261	до	c	523
до-диез	C#(R)	277	до-диез	c#(r)	554
ре	D	293	ре	d	587
ре-диез	D#(S)	311	ре-диез	d#(s)	622
ми	E	329	ми	e	659
фа	F	349	фа	f	698
фа-диез	F#(T)	370	фа-диез	f#(t)	740
соль	G	392	соль	g	784
соль-диез	G#(U)	415	соль-диез	g#(u)	830
ля	A	440	ля	a	880
си-бимоль	B	466	си-бимоль	b	932
си	H	494	си	h	988

Составим мелодию понотно, занесем в массив `melody[]`, список длительностей нот занесем в массив `duration[]`. Данные с обозначе-

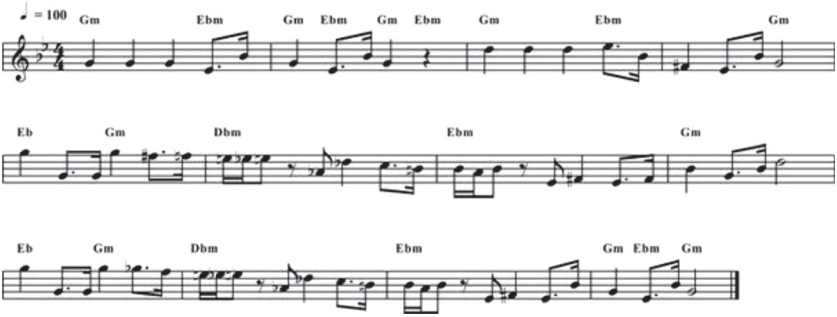

Рис. 10.2. Фрагмент Имперского марша

нием нот занесем в массив `notes[]`, а данные с частотами для соответствующих нот – в массив `frequency[]`. Содержимое скетча представлено в листинге 10.1.

Листинг 10.1

```
// МЕЛОДИЯ - массив нот и массив длительностей
char melody[]={'G','G','G','E','H',
'G','E','H','G','*',
'd','d','d','e','H',
'T','E','H','F',
'g','G','G','g','t','e',
's','s','s','*','U','r','c','B',
'H','A','H','*','E','T','E','F',
'H','G','H','d',
'g','G','G','g','t','f',
's','s','s','*','U','r','c','B',
'H','A','H','*','E','T','E','H',
'G','E','H','G',
'%','%'};
int bb[]={8,8,8,6,2,
8,6,2,8,8,
8,8,8,6,2,
8,6,2,16,
8,6,2,8,6,2,
2,2,4,4,2,8,6,2,
2,2,4,4,2,8,6,2,
8,6,2,16,
8,6,2,8,6,2,
2,2,4,4,2,8,6,2,
2,2,4,4,2,8,6,2,
8,6,2,16,
64,64};
```

```
// подключить динамик к pin 8
int speakerPin = 8;
// темп воспроизведения, ноты, длительности
int tempo,notes,beats;
// процедура проигрыша ноты
void playNote(char note, int duration)
   {
   // массив для наименований нот в пределах двух октав
   char names[]={'c','r','d','s','e','f','t','g','u','a','b',
      'h','C','R','D','S','E','F','T','G','U','A','B', 'H','F'};
   // массив частот нот
   int tones[]={261,277,293,311,329,349,370,392,415,440,466,
       494, 523,554,587,622,659,698,740,784,830,880,932,988};
   // проиграть тон, соответствующий ноте
   for (int i = 0; i < sizeof(tones); i++)
      {
      if (names[i] == note)
         {
         tone(speakerPin,tones[i],duration);
         }
      }
   }
void setup()
   {
   pinMode(speakerPin, OUTPUT);
   tempo=50;    // скорость воспроизведения мелодии
   }
void loop()
   {
   for(int i=0;i<sizeof(melody);i++)
      {
      notes=melody[i];
      beats=bb[i];
      if (notes == '*')
       tone(speakerPin,0, beats*tempo); // пауза
      else
       playNote(notes, beats*tempo);
      // пауза между нотами
      delay(beats*tempo+tempo);
      }
   }
```

Порядок подключения:

1. Подключаем пьезоэлемент к плате Arduino по схеме на рис. 10.1.
2. Загружаем в плату Arduino скетч из листинга 10.1.

3. После загрузки скетча слушаем мелодию на монтажной плате (рис. 10.1).

На странице http://arduino-kit.ru/0010 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 10.1.

11 Транзистор MOSFET. Показываем усилительные качества транзистора. На примере электродвигателя изменяем обороты

В этом эксперименте мы познакомимся с транзистором MOSFET и с помощью него будем управлять мощной нагрузкой – электродвигателем.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- транзистор MOSFET IRF540;
- диод 1N4004;
- двигатель DC;
- потенциометр 10 кОм;
- блок питания 5 В;
- провода папа-папа.

Выводы Arduino, сконфигурированные как OUTPUT, находятся в низкоимпедансном состоянии и могут отдавать 40 мА в нагрузку и не в состоянии обеспечить питание мощной нагрузки и большого напряжения. Одним из способов управления мощной нагрузкой является использование полевых MOSFET-транзисторов. MOSFET-транзистор – это ключ для управления большими токами при помощи небольшого напряжения (в отличие от биполярных транзисторов, управляемых током).

В нашем эксперименте мы будем управлять скоростью вращения мотора изменением напряжения, подаваемого на MOSFET. Управлять напряжением, подаваемым на MOSFET, будем с помощью ШИМ (широтно-импульсной модуляции). В эксперименте 5 мы уже рассматривали использование ШИМ для получения изменяющегося аналогового значения посредством цифровых сигналов. Для регулирования скорости двигателя будем использовать потенциометр. Схема подключения элементов для данного эксперимента показана на рис. 11.1.

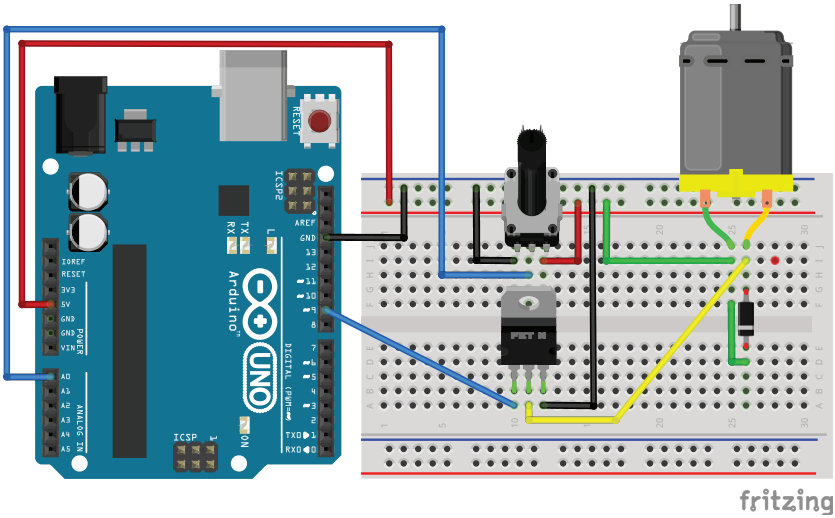

Рис. 11.1. Схема подключения мотора к Arduino

Скетч данного эксперимента показан в листинге 11.1. В цикле `loop()` считываем аналоговое значение потенциометра и, масштабируя функцией `map()`, выдаем ШИМ-сигнал на MOSFET, к которому подключен мотор.

Листинг 11.1

```
const int MOTOR=9;        // Выход для подключения MOSFET
const int POT=0;          // Аналоговый вход A0 для подключения потенциометра
int valpot = 0;           // переменная для хранения значения потенциометра
int speedMotor = 0;       // переменная для хранения скорости двигателя

void setup()
   {
   //
   pinMode(MOTOR,OUTPUT);
   }
void loop()
   {
   valpot = analogRead(POT);   // чтение данных потенциометра
   // масштабируем значение к интервалу 0-255
   speedMotor=map(valpot,0,1023,0,255);
   // устанавливаем новое значение ШИМ
   analogWrite(MOTOR,speedMotor);
   delay(1000);     // пауза
    }
```

Порядок подключения:

1. Подключаем элементы к плате Arduino по схеме на рис. 11.1.
2. Загружаем в плату Arduino скетч из листинга 11.1.
3. Крутим потенциометр – изменяем скорость вращения мотора.

На странице http://arduino-kit.ru/0012 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 11.1.

12 Реле. Управляем реле через транзистор

В этом эксперименте мы познакомимся с реле, с помощью которого с Arduino можно управлять мощной нагрузкой не только постоянного, но и переменного тока.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- биполярный транзистор C945;
- диод 1N4004;
- реле;
- провода папа-папа;
- провода папа-мама.

Реле – это электрически управляемый, механический переключатель, имеет две раздельные цепи: цепь управления, представленная контактами (A1, A2), и управляемая цепь, контакты 1, 2, 3 (см. рис. 12.1). Цепи никак не связаны между собой. Между контактами A1 и A2 установлен металлический сердечник, при протекании тока по которому к нему притягивается подвижный якорь (2). Контакты же 1 и 3 неподвижны. Стоит отметить, что якорь подпружинен, и пока мы не пропустим ток через сердечник, якорь будет прижатым к контакту 3. При подаче тока, как уже говорилось, сердечник превращается в электромагнит и притягивается к контакту 1. При обесточивании пружина снова возвращает якорь к контакту 3.

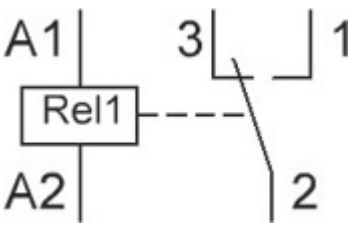

Рис. 12.1. Схема подключения реле к Arduino (n-канальное управление)

При подключении реле к Arduino контакт микроконтроллера не может обеспечить мощность, необходимую для нормальной работы катушки. Поэтому следует усилить ток – поставить транзистор. Для усиления удобнее применять n-p-n-транзистор, включенный по схеме ОЭ (см. рис. 12.2). При таком способе можно подключать нагрузку с большим напряжением питания, чем питание микроконтроллера. Резистор на базе – ограничительный. Может варьироваться в широких пределах (1–10 кОм), в любом случае, транзистор будет работать в режиме насыщения. В качестве транзистора может быть любой n-p-n-транзистор. Коэффициент усиления практически не имеет значения. Выбирается транзистор по току коллектора (нужный нам ток) и напряжению коллектор–эмиттер (напряжение, которым запитывается нагрузка).

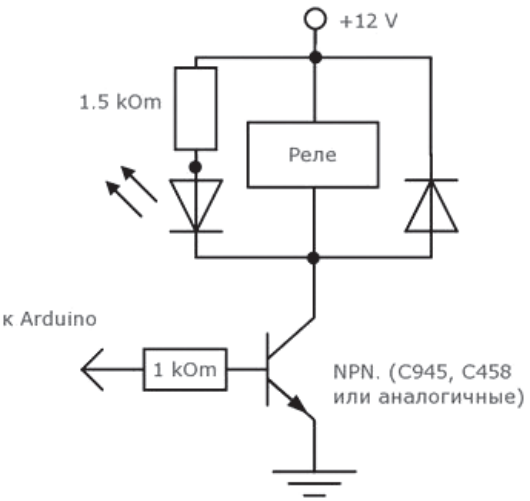

Рис. 12.2. Схема подключения реле к Arduino (n-канальное управление)

Для включения реле, подключенного по схеме с ОЭ, на вывод Arduino необходимо подать 1, для выключения – 0. Подключим реле к плате Arduino по схеме на рис. 12.3 и напишем скетч управления реле. Каждые 5 секунд реле будет переключаться (включаться/выключаться). При переключении реле раздается характерный щелчок. Содержимое скетча показано в листинге 12.1.

Листинг 12.1

```
int relayPin = 10;              // подключение к выводу D10 Arduino
void setup()
   {
   pinMode(relayPin, OUTPUT);    // настроить вывод как выход (OUTPUT)
   }

// функция выполняется циклически бесконечное число раз
void loop()
   {
   digitalWrite(relayPin, HIGH);     // включить реле
   delay(5000);
   digitalWrite(relayPin, LOW);      // выключить реле
   delay(5000);
   }
```

Порядок подключения:

1. Подключаем элементы к плате Arduino по схеме на рис. 12.3.
2. Загружаем в плату Arduino скетч из листинга 12.1.
3. Каждые 5 секунд происходит щелчок переключения реле – если подключить контакты реле, например в разрыв подключенной к сети 220 В патрона с лампой накаливания, то увидим процесс включения/выключения лампы накаливания раз в 5 секунд (рис. 12.3).

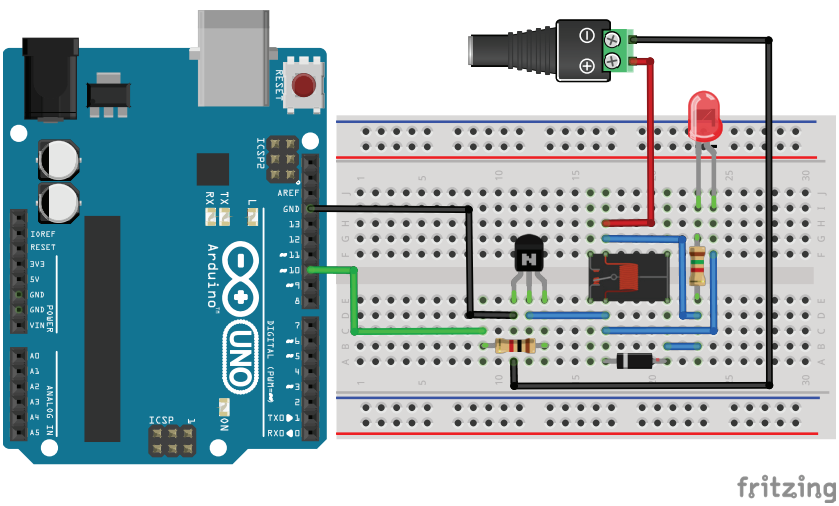

Рис. 12.3

На странице http://arduino-kit.ru/0012 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 12.1.

13 Фоторезистор. Обрабатываем освещенность, зажигая или гася светодиоды

В этом эксперименте мы познакомимся с аналоговым датчиком для измерения освещенности – фоторезистором (рис. 13.1).

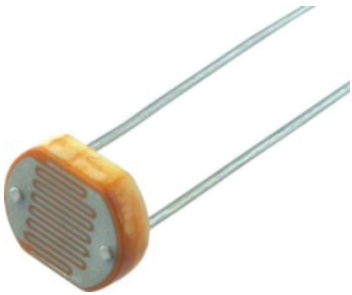

Рис. 13.1. Фоторезистор

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- фоторезистор;
- резистор 10 кОм;
- резистор 220 Ом – 8 штук;
- светодиод – 8 штук;
- провода папа-папа.

Распространенное использование фоторезистора – измерение освещенности. В темноте его сопротивление довольно велико. Когда на фоторезистор попадает свет, сопротивление падает пропорционально освещенности. Схема подключения фоторезистора к Arduino показана на рис. 13.2. Для измерения освещенности необходимо собрать делитель напряжения, в котором верхнее плечо будет представлено фоторезистором, нижнее – обычным резистором достаточно большого номинала. Будем использовать резистор 10 кОм. Среднее плечо делителя подключаем к аналоговому входу A0 Arduino.

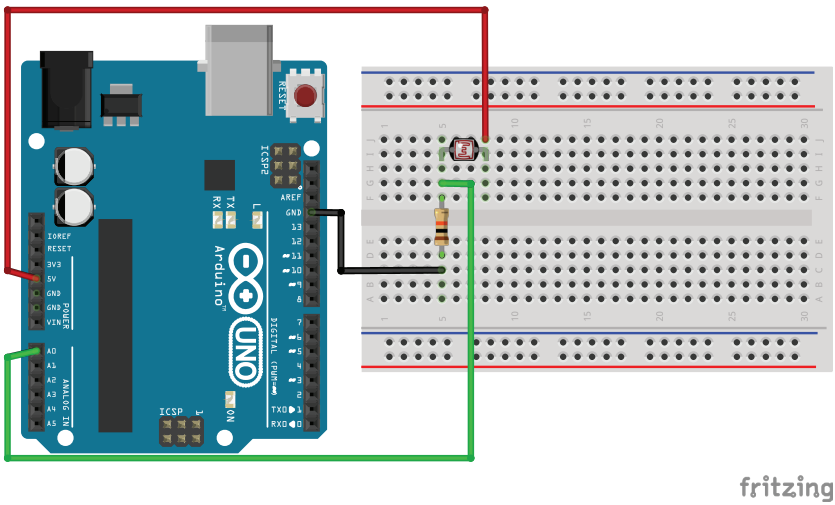

Рис. 13.2. Схема подключения фоторезистора к Arduino

Напишем скетч чтения аналоговых данных и отправки их в последовательный порт. Содержимое скетча показано в листинге 13.1.

Листинг 13.1

```
int light;        // переменная для хранения данных фоторезистора
void setup()
   {
   Serial.begin(9600);
   }

void loop()
   {
   light = analogRead(0);
   Serial.println(light);
   delay(100);
   }
```

Порядок подключения:

1. Подключаем фоторезистор по схеме на рис. 13.2.
2. Загружаем в плату Arduino скетч из листинга 13.1.
3. Регулируем рукой освещенность фоторезистора и наблюдаем вывод в последовательный порт изменяющихся значений, запоминаем показания при полной освещенности помещения и при полном перекрывании светового потока.

Теперь создадим индикатор освещенности с помощью светодиодного ряда из 8 светодиодов. Количество горящих светодиодов пропорционально текущей освещенности. Собираем светодиоды по схеме на рис. 13.3, используя ограничительные резисторы номиналом 220 Ом.

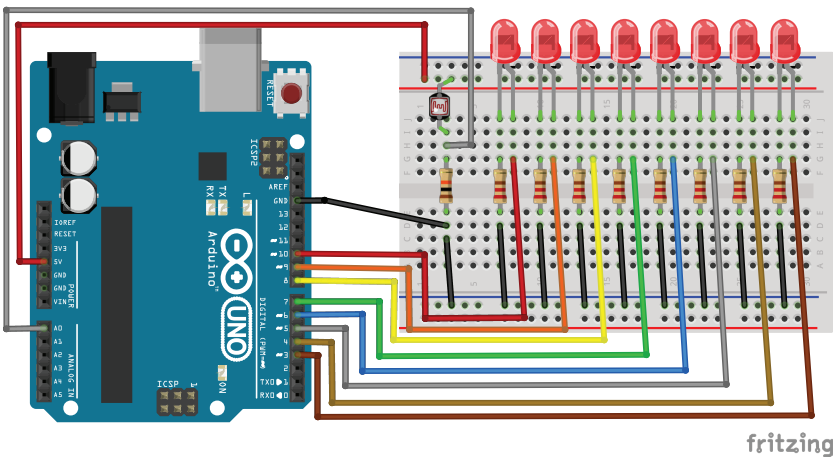

Рис. 13.3. Схема подключения фоторезистора и светодиодов к Arduino

Содержимое скетча для отображения текущей освещенности на линейке светодиодов показано в листинге 13.2.

Листинг 13.2

```
// Контакт подключения светодиодов
const int leds[]={3,4,5,6,7,8,9,10};
const int LIGHT=A0;          // Контакт A0 для входа фоторезистора
const int MIN_LIGHT=200;     // Нижний порог освещенности
const int MAX_LIGHT=900;     // верхний порог освещенности
// Переменная для хранения данных фоторезистора
int val = 0;

void setup()
  {
  // Сконфигурировать контакты светодиодов как выход
  for(int i=0;i<8;i++)
      pinMode(leds[i],OUTPUT);
  }
void loop()
```

```
{
val = analogRead(LIGHT);   // Чтение показаний фоторезистора
// Применение функции map()
val = map(val, MIN_LIGHT, MAX_LIGHT, 8, 0);
// ограничиваем, чтобы не превысило  границ
val = constrain(val, 0, 8);
// зажечь кол-во светодиодов, пропорциональное освещенности,
// остальные потушить
for(int i=1;i<9;i++)
   {
    if(i>=val)              // зажечь светодиоды
        digitalWrite(leds[i-1],HIGH);
    else                    // потушить светодиоды
   digitalWrite(leds[i-1],LOW);
   }
delay(1000);                // пауза перед следующим измерением
}
```

Нижний и верхний пределы освещенности мы берем из запомненных значений при проведении эксперимента по предыдущему скетчу (листинг 13.1). Промежуточное значение освещенности мы масштабируем на 8 значений (8 светодиодов) и зажигаем количество светодиодов пропорциональное значению между нижней и верхней границами.

Порядок подключения:

1. Подключаем фоторезистор и светодиоды по схеме на рис. 13.3.
2. Загружаем в плату Arduino скетч из листинга 13.2.
3. Регулируем рукой освещенность фоторезистора и по количеству горящих светодиодов определяем текущий уровень освещенности (рис. 13.3).

На странице http://arduino-kit.ru/0013 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 13.1, 13.2.

14 Датчик температуры аналоговый LM335. Принцип работы, пример работы

В этом эксперименте мы познакомимся с аналоговым датчиком для измерения температуры LM335.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- датчик температуры LM335;
- резистор 2,2 кОм;
- RGB-светодиод;
- резистор 220 Ом – 3 штуки;
- провода папа-папа.

LM335 – это недорогой температурный чувствительный элемент с диапазоном от –40 °C до +100 °C и точностью в 1 °C. По принципу действия датчик LM335 представляет собой стабилитрон, у которого напряжение стабилизации зависит от температуры. При повышении температуры на один градус Кельвина напряжение стабилизации увеличивается на 10 милливольт. Для измерения температуры используются 2 вывода, третий нужен для калибровки датчика. В качестве примера использования датчика LM335 создадим индикатор температуры окружающей среды на RGB-светодиоде. Схема подключения показана на рис. 14.1.

Приступим к написанию скетча. Нам необходимо получить значение с аналогового входа A0 подключения датчика LM335, перевести в значение в вольтах, исходя из значения опорного напряжения +5 В. Мы получим значение температуры в Кельвинах. Для получения значения в градусах Цельсия полученное значение необходимо уменьшить на величину 273.15. Определим комфортное значение температуры в интервале MIN_T–MAX_T (20–27 °C). При попадании значения в этот интервал RGB-светодиод горит желтым цветом, при пониженном значении – синим, при повышенном – красным. Для проверки выводим значение температуры в монитор последовательного порта Arduino IDE.

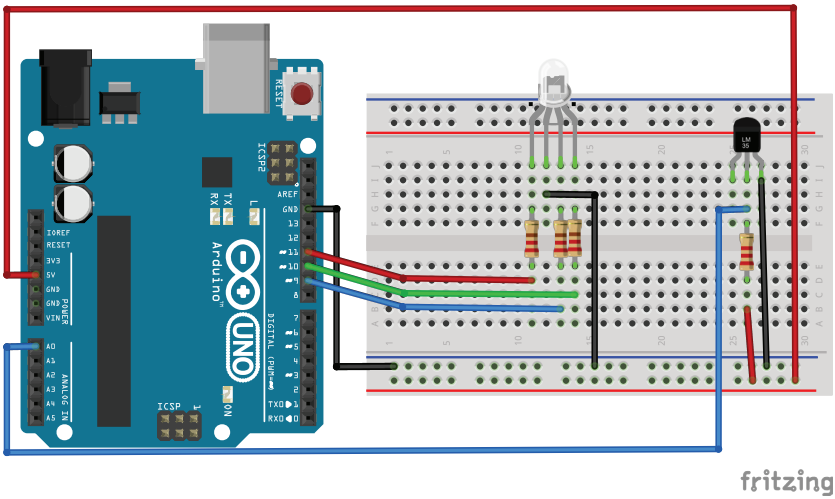

Рис. 14.1. Схема подключения датчика температуры и RGB-светодиода к Arduino

Листинг 14.1

```
const int BLUE=9;     // Вывод BLUE RGB-светодиода
const int GREEN=10;   // Вывод GREEN RGB-светодиода
const int RED=11;     // Вывод RED RGB-светодиода
const int lm335=A0;   // для подключения LM335

int MIN_T=20;         // Нижний порог
int MAX_T=30;         // Верхний порог
int val = 0;

void setup()
  {
    // конфигурируем выводы светодиоды как OUTPUT
    pinMode(RED,OUTPUT);
    pinMode(GREEN,OUTPUT);
    pinMode(BLUE,OUTPUT);
}
void loop()
  {
    double val = analogRead(lm335);          // чтение
    double voltage = val*5.0/1024;           // перевод в вольты
    double temp = voltage*100 - 273.15;      // в градусы Цельсия
    Serial.print(" temp =   ");
```

```
   Serial.println(temp);
   if(temp < MIN_T)          // синий цвет RGB-светодиода
      setRGB(0,0,1);
   else if(temp > MIN_T)     // красный цвет RGB-светодиода
      setRGB(1,0,0);

   else                      // желтый цвет RGB-светодиода
      setRGB(1,0,0);
   delay(1000);              // пауза перед следующим измерением
  }
// установка цвета RGB-светодиода
void setRGB(int r, int g, int b)
   {
   digitalWrite(RED,r);
   digitalWrite(GREEN,g);
   digitalWrite(BLUE,b);
   }
```

Порядок подключения:

1. Подключаем датчик LM335 и RGB-светодиод по схеме на рис. 14.1.
2. Загружаем в плату Arduino скетч из листинга 14.1.
3. Смотрим в мониторе последовательного порта Arduino IDE вывод значений температуры, RGB-светодиод показывает интервал комфортности температуры.

На странице http://arduino-kit.ru/0014 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 14.1.

15 Индикатор LCD1602. Принцип подключения, вывод информации на него

В этом эксперименте мы познакомимся с жидкокристаллическими индикаторами Winstar для вывода символьной информации. Научимся в Arduino-проектах применять библиотеки и создадим проект вывода показаний датчика температуры LM335 на экран дисплея.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- LCD-экран WH1602;
- резистор 2,2 кОм;
- резистор 51 Ом;
- потенциометр 1 кОм;
- датчик температуры LM335;
- провода папа-папа;
- внешний блок питания +5 В.

Жидкокристаллические индикаторы (ЖКИ, англ. LCD) являются удобным и недорогим средством для отображения данных ваших проектов. Символьный индикатор WH1602 позволяет выводить на экран 2 строки по 16 символов (размером 5×7 или 5×10 и дополнительная строка под курсор). Управляет работой дисплея контроллер. На рис. 15.1 показан ЖКИ Winstar с контроллером HD44780.

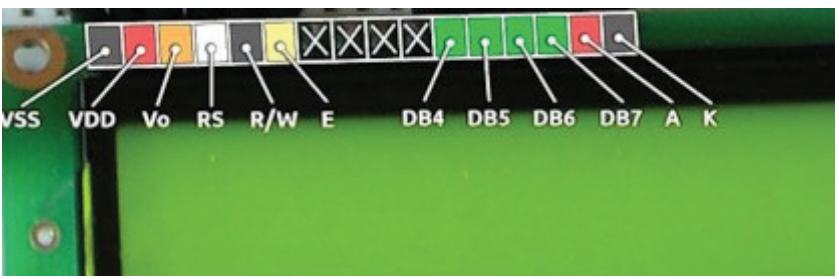

Рис. 15.1. ЖКИ WH1602 на контроллере HD44780

Назначение выводов контроллера:

- DB0–DB7 – отвечают за входящие/исходящие данные;
- RS – высокий уровень означает, что сигнал на выходах DB0–DB7 является данными, низкий – командой;
- R/W – определяет направление данных (чтение/запись). Так как операция чтения данных из индикатора обычно бывает невостребованной, то можно установить постоянно на этом входе низкий уровень;
- E – импульс длительностью не менее 500 мс на этом выводе определяет сигнал для чтения/записи данных с выводов DB0–DB7, RS и W/R;
- V0 – используется для задания контраста изображения;
- A, K – питание подсветки (анод и катод), если она имеется;
- VSS – земля;
- VDD – питание ЖК-индикатора.

Для управления ЖК-индикатором необходимо 6 или 10 выводов Arduino, в зависимости от того, выбран 4- или 8-битный режим обмена данными. Для сокращения требуемого числа выводов микроконтроллера можно работать в 4-битном режиме. В этом случае на выводах DB4–DB7 индикатора сначала будут передаваться старшие четыре бита данных/команды, затем – младшие четыре бита. Выводы DB0–DB3 останутся незадействованными.

В нашем эксперименте мы будем считывать данные с датчика температуры LM335, который мы рассмотрели в эксперименте 13, и выводить на экран ЖКИ значение температуры в Кельвинах и градусах Цельсия. Схема подключения датчика температуры и ЖКИ в 4-битном режиме к плате Arduino показана на рис. 15.2. Заметьте, что для питания ЖКИ нужен отдельный блок питания +5 В.

Приступим к написанию скетча. Функционал Arduino может быть расширен за счет использования библиотек. Библиотеки Arduino предоставляют дополнительную функциональность для использования в скетчах и сильно упрощают процесс написания программ. Ряд основных библиотек устанавливается вместе со средой Arduino IDE, а дополнительные, которых очень много, вы можете установить сами. При работе Arduino с ЖКИ-дисплеями на контроллере HD44780 будем использовать библиотеку LiquidCrystal. Для подключения библиотеки в начале скетча вставляем строку

```
#include <LiquidCrystal.h>
```

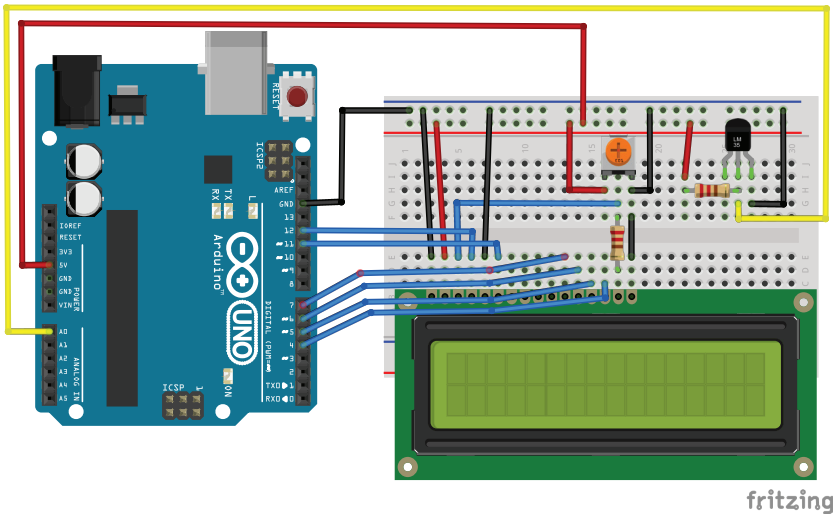

Рис. 15.2. Схема подключения датчика температуры и ЖКИ к Arduino

Затем создаем переменную типа LiquidCrystal

```
LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
```

где 12, 11, 7, 6, 5, 4 – номера контактов RS, E, D4, D5, D6, D7.

В `setup()` запускаем функцию `lcd.begin()`, определяющую размерность индикатора, для установки курсора в определенную позицию – `lcd.setCursor()`, для вывода информации на экран дисплея – `lcd.print()`. Содержимое данного скетча показано в листинге 15.1.

Листинг 15.1

```
// Подключение библиотеки
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
const int LM335=A0;    // для подключения LM335

void setup() {
   // установить размерность дисплея
   lcd.begin(16, 2);
}

void loop()
 {
```

```
  double val = analogRead(LM335);          // чтение
  double voltage = val*5.0/1024;           // перевод в вольты
  // вывод значения в Кельвинах
  lcd.setCursor(2,0);
  lcd.print("Tk="); lcd.print(voltage*100); lcd.print("K");
  double temp = voltage*100 - 273.15;     // в градусы Цельсия
  // вывод значения в градусах Цельсия
  lcd.setCursor(2,1);
  lcd.print("Tc="); lcd.print(temp); lcd.print("");
  delay(1000);       // пауза перед следующим измерением
}
```

Порядок подключения:

1. Подключаем датчик LM335 и ЖКИ по схеме на рис. 15.2.
2. Загружаем в плату Arduino скетч из листинга 15.1.
3. Смотрим на экране дисплея показания датчика температуры в Кельвинах и градусах Цельсия (рис. 15.2).

На странице http://arduino-kit.ru/0015 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 15.1.

16 Графический индикатор на примере Nokia 5110

В этом эксперименте мы рассмотрим графический дисплей Nokia 5110, который можно использовать в проектах Arduino для вывода графической информации.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- графический дисплей Nokia 5110;
- провода папа-папа;
- фоторезистор;
- резистор 10 кОм.

Жидкокристаллический дисплей Nokia 5110 – монохромный дисплей с разрешением 84×48 на контроллере PCD8544, предназначен для вывода графической и текстовой информации. Питание дисплея должно лежать в пределах 2.7–3.3 В (максимум 3.3 В, при подаче 5 В на вывод VCC дисплей может выйти из строя). Но выводы контроллера толерантны к +5 В, поэтому их можно напрямую подключать к входам Arduino. Немаловажный момент – низкое потребление, что позволяет

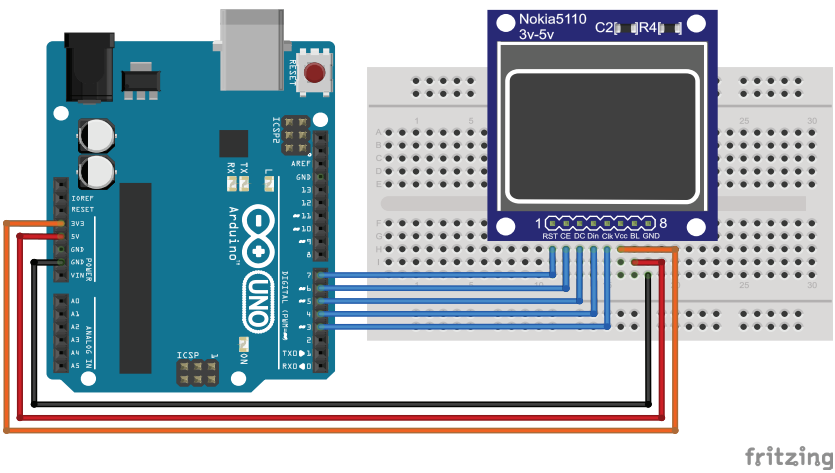

Рис. 16.1. Схема подключения Nokia 5110 к Arduino

питать дисплей от платы Arduino без внешнего источника питания. Схема подключения Nokia 5110 к Arduino показана на рис. 16.1.

Для работы с дисплеем Nokia 5110 будем использовать библиотеку Adafruit_GFX, которая имеет богатые возможности для вывода графики и текста. В нашем эксперименте мы будем получать данные освещенности с фоторезистора, подключенного к аналоговому входу Arduino A0, и выводить данные освещенности в числовом и графическом представлениях. Схема подключения показана на рис. 16.2.

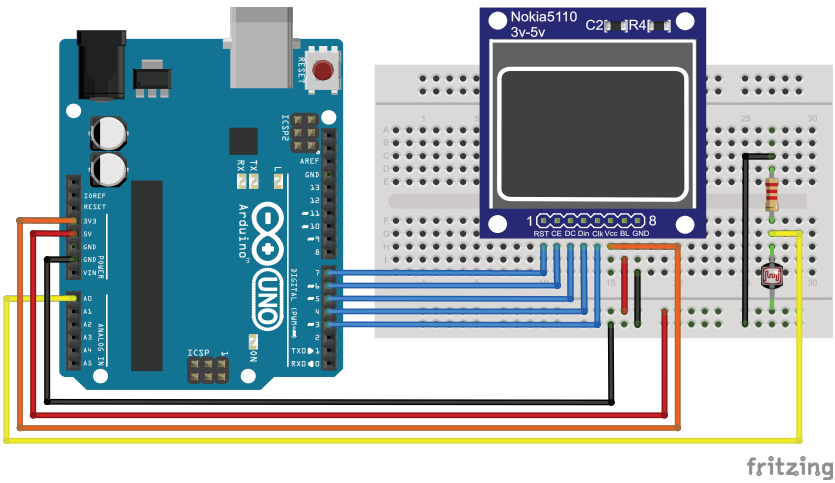

Рис. 16.2. Схема подключения Nokia 5110 и фоторезистора к Arduino

Код скетча нашего эксперимента показан в листинге 16.1. Мы считываем данные с фоторезистора и отображаем числовое значение, а также в графическом виде (прогресс-бар) значение освещенности в процентах от максимального значения. Значения минимальной и максимальной освещенности берем из эксперимента 13.

Листинг 16.1

```
// Подключение библиотеки
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

// PIN 7 - RST   Pin 1 on LCD
// PIN 6 - CE    Pin 2 on LCD
// PIN 5 - DC    Pin 3 on LCD
// PIN 4 - DIN   Pin 4 on LCD
```

```
// PIN 3 - CLK  Pin 5 on LCD

Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);
const int LIGHT=A0;      // Контакт A0 для входа фоторезистора
const int MIN_LIGHT=200; // Нижний порог освещенности
const int MAX_LIGHT=900; // Верхний порог освещенности
// Переменная для хранения данных фоторезистора

int val1,val2 = 0;
void setup()
   {
   display.begin();
   // установить контраст фона экрана
   // очень важный параметр!
   display.setContrast(60);
   display.clearDisplay();       // очистить экран
   delay(2000);
}
void loop()
   {
   val1 = analogRead(LIGHT);    // Чтение показаний фоторезистора
   drawText(val1,1);            // вывести текст
   // масштабирование значения потенциометра к 0-75
   val2= map(val1, MIN_LIGHT, MAX_LIGHT, 0, 75);
   // вывод черного прямоугольника в %
   display.fillRect(5, 25, val2, 10, 1);
   // вывод белой части прямоугольника
   display.fillRect(5+val2,25, 75-val2, 10, 0);
   display.display();
   delay(1000);                 // пауза перед новым измерением
   drawText(val1,2);            // стереть текст
}
// процедура вывода текста
void drawText(unsigned long num,int color)
   {
   display.setTextSize(2);      // размер шрифта
   display.setCursor(20,5);     // позиция курсора
   if(color==1)
   display.setTextColor(BLACK); // вывести значение
   else
   display.setTextColor(WHITE); // стереть (белым по белому)
   display.print(num);
}
```

Порядок подключения:

1. Подключаем датчик дисплея Nokia 5110 и фоторезистор по схеме на рис. 16.2.
2. Загружаем в плату Arduino скетч из листинга 16.1.

3. Перекрывая рукой поток света, смотрим на экране дисплея изменение показаний освещенности.

На странице http://arduino-kit.ru/0016 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 16.1, и библиотеку Adafruit_GFX для работы с дисплеем Nokia 5110.

17 Сервопривод. Крутим потенциометр, меняем положение

В этом эксперименте мы рассмотрим популярный сервопривод SG90 и создадим проект, реализующий поворот выходного вала сервопривода в зависимости от сигнала с потенциометра.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- сервопривод SG90;
- потенциометр 1 кОм;
- провода папа-папа;
- внешний блок питания +5 В.

Сервопривод (см. рис. 17.1) – устройство, обеспечивающее преобразование сигнала в строго соответствующее этому сигналу перемещение (как правило, поворот) исполнительного устройства. Представляет собой прямоугольную коробку с мотором, схемой и редуктором внутри и выходным валом, который может поворачиваться на строго фиксированный угол, определяемый входным сигналом. Как правило, этот угол имеет предел от 60 до 180 градусов. Кроме этого, еще бывают сервоприводы и постоянного вращения.

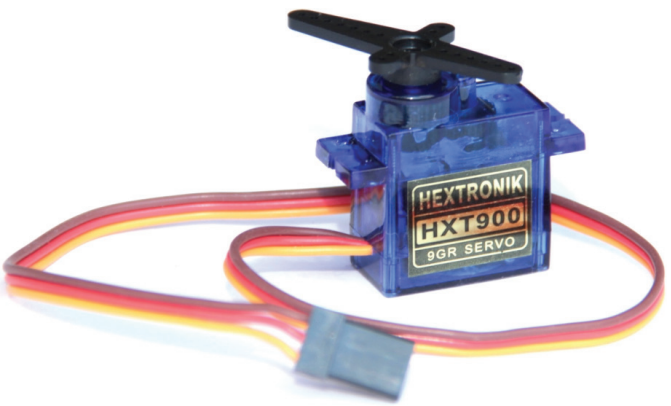

Рис. 17.1. Сервопривод

Сервопривод подключается с помощью трех проводов к управляющему устройству (драйверу или контроллеру) и источнику питания. Сервопривод управляется с помощью импульсов переменной длительности. Угол поворота определяется длительностью импульса, который подается по сигнальному проводу. Это называется широтно-импульсной модуляцией. Сервопривод ожидает импульса каждые 20 мс. Длительность импульса определяет, насколько далеко должен поворачиваться мотор. Например, импульс в 1,5 мс диктует мотору поворот в положение 90° (нейтральное положение).

Когда сервопривод получает команду на перемещение, его управляющий орган перемещается в это положение и удерживает его. Если внешняя сила действует на сервопривод, когда он удерживает заданное положение, сервопривод будет сопротивляться перемещению из этого положения. Максимальная величина силы, которую может выдерживать сервопривод, характеризует вращающий момент сервопривода. Однако сервопривод не навсегда удерживает свое положение, импульсы позиционирования должны повторяться, информируя сервопривод о сохранении положения.

В нашем эксперименте мы будем управлять положением сервопривода с помощью потенциометра. Схема подключения сервопривода и потенциометра к плате Arduino показана на рис. 17.2. Сервопривод подключается тремя проводами: питание (Vcc), «земля» (Gnd) и сигнальный (C). Питание– красный провод, он может быть подключен к +5 В внешнего источника питания, черный (или коричневый) провод – «земля» – подключается к GND-выводу Arduino GND, сигнальный (оранжевый/желтый/белый) провод подключается к цифровому выводу контроллера Arduino. Для питания сервопривода используем отдельный блок питания +5 В.

Для управления сервоприводом в Arduino имеется стандартная библиотека Servo. На платах, отличных от Mega, использование библиотеки отключает возможность применения `analogWrite()` (ШИМ) на пинах 9 и 10 (вне зависимости, подключены к этим пинам сервы или нет). На платах Mega до 12 сервоприводов могут использоваться без влияния на функциональность ШИМ, но использование от 12 до 23 сервомашинок отключит PWM ШИМ на пинах 11 и 12.

Аналоговые данные потенциометра (0–1023) масштабируем функцией `map()` в значения угла поворота сервопривода (0–180) и с помощью библиотечной функции servo.write(angle) даем сервоприводу команду для поворота. Скетч приведен в листинге 17.1.

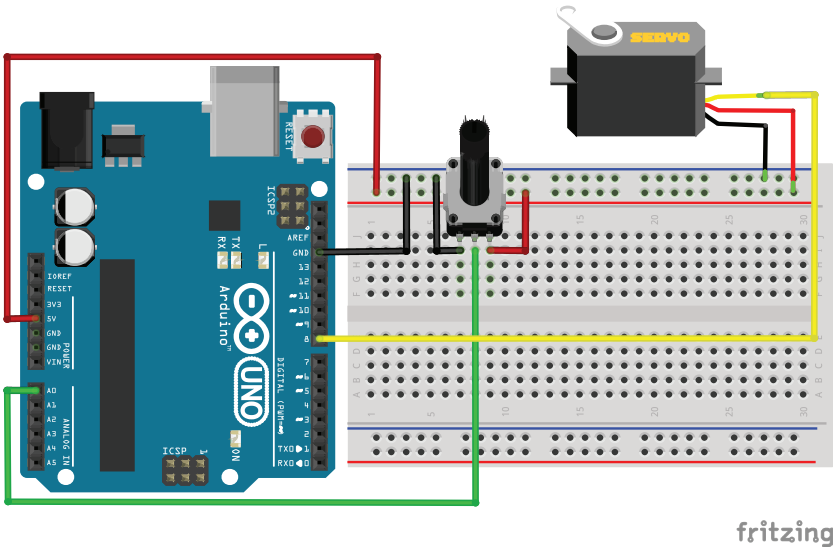

Рис. 17.2. Схема подключения сервопривода и потенциометра к Arduino

Листинг 17.1

```
#include <Servo.h>        // подключение библиотеки Servo
Servo servo1;
const int pinServo=8;     // Пин для подключения сервопривода
const int POT=0;          // Аналоговый вход A0 для подключения потенциометра
int valpot = 0;           // переменная для хранения значения потенциометра
int angleServo = 0;       // переменная для хранения угла поворота сервы

void setup()
   {
   // подключить переменную servo к выводу pinServo
   servo1.attach(pinServo);
   }
void loop()
   {
   valpot = analogRead(POT);   // чтение данных потенциометра
   // масштабируем значение к интервалу 0-180
   angleServo=map(valpot,0,1023,0,180);
   // поворот сервопривода на полученный угол
   servo1.write(angleServo);
   delay(15);     // пауза для ожидания поворота сервопривода
}
```

Порядок подключения:

1. Подключаем датчик сервопривода и потенциометр по схеме на рис. 17.2.
2. Загружаем в плату Arduino скетч из листинга 17.1.
3. Поворотом ручки потенциометра управляем положением сервопривода.

На странице http://arduino-kit.ru/0017 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 17.1.

18 Джойстик. Обрабатываем данные от джойстика. Управление Pan/Tilt Bracket с помощью джойстика

В этом эксперименте мы рассмотрим подключение к Arduino двухосевого аналогового джойстика.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- модуль джойстика;
- кронштейн Pan/Titl Bracket;
- сервопривод – 2 шт.;
- провода папа-папа.

Для управления каким-либо устройством на основе Arduino, перемещающимся в двухмерной системе координат, отлично подойдет джойстик. Для плат Arduino существуют модули аналогового джойстика, имеющие ось X, Y (потенциометры 10 кОм) и дополнительную кнопку – ось Z. Джойстик позволяет плавно и точно отслеживать степень отклонения от нулевой точки. Сам джойстик подпружиненный, поэтому он будет возвращаться в центральное состояние после его отпускания из определенной позиции.

Контакты Vcc и GND между всеми тремя группами контактов соединены. Таким образом, для подключения нужно 5 проводов: ось X, ось Y, кнопка Z, питание Vcc и общий GND. Джойстики – пассивные модули и не потребляют какую-либо энергию от платы Arduino. Выводы VERT и HORZ подключаются к аналоговым входам A0 и A1 Arduino, SEL – к цифровому входу D2. Схема подключения показана на рис. 18.1.

Напишем скетч считывания данных джойстика и вывода значений в монитор последовательного порта Arduino IDE. Содержимое скетча показано в листинге 18.1.

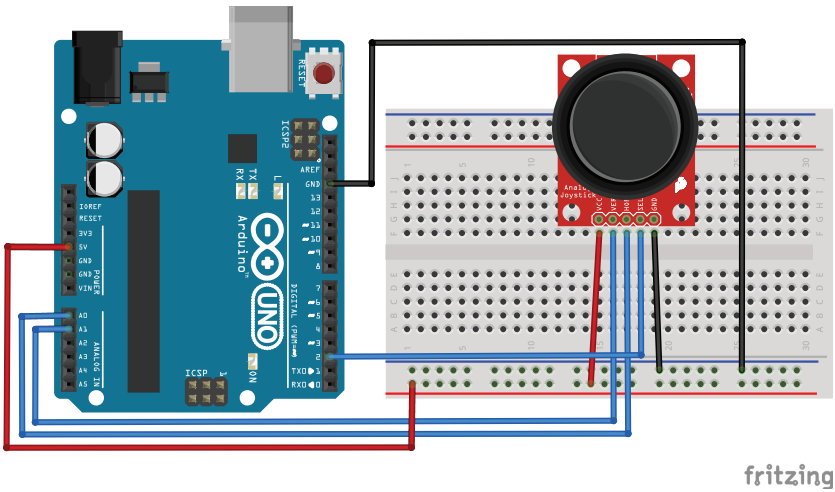

Рис. 18.1. Схема подключения платы джойстика к Arduino

Листинг 18.1

```
const int axisX=A0                // ось X подключена к A0
const int axisY=A1                // ось Y подключена к A1
const int axisZ=2                 // ось Z (кнопка джойстика) подключена к D2
int valX, valY, valZ = 0;         // переменные для хранения значений осей

void setup()
   {
   pinMode(axis_Z, INPUT_PULLUP); // конфигурируем D2 как INPUT с включением
                                  // подтягивающего резистора внутри процессора
   Serial.begin(9600);
  }
void loop() {
   valX = analogRead(axisX);      //   значение оси X
   valY = analogRead(axisY);      // значение оси Y
   valZ = 1-digitalRead(axisZ);   // значение оси Z (кнопка)
   // выводим значения в монитор
   Serial.print("X:");Serial.print(valX, DEC);
   Serial.print(" | Y:");Serial.print(valY, DEC)
   Serial.print(" | Z: ");Serial.println(valZ, DEC);
   delay(500);                    // пауза перед следующим считыванием данных
  }
```

Порядок подключения:

1. Подключаем джойстик к плате Arduino по схеме на рис. 18.2.

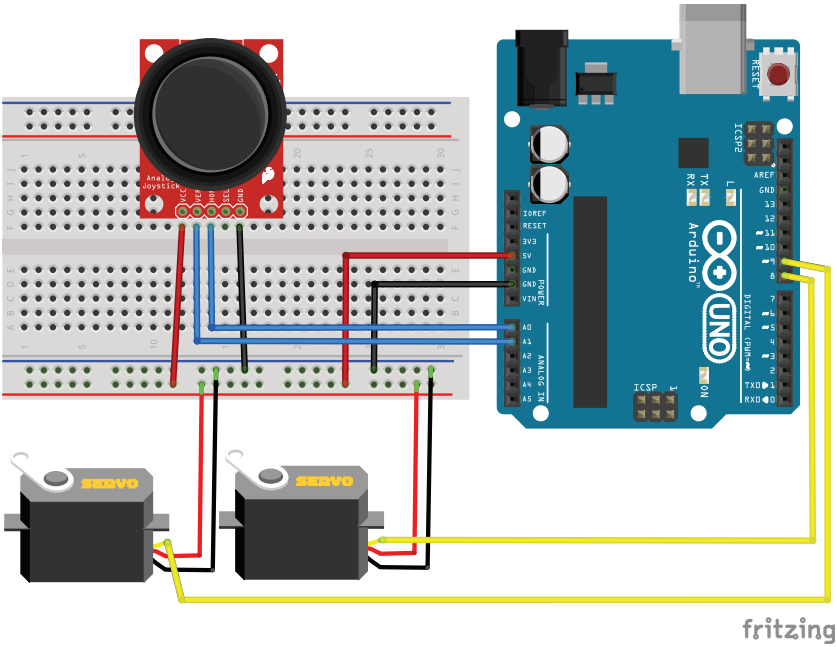

Рис. 18.2. Схема подключения платы джойстика и Pan/Title Bracket к Arduino

2. Загружаем в плату Arduino скетч из листинга 18.1.
3. Смотрим в мониторе последовательного порта Arduino IDE вывод значений смещения по осям X и Y и состояние кнопки (ось Z).

Создадим более понятный пример использования джойстика – для управления положением кронштейна Pan/Titl Bracket с двумя сервоприводами, на котором можно разместить, например, камеру и менять положение камеры влево/вправо и вниз/вверх с помощью джойстика. Схема соединений для данного эксперимента показана на рис. 18.2.

Перемещением джойстика по оси X мы будем управлять поворотом нижнего сервопривода (влево/вправо), перемещением джойстика по оси Y будем управлять поворотом верхнего сервопривода (вверх/вниз). Среднее нейтральное положение джойтика по каждой оси (при аналоговом значении 512) соответствует углу поворота сервопривода на угол 90°. Содержимое скетча показано в листинге 18.2.

Листинг 18.2

```
#include <Servo.h>        // подключение библиотеки Servo
Servo servo1, servo2;

const int pinServo1=8;   // Пин для подключения 1 сервопривода
const int pinServo2=9;   // Пин для подключения 2 сервопривода
// переменные для хранения углов поворота сервоприводов
int angleServo1,angleServo2 = 0;
const int axisX=A0;        // ось X подключена к A0
const int axisY=A1;        // ось Y подключена к A1
int valX, valY = 0;       // переменные для хранения значений осей

void setup()
   {
   // подключить переменную servo1 к выводу pinServo1
   servo1.attach(pinServo1);
   // подключить переменную servo2 к выводу pinServo2
   Servo2.attach(pinServo2);
   }
void loop()
   {
   valX = analogRead(axisX);     //   значение оси X
   valY = analogRead(axisY);     // значение оси Y
   // масштабируем значение к интервалу 0-180
   angleServo1=map(valX,0,1023,0,180);
   angleServo2=map(valY,0,1023,0,180);
   // поворот сервоприводов на полученный угол
   servo1.write(angleServo1);
   servo2.write(angleServo2);
   delay(15);     // пауза для ожидания поворота сервоприводов
}
```

Порядок подключения:

1. Собираем Pan/Title Bracket и сервоприводы.
2. Подключаем джойстик и Pan/Title Bracket к плате Arduino по схеме на рис. 18.2.
3. Загружаем в плату Arduino скетч из листинга 18.2.
4. Управляем положением Pan/Title Bracket перемещением джойстика по осям X и Y.

На странице http://arduino-kit.ru/0018 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 18.1, 18.2.

19 Шаговый двигатель 4-фазный, с управлением на ULN2003 (L293)

В этом эксперименте мы рассмотрим подключение к Arduino шагового двигателя.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- шаговый двигатель;
- микросхема L293;
- провода папа-папа;
- кнопка – 2 шт.;
- резистор 1 кОм – 2 шт.;
- внешний блок питания +5 В.

Шаговые двигатели представляют собой электромеханические устройства, задачей которых является преобразование электрических импульсов в перемещение вала двигателя на определенный угол. ШД нашли широкое применение в области, где требуется высокая точность перемещений или скорости. Наглядными примерами устройств с ШД могут служить принтеры, факсы и копировальные машины, а также более сложные устройства: станки с ЧПУ (числовым программным управлением), фрезерные, гравировальные машины и т. д.

Шаговый двигатель – синхронный бесщеточный электродвигатель с несколькими обмотками, в котором ток, подаваемый в одну из обмоток статора, вызывает фиксацию ротора. Последовательная активация обмоток двигателя вызывает дискретные угловые перемещения (шаги) ротора. Напрямую, к выводам Arduino подключать ШД нельзя, для подключения используют либо драйверы шаговых двигателей (например, A4988), либо драйверы двигателей постоянного тока (ULN2003, L293). В эксперименте будем использовать микросхему L393, которая содержит в себе четыре мощных усилителя (см. рис. 19.1). Если на вход усилителя подается 1, то выход сажается на 12 В, если на вход подается 0, то вывод сажается на землю. Таким образом, подавая комбинации 0 и 1 на разные входы,

можно сажать выводы двигателя на шины разной полярности, вращая движок в разные стороны.

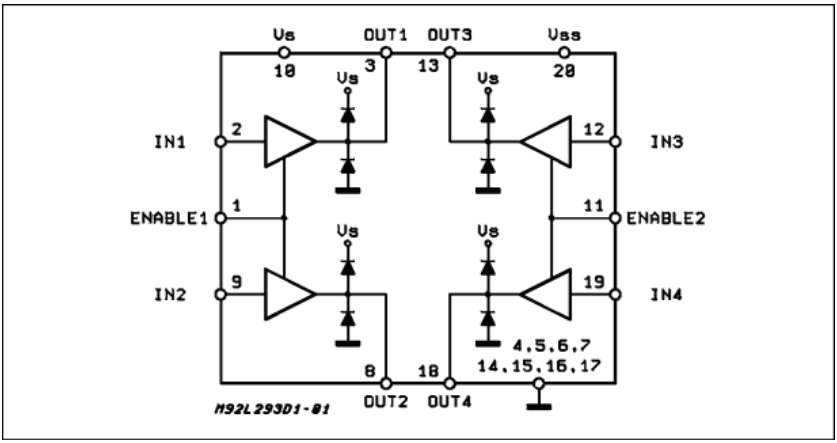

Рис. 19.1. Драйвер двигателей L293

Подключим к Arduino шаговый двигатель и с помощью кнопок будем задавать перемещение шагового двигателя в разные стороны. Схема соединений для данного эксперимента показана на рис. 19.2.

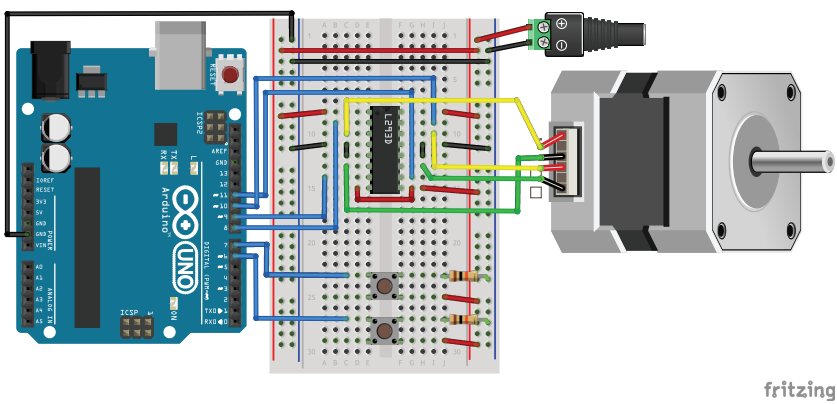

Рис. 19.2. Схема соединений шагового двигателя и Arduino

Напишем скетч управления поворотом шагового двигателя с помощью кнопок. При нажатии на первую кнопку шаговый двигатель

перемещается на 200 шагов по часовой стрелке, при нажатии на другую кнопку шаговый двигатель перемещается на 200 шагов против часовой стрелки. При написании скетча будем использовать Arduino-библиотеку Stepper. Содержимое скетча показано в листинге 19.1.

Листинг 19.1

```
#include <Stepper.h>
#define STEPS 200      // Количество шагов
Stepper stepper(STEPS, 8, 9, 10, 11);
// клавиши
int pinButtons1[]={6,7};
int lastButtons1[]={0,0};
int currentButtons1[]={0,0};
int countButtons1=2;

void setup()
   {
   stepper.setSpeed(50);
   }
void loop()
   {
   // проверка нажатия кнопок
   for(int i=0;i<countButtons1;i++)
      {
      currentButtons1[i] = debounce(lastButtons1[i],pinButtons1[i]);
      if (lastButtons1[i] == 0 && currentButtons1[i] == 1)
      // если нажатие...
         {
         if(i==0)
            stepper.step(10*STEPS);
         else
            stepper.step(-10*STEPS);
         }
      lastButtons1[i] = currentButtons1[i];
      }
   }

// Функция сглаживания дребезга
int debounce(int last,int pin1)
   {
   int current = digitalRead(pin1); // Считать состояние кнопки
   if (last != current)             // если изменилось...
      {
      delay(5);                     // ждем 5 мс
      current = digitalRead(pin1);   // считываем состояние кнопки
      return current;               // возвращаем состояние кнопки
      }
   }
```

Порядок подключения:

1. Подключаем элементы к плате Arduino по схеме на рис. 19.2.
2. Загружаем в плату Arduino скетч из листинга 19.1.
3. При нажатии на одну из кнопок шаговый двигатель делает 200 шагов в одну сторону и останавливается, при нажатии на другую кнопку двигатель делает 200 шагов в обратную сторону.

На странице http://arduino-kit.ru/0019 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 19.1.

20 Датчик температуры DS18B20

В этом эксперименте мы рассмотрим популярный цифровой датчик температуры DS18B20, работающий по протоколу 1-Wire, и создадим проект вывода показаний датчика на экран ЖКИ WH1602.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- датчик DS18B20;
- LCD-экран WH1602;
- резистор 50 Ом;
- потенциометр 1 кОм;
- провода папа-папа;
- внешний блок питания +5 В.

DS18B20 – цифровой термометр с программируемым разрешением от 9 до 12 битов, которое может сохраняться в EEPROM-памяти прибора. DS18B20 обменивается данными по шине 1-Wire и при этом может быть как единственным устройством на линии, так и работать в группе. Все процессы на шине управляются центральным микропроцессором.

Диапазон измерений датчика: от –55 °C до +125 °C с точностью 0,5 °C в диапазоне от –10 °C до +85 °C. В дополнение DS18B20 может питаться напряжением линии данных (так называемое питание от паразитного источника) при отсутствии внешнего источника напряжения.

Каждый датчик типа DS18B20 имеет уникальный 64-битный последовательный код, который позволяет общаться со множеством датчиков DS18B20, установленных на одной шине. Первые 8 битов – код серии (для DS18B20 – 28h), затем 48 битов уникального номера, и в конце 8 битов CRC-кода. Такой принцип позволяет использовать один микропроцессор, чтобы контролировать множество датчиков DS18B20, распределенных по большому участку.

В нашем эксперименте мы будем считывать данные с датчика температуры DS18B20 и выводить на экран ЖКИ WH1602, который мы рассматривали в эксперименте 16. Схема подключения датчика температуры DS18B20 и WH1602 к плате Arduino показана на рис. 20.1.

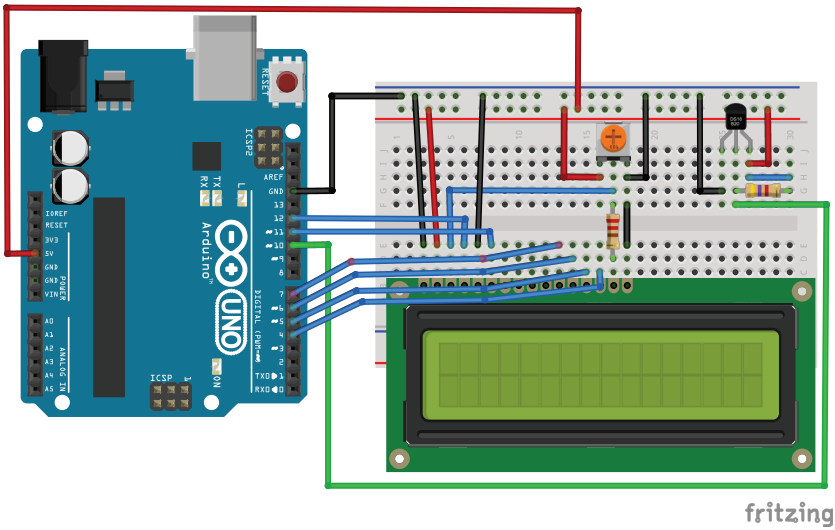

Рис. 20.1. Схема подключения датчика температуры DS18B20 и WH1602 к Arduino

Приступим к написанию скетча. Для работы с устройствами 1-Wire в Arduino есть стандартная библиотека OneWire. Содержимое скетча для чтения данных с датчика и вывода на экран индикатора WH1602 показано в листинге 20.1. Последовательность данных для чтения данных с устройств 1-Wire следующая:

1. Произвести RESET и поиск устройств на линии 1-Wire.
2. Выдать команду 0x44, чтобы запустить конвертацию температуры датчиком.
3. Подождать не менее 750 мс.
4. Выдать команду 0xBE, чтобы считать ОЗУ датчика (данные о температуре будут в первых двух байтах).

Листинг 20.1

```
#include <OneWire.h>
OneWire ds(10);       // линия 1-Wire будет на pin 10
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(12, 11, 7, 6, 5, 4);

void setup(void)
  {
  Serial.begin(9600);
```

```
   // установить размерность дисплея
   lcd.begin(16, 2);
   lcd.clear();
  }

void loop(void)
  {
   int t=get_temp();
   lcd.setCursor(0,1);lcd.print("             ");
   lcd.setCursor(0,1);
   lcd.print(t/16);lcd.print(".");lcd.print((t%16)*100/16);
}
// получение данных с датчика DS18B20
int get_temp()
   {
   byte i;
   byte present = 0;
   byte data[12];
   byte addr[8];
   int Temp;

if ( !ds.search(addr))
   {Serial.print("No more addresses.\n");
    ds.reset_search();
    return -1;
    }
// вывод в монитор уникального адреса 1-Wire устройства
lcd.setCursor(0,0);
lcd.print("R=");
for( i = 0; i < 8; i++)
   {lcd.print(addr[i], HEX);lcd.print(" ");}

if ( OneWire:crc8( addr, 7) != addr[7])
   {
   Serial.print("CRC is not valid!\n");
   return -1;
   }
if ( addr[0] != 0x28)
   {
   Serial.print("Device is not a DS18S20 family device.\n");
   return -1;
   }
ds.reset();
// запустить конвертацию температуры датчиком
ds.write(0x44,1);
```

```
delay(750);            // ждем 750 мс
present = ds.reset();
ds.select(addr);
ds.write(0xBE);     /
// считываем ОЗУ датчика
for ( i = 0; i < 9; i++)
   { data[i] = ds.read();}
Temp=(data[1]<<8)+data[0];
return Temp;
}
```

Порядок подключения:

1. Подключаем датчик DS18B20 и WH1602 по схеме на рис. 20.1.
2. Загружаем в плату Arduino скетч из листинга 20.1.
3. Смотрим на экране дисплея показания датчика температуры.

На странице http://arduino-kit.ru/0020 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 20.1.

21 Датчик влажности и температуры DHT11

В этом эксперименте мы рассмотрим датчик для измерения относительной влажности воздуха и температуры DHT11 и создадим проект вывода показаний датчика на экран ЖКИ WH1602.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- датчик DHT11;
- LCD-экран WH1602;
- резистор 50 Ом;
- потенциометр 1 кОм;
- провода папа-папа.

Датчик DHT11 (см. рис. 21.1) не отличается высоким быстродействием и точностью, однако может найти свое применение в радиолюбительских проектах из-за своей невысокой стоимости. Датчик DHT11 состоит из емкостного датчика влажности и термистора. Кроме того, датчик содержит в себе простенький АЦП для преобразования аналоговых значений влажности и температуры.

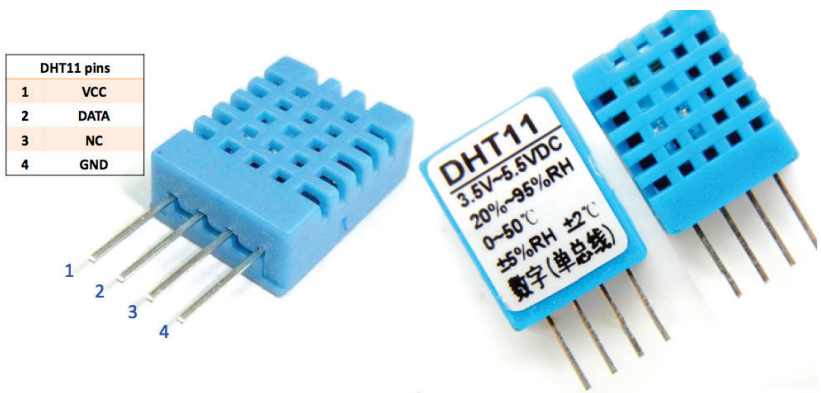

Рис. 21.1. Датчик DHT11

Датчик имеет 4 вывода в одну линию с шагом 2,54 мм:

- 1 – VCC (питание 3–5 В);
- 2 – DATA (вывод данных);
- 3 – не используется;
- 4 – GND (земля).

Протокол обмена – однопроводный, по структуре весьма похож на DS18B20, но с важными оговорками:

- DHT11 не умеет работать в «паразитном» режиме (питание по линии данных);
- каждый DS18B20 имеет персональный идентификатор, что дает возможность подключения нескольких таких датчиков к одному пину Arduino. Однако у DHT11 подобной возможности нет – один датчик будет использовать строго один цифровой пин.

В нашем эксперименте мы будем считывать данные с датчика DHT11 и выводить на экран ЖКИ WH1602, который мы рассматривали в эксперименте 16. Рекомендуемая схема подключения к Arduino содержит обязательный для однопроводных линий резистор-подтяжку к VCC, в качестве опции рекомендуется конденсатор (фильтр по питанию между VCC и GND). У нас в наличии DHT11 в виде модуля, его можно подключать к Arduino напрямую – резистор и конденсатор там уже есть. Схема подключения датчика DHT11 и WH1602 к плате Arduino показана на рис. 21.2. Для питания ЖКИ нужен отдельный блок питания +5 В.

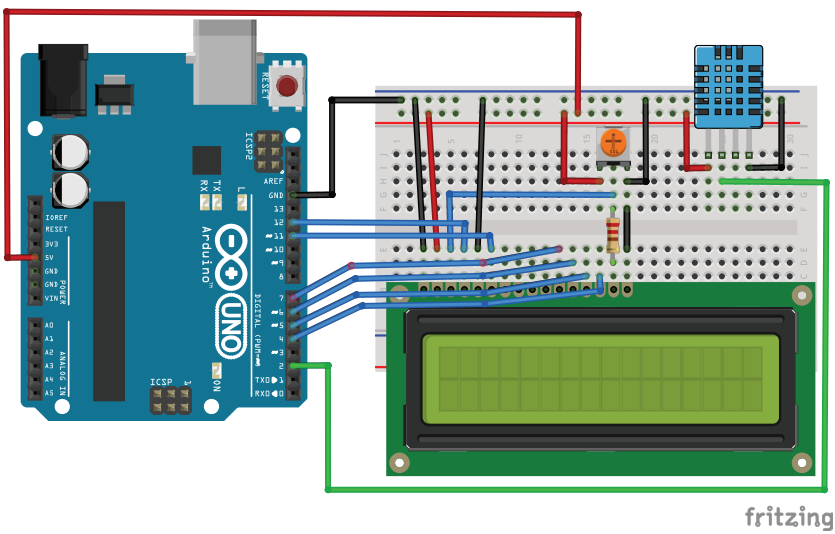

Рис. 21.2. Схема подключения датчика DHT11 и WH1602 к Arduino

Приступим к написанию скетча. Для работы с датчиками DHT11 (DHT21, DHT22) в Arduino есть библиотека OneWire. Содержимое скетча для чтения данных с датчика и вывода на экран индикатора WH1602 показано в листинге 21.1.

Листинг 21.1

```
#include "DHT.h"
#define DHTPIN 2             // пин подключения контакта DATA
#define DHTTYPE DHT11         // DHT 11
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
DHT dht(DHTPIN, DHTTYPE);

void setup()
  {
  lcd.begin(16,2);   // режим работы
  dht.begin();
  }

void loop()
  {
  // получение с датчика данных влажности и температуры
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(t) || isnan(h))      // ошибка получения данных
    {
    lcd.clear();lcd.setCursor(0,0);
    lcd.print("Failed to read");
    }
else    // вывести данные на ЖКИ
    {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Humidity: ");lcd.print(h); lcd.println(" %");
    lcd.setCursor(0,1);
    lcd.print("Temp: "); lcd.print(t);lcd.println(" *C");
    }
  delay(2000);   // пауза перед следующим измерением
}
```

Порядок подключения:

1. Подключаем датчик DHT11 и WH1602 по схеме на рис. 21.2.
2. Загружаем в плату Arduino скетч из листинга 21.1.

3. Смотрим на экране дисплея показания относительной влажности и температуры.

На странице http://arduino-kit.ru/0022 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 21.1, и библиотеку DHT.

22 Датчики газов. Принцип работы, пример работы

В этом эксперименте мы рассмотрим датчики газа для подключения к Arduino на примере датчика MQ-4 и создадим проект схемы, реагирующей на превышение концентрации газа.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- датчик газа MQ-4;
- газовая зажигалка;
- светодиод;
- резистор 220 Ом;
- провода папа-папа.

Серия MQ-сенсоров для Ардуино, построены на базе мини-нагревателя внутри и используют электрохимический сенсор. Они чувствительны для определенных диапазонов газов и используются в помещениях при комнатной температуре. Вот некоторые из них:

- MQ-3 – сенсор паров алкоголя;
- MQ-4 – сенсор для обнаружения метана, пропана;
- MQ-5 и MQ-6 – предназначены для обнаружения пропана, бутана;
- MQ-7 – чувствителен к угарному газу;
- MQ-8 – специализируется по водороду H2.

Датчики содержат аналоговый и цифровой выходы для подключения к Arduino. Советуют подключать оба выхода для более точного результата, что вовсе не обязательно.

В нашем эксперименте подключим датчик MQ4 к плате Arduino и посмотрим, как он реагирует на наличие газов. Схема подключения датчика показана на рис. 22.1.

Напишем скетч, считывающий показания с датчика MQ4 и выводящий показания в монитор последовательного порта. Если аналоговое значение с датчика превысит 750 (опасный уровень), будем зажигать светодиод, подключенный к цифровому выводу 8. Содержимое скетча показано в листинге 22.1.

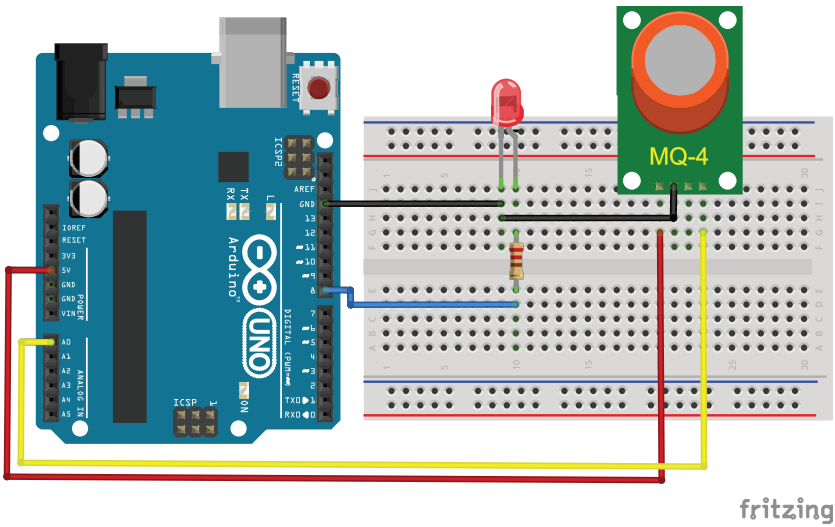

Рис. 22.1. Схема подключения датчика MQ4

Листинг 22.1

```
// контакт подключения аналогового вывода MQ4
const int analogInPin = A0;
const int ledPin = 8;   // контакт подключения светодиода
int sensorValue = 0;    // переменная для хранения значения датчика
void setup()
   {
   Serial.begin(9600);
   pinMode(ledPin, OUTPUT);
   }
void loop()
   {
   sensorValue = analogRead(analogInPin);   // получить значение
   if (sensorValue >= 750)                  // превышение уровня
      digitalWrite(ledPin, HIGH);           // зажечь светодиод превышения
   else
     digitalWrite(ledPin, LOW);             // потушить светодиод превышения
     // вывести значение в последовательный порт
     Serial.print("sensor = " );
     Serial.println(sensorValue);   // пауза перед следующим измерением
     delay(1000);
}
```

Порядок подключения:

1. Подключаем датчик MQ4 к плате Arduino по схеме на рис. 22.1.
2. Загружаем в плату Arduino скетч из листинга 22.1.
3. Открываем монитор последовательного порта Arduino IDE.
4. После подачи питания датчику необходимо время, чтобы выйти на рабочий режим, примерно 10–15 секунд. Это время нужно, чтобы нагреватель внутри датчика поднял температуру до необходимого значения.
5. Подносим газовую зажигалку к датчику и открываем газ, наблюдаем изменение показаний от датчика MQ4 в мониторе последовательного порта Arduino IDE.

На странице http://arduino-kit.ru/0022 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 22.1.

23 Ультразвуковой датчик расстояния HC-SR04. Принцип работы, подключение, пример

В этом эксперименте мы рассмотрим ультразвуковой датчик для измерения расстояния и создадим проект вывода показаний датчика на экран ЖКИ WH1602.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- ультразвуковой датчик расстояния HC-SR04;
- пьезоизлучатель;
- резистор 100 Ом;
- сервопривод;
- провода папа-папа;
- внешний блок питания +5 В.

Ультразвуковой дальномер HC-SR04 (рис. 23.1) – это помещенные на одну плату приемник и передатчик ультразвукового сигнала. Излучатель генерирует сигнал, который, отразившись от препятствия, попадает на приемник. Измерив время, за которое сигнал проходит до объекта и обратно, можно оценить расстояние. Кроме самих приемника и передатчика, на плате находится еще и необходимая обвязка, чтобы сделать работу с этим датчиком простой и удобной.

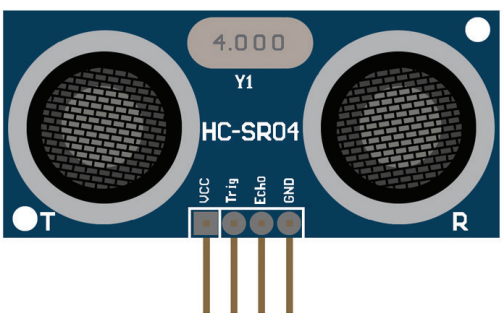

Рис. 23.1. Ультразвуковой дальномер HC-SR04

Характеристики ультразвукового дальномера HC-SR04:

- измеряемый диапазон – от 2 до 500 см;
- точность – 0,3 см;
- угол обзора – < 15°;
- напряжение питания – 5 В.

Датчик имеет 4 вывода стандарта 2,54 мм:

- VCC – питание +5 В;
- Trig (T) – вывод входного сигнала;
- Echo (R) – вывод выходного сигнала;
- GND – земля.

Последовательность действий для получения данных такова:

- подаем импульс продолжительностью 10 мкс на вывод Trig;
- внутри дальномера входной импульс преобразуется в 8 импульсов частотой 40 кГц и посылается вперед через излучатель T;
- дойдя до препятствия, посланные импульсы отражаются и принимаются приемником R, в результате получаем выходной сигнал на выводе Echo;
- непосредственно на стороне контроллера переводим полученный сигнал в расстояние по формуле:
  - ширина импульса (мкс) / 58 = дистанция (см);
  - ширина импульса (мкс) / 148 = дистанция (дюйм).

В нашем эксперименте мы создадим звуковую сигнализацию, которая будет включаться при приближении к плате Arduino на расстояние меньше 1 м. Датчик размещен на вращающемся кронштейне, прикрепленном к сервоприводу и контролирует пространство с углом обзора 180°. Если датчик обнаруживает объект в радиусе 1 м, подается звуковой сигнал на пьзоизлучатель, вращение сервопривода прекращается. Схема соединения элементов представлена на рис. 23.2.

При написании скетча будем использовать библиотеку Servo для работы с сервоприводом и библиотеку Ultrasonic.

Для работы Arduino с датчиком HC-SR04 имеется готовая библиотека – Ultrasonic. Конструктор Ultrasonic принимает два параметра: номера пинов, к которым подключены выводы Trig и Echo, соответственно:

```
Ultrasonic ultrasonic(12,13);
```

Содержимое скетча показано в листинге 23.1.

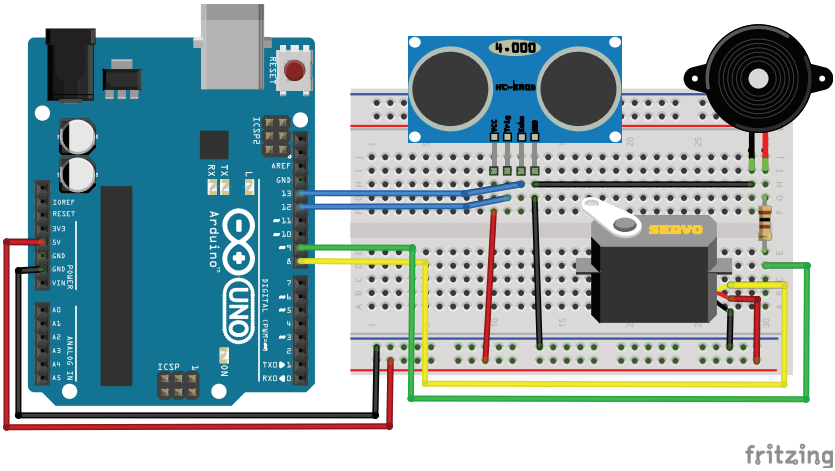

Рис. 23.2. Схема соединения элементов для звуковой сигнализации

Листинг 23.1

```
#include <Servo.h>          // подключение библиотеки Servo
Servo servo1;
const int pinServo=8;       // пин для подключения сервопривода
int pos = 0;                // переменная для хранения позиции сервопривода
int dir =1;                 // направление перемещения сервопривода
// Выводы для подключения HC-SR04 Trig - 12, Echo - 13
Ultrasonic ultrasonic(12, 13);
float dist_cm;              // переменная для дистанции, см
// подключить динамик к pin 9
int speakerPin = 9;

void setup()
   {
   // подключить переменную servo1 к выводу pinServo1
   servo1.attach(pinServo1);
   pinMode(speakerPin, OUTPUT);

   }
void loop()
   {
   servo1.write(pos);       // поворот сервоприводов на полученный угол
   delay(15);               // пауза для ожидания поворота сервоприводов
   float dist_cm = ultrasonic.Ranging(CM);
   if(dist_cm<100 && dist_cm>20)
```

```
    tone(speakerPin,);          // включить пьезозуммер
else
   {
   tone(speakerPin,0);          // отключить пьезозуммер
   pos=pos+dir;                 // изменение переменной положения сервопривода
   if(pos==0 || pos==180)
       dir=dir*(-1);            // изменение направления движения
   }
}
```

Порядок подключения:

1. Закрепляем датчик расстояния HC-SR04 на сервоприводе.
2. Подключаем датчик HC-SR04, пьезозуммер и сервопривод к плате Arduino по схеме на рис. 23.2.
3. Загружаем в плату Arduino скетч из листинга 23.1.
4. Наблюдаем за циклическим перемещением сервопривода, при попадании объекта в поле зрения датчика HC-SR04 пьезозуммер издает сигнал, сервопривод останавливается, при исчезновении объекта из поля зрения датчика сервопривод возобновляет движение.

На странице http://arduino-kit.ru/0023 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 23.1, и библиотеку Ultrasonic.

24 3-осевой гироскоп + акселерометр на примере GY-521

В этом эксперименте мы познакомимся с акселерометром и гироскопом и будем с помощью Arduino получать показания с этих датчиков.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- модуль GY-521;
- провода папа-папа.

Модуль GY-521 на микросхеме MPU6050 содержит гироскоп, акселерометр и температурный сенсор. На плате модуля GY-521 расположена необходимая обвязка MPU6050, в том числе подтягивающие резисторы, стабилизатор напряжения на 3,3 В с малым падением напряжения с фильтрующими конденсаторами. Обмен с микроконтроллером осуществляется по шине I2C.

Гироскоп представляет собой устройство, реагирующее на изменение углов ориентации контролируемого тела. Акселерометр – это устройство, которое измеряет проекцию кажущегося ускорения, то есть разницы между истинным ускорением объекта и гравитационным ускорением.

Схема соединений платы GY-521 к Arduino показана на рис. 24.1.

Код простого скетча для считывания значений гироскопа и акселерометра с датчика MPU6050 показан в листинге 24.1.

Листинг 24.1

```
// подключение библиотек
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

MPU6050 accelgyro;
int16_t ax, ay, az;
int16_t gx, gy, gz;

void setup()
   {
```

```
    Wire.begin();
    Serial.begin(38400);
    // инициализация
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();
    delay(100);
    }
void loop()
    {
    // чтение значений гироскопа и акселерометра
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    // вывод значений в монитор
    Serial.print("a/g:\t");
    Serial.print(ax); Serial.print("\t");
    Serial.print(ay); Serial.print("\t");
    Serial.print(az); Serial.print("\t");
    Serial.print(gx); Serial.print("\t");
    Serial.print(gy); Serial.print("\t");
    Serial.println(gz);
    }
```

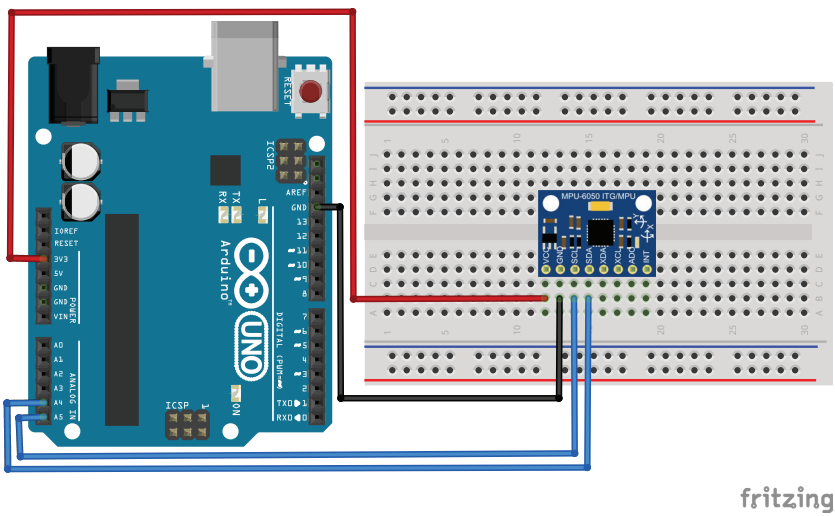

Рис. 24.1. Схема соединения GY-521 к Arduino

Порядок подключения:

1. Подключаем плату GY521 к плате Arduino по схеме на рис. 24.1.

2. Загружаем в плату Arduino скетч из листинга 24.1.
3. Открываем монитор последовательного порта Arduino IDE и смотрим вывод данных гироскопа и акселерометра (см. рис. 24.2).
4. При поворотах датчика данные изменяются.

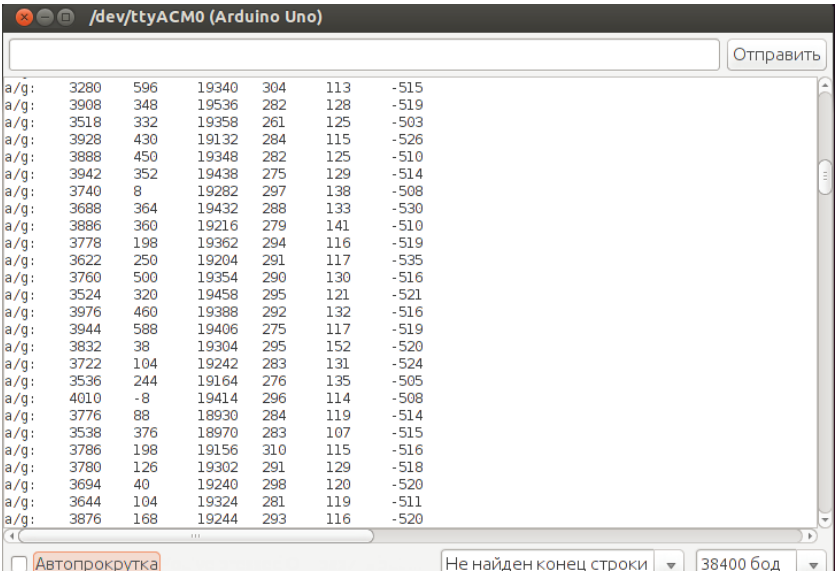

Рис. 24.2. Вывод данных гироскопа и акселерометра в монитор Arduino IDE

Область применения таких датчиков достаточно широка. Данный модуль часто применяют для стабилизации полета квадрокоптера по причине совместного использования гироскопа и акселерометра. Кроме этого, модуль можно использовать для координации различных устройств – от просто детектора движения до системы ориентации различных роботов или управления движениями какими-либо устройствами. Область подобных сенсорных устройств достаточно новая и интересная для изучения и применения в любительской технике.

На странице http://arduino-kit.ru/0024 вы можете посмотреть видеоурок данного эксперимента, скачать скетч, представленный в листинге 24.1, и библиотеки MPU6050 и I2Cdev.

25 ИК-фотоприемник и ИК-пульт. Обрабатываем команды от пульта

В этом эксперименте мы организуем беспроводную ИК-связь, которая нам позволит отправлять на плату Arduino команды с помощью любого ИК-пульта.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- ИК-приемник 36 кГц TSOP 31236;
- конденсатор 10 мкФ 25 В – 2 шт.;
- конденсатор керамический 0,1 мкФ – 2 шт.;
- резистор 100 Ом;
- светодиод – 8 шт.;
- резистор 220 Ом – 8 шт.;
- любой ИК-пульт;
- провода папа-папа.

Устройства инфракрасного (ИК) диапазона волн часто применяются в робототехнике. Наличие дешевых приемников диапазона 36–40 кГц, а также наличие большого количества пультов от бытовых приборов позволяет организовать простое и понятное беспроводное управление. В качестве приемника будем использовать микросхему TSOP31236. В одном корпусе она объединяет фотодиод, предусилитель и формирователь. На выходе формируется обычный ТТЛ-сигнал без заполнения, пригодный для дальнейшей обработки микроконтроллером. Несущая частота 36 кГц. В качестве передатчика – любой пульт для управления бытовой техникой.

Для обеспечения надежного приема и гарантированной защиты от помех при инфракрасной передаче используются модуляция сигнала и кодирование. К сожалению, нет единого и универсального протокола для ИК-пультов дистанционного управления, хотя среди всего многообразия есть наиболее распространенные. Наиболее распространенными протоколами для ИК-пультов дистанционного управления являются следующие:

- RC5;
- NEC;

- JVC;
- Sony.

Можно узнать протокол вашего пульта и написать скетч для получения кодов, отправляемых с пульта. К счастью, уже написана универсальная библиотека для приема и обработки кодов с любого пульта – Irremote. Ее мы и будем использовать при написании скетча. В нашем эксперименте мы будем с ИК-пульта зажигать светодиоды, подключенные к плате Arduino. Схема соединений показана на рис. 25.1. Выход ИК-приемника подсоединен к выводу 1 платы Arduino. Без фильтра питания будет работать нестабильно, с пропуском посылок, поэтому ставим RC-фильтр.

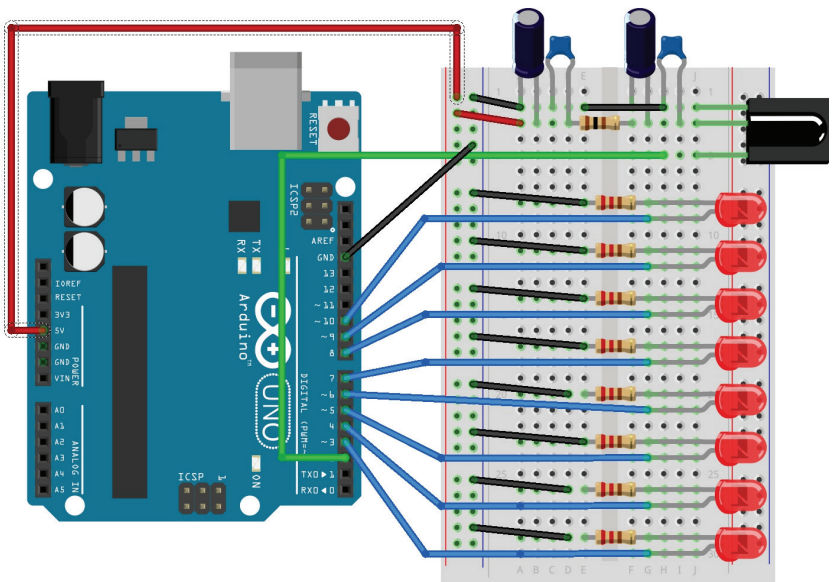

Рис. 25.1. Схема соединений ИК-управления

Сначала загрузим скетч, определяющий коды, приходящие с приемника, и выводящий их в монитор последовательного порта. Содержимое скетча показано в листинге 25.1.

Листинг 25.1

```
#include <IRremote.h>        // подключение библиотеки
int RECV_PIN = 1;            // контакт подключения ИК-приемника
IRrecv irrecv(RECV_PIN);
```

```
decode_results results;

void setup()
  {
  Serial.begin(9600);
  irrecv.enableIRIn(); // включить приемник
  }
void loop()
  {
  if (irrecv.decode(&results))
    {
    Serial.println(results.value, HEX);
    irrecv.resume(); // получить следующее значение
    }
  }
```

Порядок подключения:

1. Подключаем ИК-приемник и светодиоды к плате Arduino по схеме на рис. 25.1.
2. Загружаем в плату Arduino скетч из листинга 25.1.
3. Открываем монитор последовательного порта Arduino IDE и смотрим коды, приходящие при нажатии кнопок на ИК-пульте.

Запоминаем коды, приходящие при нажатии кнопок 2–9 на ИК-пульте. Пишем скетч, переключающий состояние светодиодов на контактах D2–D9, при получении определенного кода. Содержимое скетча показано в листинге 25.2. Значение констант K2–K9 (кодов для клавиш 2–9) у вас будет другим.

Листинг 25.2

```
// коды клавиш ИК-пульта
#define K2 1936
#define K3 3984
#define K4 144
#define K5 2192
#define K6 3472
#define K7 1424
#define K8 3216
#define K9 1168
#include <IRremote.h>       // подключение библиотеки
int RECV_PIN = 2;           // контакт подключения ИК-приемника
IRrecv irrecv(RECV_PIN);
decode_results results;
// значения на D2 - D9 Arduino
int val_pins[]={3,4,5,6,7,8,9,10};
```

```
int res=0;

void setup()
 {
 Serial.begin(9600);
 irrecv.enableIRIn(); // включить приемник
 for(int i=3;i<10;i++)
     {
      pinMode(i,OUTPUT);
      digitalWrite(i,LOW);
     }
 }
void loop()
 {
 if (irrecv.decode(&results))
  {
  switch(results.value)
       {
       case K2:   res=2;   break;
       case K3:   res=3;   break;
       case K4:   res=4;   break;
       case K5:   res=5;   break;
       case K6:   res=6;   break;
       case K7:   res=7;   break;
       case K8:   res=8;   break;
       case K9:   res=9;   break;
       default:   res=0;   break;
       }
  if(res>0)
     {
     pins[res-2]=1- pins[res-2];
     // переключить светодиод
     digitalWrite(res, pins[res-2]);
     }
  irrecv.resume(); // получить следующее значение
  }
 }
```

Загружаем скетч 25.2 на плату Arduino и нажатием на пульте кнопок 2–9 переключаем состояние светодиодов, подключенных к выводам 2–9 платы Arduino (см. рис. 25.2).

На странице http://arduino-kit.ru/0025 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 25.1 и 25.2, а также Arduino-библиотеку IRremote.

26 Часы реального времени. Принцип работы, подключение, примеры

В этом эксперименте мы рассмотрим модуль часов реального времени на микросхеме DS1307.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- модуль часов реального времени DS1307;
- батарейка CR2032 3 В;
- ЖКИ WH1602;
- потенциометр 10 кОм;
- резистор 51 Ом;
- провода папа-папа;
- внешний блок питания +5 В.

Микросхема Dallas DS1307 представляет собой часы реального времени с календарем и дополнительной памятью NW SRAM (56 байт). Микросхема подключается к микроконтроллеру при помощи шины I2C. Количество дней в месяце рассчитывается с учетом високосных лет до 2100 г. В микросхеме DS1307 имеется встроенная схема, определяющая аварийное отключение питания и автоматически подключающая резервную батарейку. При этом отсчет времени продолжается, и после восстановления питания часы показывают правильное время. Также в этой микросхеме имеется программируемый генератор прямоугольных импульсов, позволяющий вырабатывать одну из четырех частот (1 Гц, 4096 Гц, 8192 Гц или 32768 Гц). Часы подключаются по протоколу I2C всего двумя проводами. SCL и SDA – это выводы интерфейса I2C. Необходимо дополнительно подтянуть выводы, к которым подключаются часы к шине питания, с помощью резисторов 2 кОм. SCL и SDA на разных платах расположены на разных выводах:

- Uno, Nano – A4 (SDA), A5 (SCL);
- Mega2560 – 20 (SDA), 21 (SCL);
- Leonardo – 2 (SDA), 3 (SCL).

Вывод SDA часов подключается к выводу SDA контроллера. SDL часов – соответственно, к SDL контроллера. В нашем эксперименте мы будем выводить дату и время, получаемые с микросхемы DS1307, на экран LCD индикатора WH1602. Схема подключения показана на рис. 26.1.

При написании скетча используем библиотеку Time, которая является «оберткой» для библиотеки DS1307, и библиотеку Wire для работы с I2C-устройствами. Для работы с ЖКИ используем библиотеку LiquidCrystal. Содержимое скетча показано в листинге 26.1.

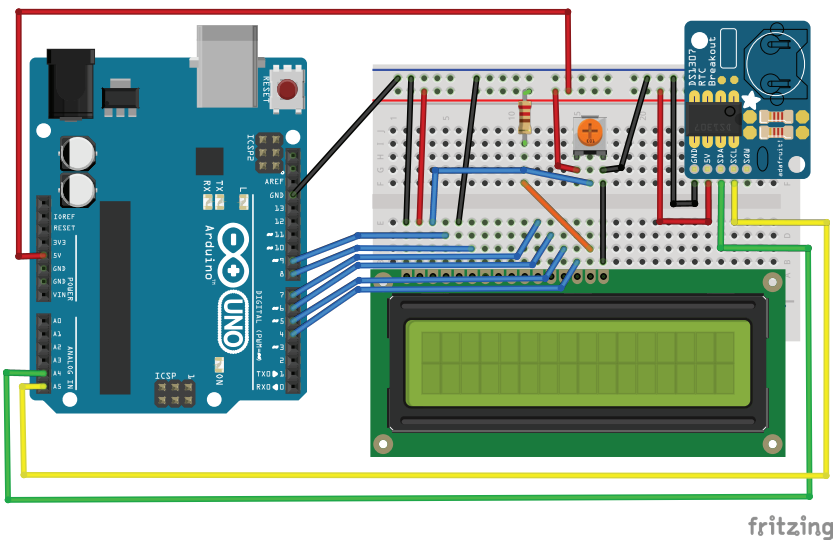

Рис. 26.1. Схема подключения модуля DS1307 и WH1602 к Arduino

Листинг 26.1

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для lcd
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);

void setup()
  {
```

```
   lcd.begin(16, 2);      // установить размерность дисплея
   }
void loop()
   {
   tmElements_t tm;
   if (RTC.read(tm))      // получение времени
      {
      print2digits(tm.Hour,0,0);
      lcd.print(":");
      print2digits(tm.Minute,3,0);
      lcd.print(":");
      print2digits(tm.Second,6,0);
      print2digits(tm.Day,0,1);
      lcd.print("/");
      print2digits(tm.Month,3,1);
      lcd.print("/");
      lcd.print(tmYearToCalendar(tm.Year));
      }
   else
      {
      if (RTC.chipPresent())
         {
         lcd.clear();
         lcd.setCursor(0, 0);
         lcd.print("DS1307 is stopped");
         }
      else
         {
         lcd.clear();
         lcd.setCursor(0, 0);
         lcd.print("DS1307 read error");
         }
      delay(9000);
      }
   delay(1000);
   }
// процедура вывода на дисплей с добавлением до двух цифр
void print2digits(int number,int col, int str)
   {
   lcd.setCursor(col, str);
   if (number >= 0 && number < 10)
      {lcd.print("0");}
      lcd.print(number);
   }
```

Порядок подключения:

1. Подключаем модуль DS1307 и ЖКИ к плате Arduino по схеме на рис. 26.1.
2. Загружаем в плату Arduino скетч из листинга 26.1.
3. Смотрим на экране ЖКИ меняющееся ежесекундно время и дату.

Однако на экране дисплея мы видим неверное время и неверную дату. Дело в том, что при отсутствии питания значение времени в микросхеме DS1307 сбрасывается на 00:00:00 01/01/2000. Чтобы при отключении питания время не сбрасывалось, предусмотрено аварийное питание модуля от батарейки 3 В.

Для установки времени в библиотеке есть функция `RTC.write (tmElements_t tm)`. Добавим в скетч возможность установки данных RTC по последовательному порту отправкой строки вида «dd/mm/YYYY hh:mm:ss». Содержимое скетча показано в листинге 26.2.

Листинг 26.2

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для lcd
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
// строка, собираемая из данных, приходящих в последовательный порт
String inputString = "";
boolean stringComplete = false;  // флаг комплектности строки


void setup()
   {
   Serial.begin(9600);         // запустить последовательный порт
   lcd.begin(16, 2);           // установить размерность дисплея
   }

void loop()
   {
   tmElements_t tm;


   // ожидание конца строки для анализа поступившего запроса:
   if (stringComplete)
```

```
    {
    tm.Day=(int(inputString[0])-48)*10+(int(inputString[1])-48);
    tm.Month=(int(inputString[3])-48)*10+(int(inputString[4])-48);
    tm.Year=CalendarYrToTm((int(inputString[6])-
                48)*1000+(int(inputString[7])-48)*100+
                (int(inputString[8])-48)*10+(int(inputString[9])-48));

    tm.Hour=(int(inputString[11])-48)*10+(int(inputString[12])-48);
    tm.Minute=(int(inputString[14])-48)*10+(int(inputString[15])-48);
    tm.Second=(int(inputString[17])-48)*10+(int(inputString[18])-48);
    RTC.write(tm);   // записать время в RTC
    // очистить строку
    inputString = "";
    stringComplete = false;
    }
  if (RTC.read(tm))
    {
    print2digits(tm.Hour,0,0);
    lcd.print(":");
    print2digits(tm.Minute,3,0);
    lcd.print(":");
    print2digits(tm.Second,6,0);
    print2digits(tm.Day,0,1);
    lcd.print("/");
    print2digits(tm.Month,3,1);
    lcd.print("/");
    lcd.print(tmYearToCalendar(tm.Year));
    }
  else
    {
    if (RTC.chipPresent())
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 is stopped");
      }
    else
      {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("DS1307 read error");
      }
    delay(9000);
    }
```

```
   delay(1000);
   }
// процедура вывода на дисплей с добавлением до двух цифр
void print2digits(int number,int col, int str)
   {
   lcd.setCursor(col, str);
   if (number >= 0 && number < 10)
      {lcd.print("0");}
   lcd.print(number);
   }
// получение данных по последовательному порту
void serialEvent()
   {
   while (Serial.available())
      { // получить очередной байт:
      char inChar = (char)Serial.read();
      // добавить в строку
      inputString += inChar;
      // /n - конец передачи
      if (inChar == '\n')
         {stringComplete = true;}
      }
   }
```

Теперь установим время из монитора последовательного порта отправкой строки «dd/mm/YYYY hh:mm:ss» и увидим на экране дисплея отображение верной даты и времени.

На странице http://arduino-kit.ru/0026 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 26.1 и 26.2, а также Arduino-библиотеки Time и DS1307.

27 SD-карта. Чтение и запись данных

В этом эксперименте мы покажем, как к плате Arduino подключить SD-карту.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- модуль SD-card;
- модуль часов реального времени DS1307 с батарейкой;
- датчик температуры LM335;
- резистор 2,2 кОм;
- провода папа-папа.

Если вашим Arduino-проектам не хватает памяти, а объем энергонезависимой памяти EEPROM в платах Arduino совсем небольшой, можно использовать внешние носители. Один из самых простых по подключению к платам Arduino – это SD-карта. Можно подсоединиться к SD-карте напрямую, а можно использовать модули. Подсоединим модуль SD-card к плате Arduino и напишем пример сохранения на SD-карте данных аналогового датчика температуры LM335. Для анализа данных температуры потребуется знать и время измерения данных, поэтому будем использовать и модуль часов реального времени RTC. Соберем схему, показанную на рис. 27.1.

При написании скетча используем библиотеку SD для работы с SD-картой, а также библиотеки Time и DS1307 для работы с модулем RTC. Содержимое скетча показано в листинге 27.1. Каждые 5 минут мы считываем данные с датчика LM335, подключенного к аналоговому входу A0, и заносим время измерения и данные температуры в файл вида d-m-Y. В начале суток создаем новый файл на новый день.

Листинг 27.1

```
// подключение библиотек для RTC
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
// подключение библиотеки для SD
```

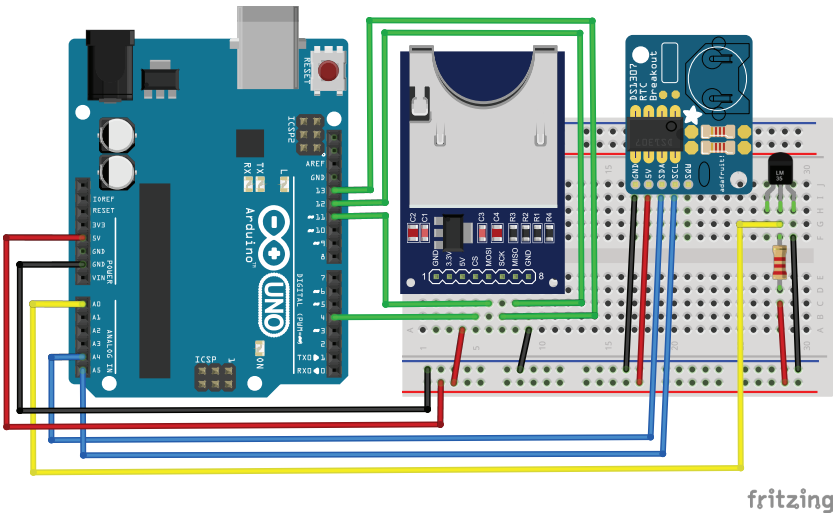

Рис. 27.1. Схема подключения модулей SD-card и DS1307 к Arduino

```
#include <SD.h>

File myFile;
String sfilename;
char filename[20];
const int LM335=A0;     // для подключения LM335
tmElements_t tm;
unsigned long millis1=0;

void setup()
  {;}
void loop()
  {
// проверка прошло 5 минут?
 if(millis()-millis1>5*60*000)
     {
     millis1=millis();
     // получить имя файла для текущего дня
     sfilename=get_file_name();
     sfilename.toCharArray(filename,20);
     // открыть файл или создать
     myFile = SD.open(filename, FILE_WRITE);
     // получить температуру
```

```
      double val = analogRead(lm335);          // чтение
      double voltage = val*5.0/1024;           // перевод в вольты
      double temp = voltage*100 - 273.15;      // в градусы Цельсия
      // получить время H:m
      // создать запись для файла
      record=get_time();
      record+=" ";
      record+=String(temp);
      myFile.println(record);
      myFile.close();
      }
// получение времени дня
String get_time()
  {
   String time1;
   RTC.read(tm);
   if(tm.Hour()<10)
    time1="0"+String(tm.Hour(),DEC);
   else
    time1=String(tm.Hour(),DEC);
   if(tm.Minute()<10)
    time1+=":0"+String(tm.Minute(),DEC);
   else
    time1+=":"+String(tm.Minute(),DEC);
   return time1;
  }
// получение имени файла для текущего дня
String get_file_name()
  {
   String filename1;
   RTC.read(datetime);
   filename1+=String(tmYearToCalendar(datetime.Year),DEC);
   if(datetime.Month<10)
    filename1+="0"+String(datetime.Month,DEC);
   else
    filename1+=""+String(datetime.Month,DEC);
   if(datetime.Day<10)
    filename1+="0"+String(datetime.Day,DEC);
   else
    filename1+=""+String(datetime.Day,DEC);
   filename1+=".txt";
   return filename1;
  }
```

Порядок подключения:

1. Подключаем модули SD card, DS1307 и датчик температуры LM335 к плате Arduino по схеме на рис. 27.1.
2. Загружаем в плату Arduino скетч из листинга 27.1.

Ждем продолжительное время, затем вынимаем карту, вставляем в компьютер, открываем файл, соответствующий текущему дню, и смотрим его содержимое – по строкам времени замера и показания температуры.

На странице http://arduino-kit.ru/0027 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 27.1.

28 Считыватель RFID на примере RC522. Принцип работы, подключение, примеры

В этом эксперименте мы покажем, как плата Arduino получает доступ к данным RFID-карт и брелоков Mifare с помощью RFID-считывателя RC522C.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- RFID-считыватель RC522;
- брелок;
- карта;
- провода папа-папа.

Радиочастотная идентификация (RFID) – это технология автоматической бесконтактной идентификации объектов при помощи радиочастотного канала связи. Базовая система RFID состоит из:

- радиочастотной метки;
- считывателя информации (ридера);
- компьютера для обработки информации.

Идентификация объектов производится по уникальному цифровому коду, который считывается из памяти электронной метки, прикрепляемой к объекту идентификации. Считыватель содержит в своем составе передатчик и антенну, посредством которых излучается электромагнитное поле определенной частоты. Попавшие в зону действия считывающего поля радиочастотные метки «отвечают» собственным сигналом, содержащим информацию (идентификационный номер товара, пользовательские данные и т. д.). Сигнал улавливается антенной считывателя, информация расшифровывается и передается в компьютер для обработки. Подавляющее большинство современных систем контроля доступа (СКД) использует в качестве средств доступа идентификаторы, работающие на частоте 125 кГц. Это проксимити-карты доступа (только чтение), самыми распространенными являются карты EM-Marin, а также HID, Indala. Карты этого стандарта являются удобным средством открывания дверей и турникетов. Но не более. Эти карты не обладают никакой защищенностью,

легко копируются и подделываются и, соответственно, ничего не дают для защиты объекта от несанкционированного проникновения. Настоящую защиту от копирования и подделки обеспечивают такие идентификаторы, в чипах которых реализована криптографическая защита. Это бесконтактные смарт-карты, работающие на частоте 13,56 МГц, наиболее распространенными из них являются карты Mifare®. В картах этих стандартов криптозащита организована на высоком уровне, и подделка таких карт практически невозможна.

Модуль RC522 – RFID-модуль 13,56 МГц с SPI-интерфейсом. В комплекте к модулю идут 2 RFID-метки – в виде карты и брелока.

Основные характеристики:

- основан на микросхеме MFRC522;
- напряжение питания: 3,3 В;
- потребляемый ток: 13–26 мА;
- рабочая частота: 13,56 МГц;
- дальность считывания: 0~60 мм;
- интерфейс: SPI, максимальная скорость передачи 10 МБит/с;
- размер: 40×60 мм;
- чтение и запись RFID-меток.

Схема подключения модуля к плате Arduino показана на рис. 28.1.

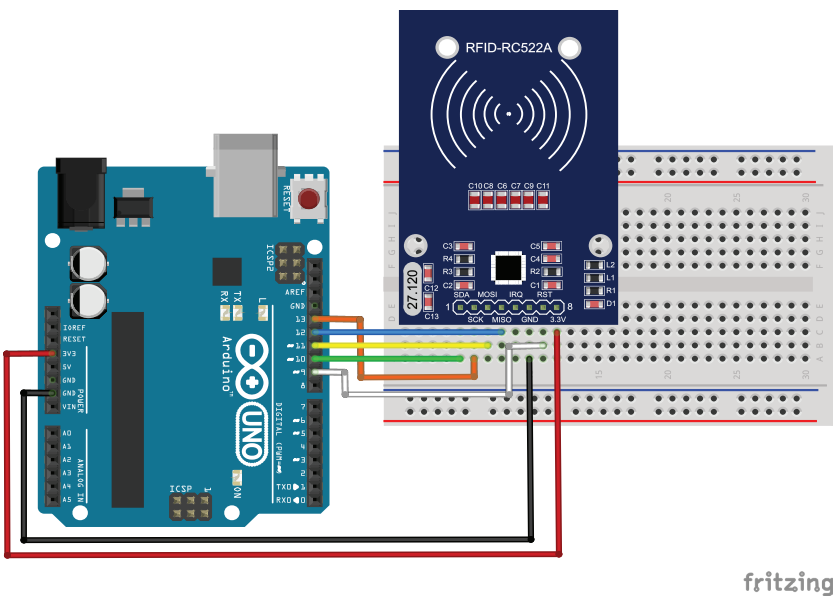

Рис. 28.1. Схема подключения модуля RFID-считывателя RC522C к Arduino

Напишем скетч считывания с карты и вывода в последовательный порт Arduino UID (уникальный идентификационный номер) RFID-метки (карты или брелока). При написании скетча будем использовать библиотеку MFRC522 (https://github.com/miguelbalboa/rfid). Содержимое скетча показано в листинге 28.1.

Листинг 28.1

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

void setup()
   {
   Serial.begin(9600);    // инициализация последовательного порта
   SPI.begin();           // инициализация SPI
   mfrc522.PCD_Init();    // инициализация MFRC522
   }

void loop()
   {
   if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
   // чтение карты
   if ( ! mfrc522.PICC_ReadCardSerial())
      return;
   // показать результат чтения UID и тип метки
   Serial.print(F("Card UID:"));
   dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
   Serial.println();
   Serial.print(F("PICC type: "));
   byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
   Serial.println(mfrc522.PICC_GetTypeName(piccType));
   delay(2000);
   }
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize)
   {
   for (byte i = 0; i < bufferSize; i++)
```

```
  {
  Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
    }
}
```

Порядок подключения:

1. Подключаем модули RFID-считывателя RC522 к плате Arduino по схеме на рис. 28.1.
2. Загружаем в плату Arduino скетч из листинга 28.1. Открываем монитор последовательного порта.
3. Подносим метку (карту или брелок) к считывателю и видим вывод в последовательный порт данных метки UID и тип (рис. 28.2).

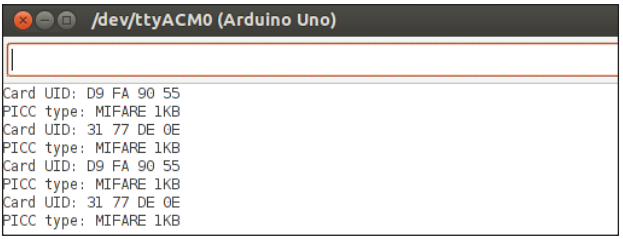

Рис. 28.2. Вывод в последовательный порт информации о метках

Метки Mirafe позволяют записывать на них информацию. В следующем скетче мы организуем на карте счетчик, который будет инкрементироваться при поднесении карты к считывателю. В последовательный порт будем выводить показания счетчика. Содержимое скетча показано в листинге 28.2.

Листинг 28.2

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
MFRC522:MIFARE_Key key;
byte sector = 1;
byte blockAddr = 4;
```

```
byte dataBlock[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
byte trailerBlock = 7;
byte status;
byte buffer[18];
byte size = sizeof(buffer);

void setup()
   {
   Serial.begin(9600);    // инициализация последовательного порта
   SPI.begin();           // инициализация SPI
   mfrc522.PCD_Init();    // инициализация MFRC522
   // Значение ключа (A или B) - FFFFFFFFFFFFh значение с завода
   for (byte i = 0; i < 6; i++)
      key.keyByte[i] = 0xFF;
   }
void loop()
   {
   if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
   // чтение карты
   if ( ! mfrc522.PICC_ReadCardSerial())
      return;
   // показать результат чтения UID и тип метки
   Serial.print(F("Card UID:"));
   dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
   Serial.println();
   Serial.print(F("PICC type: "));
   byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
   Serial.println(mfrc522.PICC_GetTypeName(piccType));


   // Чтение данных из блока 4
   Serial.print(F("Reading data from block "));
   Serial.print(blockAddr);
   Serial.println(F(" ..."));
   Serial.print(F("Data for count ")); Serial.print(blockAddr);
   Serial.println(F(":"));
   dump_byte_array(buffer, 2); Serial.println();
   Serial.println();
   for (byte i = 0; i < 16; i++) // запись в buffer[]
      dataBlock[i]=buffer[i];
   // получение байт счетчика (0 и 1)
   int count1=(buffer[0]<<8)+buffer[1];
   Serial.print("count1=");Serial.println(count1);
   count1=count1+1;   // инкремент счетчика
   dataBlock[0]=highByte(count1);
   dataBlock[1]=lowByte(count1);
```

```
   // Аутентификация key B
   Serial.println(F("Authenticating again using key B..."));
   // Запись данных в блок
   Serial.print(F("Writing data into block "));
   Serial.print(blockAddr);
   Serial.println(F(" ..."));
   dump_byte_array(dataBlock, 2); Serial.println();
   }
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize)
   {
   for (byte i = 0; i < bufferSize; i++)
    {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
     Serial.print(buffer[i], HEX);
     }
   }
```

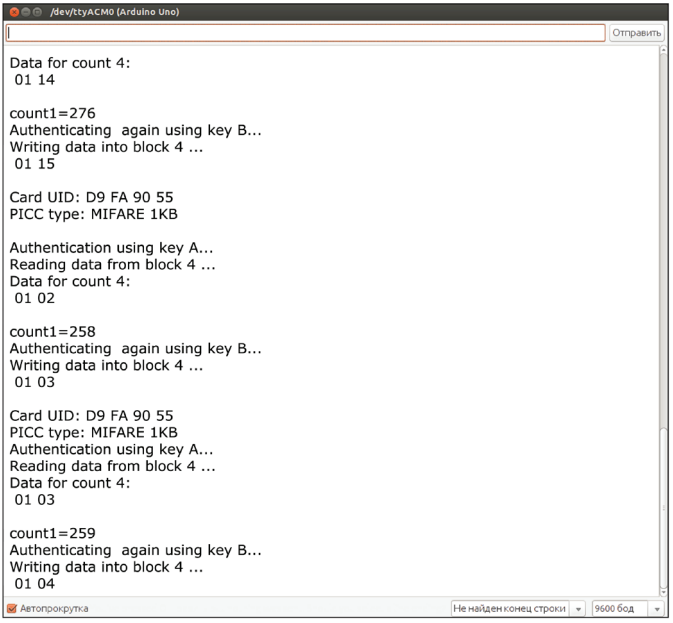

Рис. 28.3. Вывод в последовательный порт информации о счетчике на метках

На странице http://arduino-kit.ru/0028 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 28.1, 28.2, а также Arduino-библиотеку MFRC522.

29 Работа с Интернетом на примере Arduino Ethernet shield W5100

В этом эксперименте мы покажем, как нашей плате Arduino получить доступ к сети Интернет с помощью модуля Ethernet shield W5100.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- модуль Ethernet shield W5100;
- светодиод – 2 шт.;
- резистор 220 Ом – 2 шт.;
- провода папа-папа.

Ethernet Shield позволяет легко подключить вашу плату Arduino к локальной сети или сети Интернет. Он предоставляет возможность Arduino отправлять и принимать данные из любой точки мира с помощью интернет-соединения. Например, можно реализовать удаленное управление вашими исполнительными устройствами, подключенными к реле, через веб-сайт или создать устройство, которое с помощью звукового сигнала оповестит вас о новом электронном письме.

Рассмотрим использование платы Arduino с подключенным Ethernet Shield в качестве сервера, выдающего клиенту (браузеру) веб-страницу и позволяющего запросами с браузера изменять состояния подключенного к Arduino светодиода. Подключаем к плате Arduino Ethernet shield, а к выводам D7, D8 – светодиоды через резисторы 220 Ом (см. рис. 29.1).

При написании скетча используем встроенную в Arduino IDE библиотеку Ethernet. Содержимое скетча показано в листинге 29.1.

Листинг 29.1

```
#include <SPI.h>
#include <Ethernet.h>
// mac-адрес платы и ip-адрес сервера
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,0,214);

// Initialize the Ethernet server library
// with the IP address and port you want to use
```

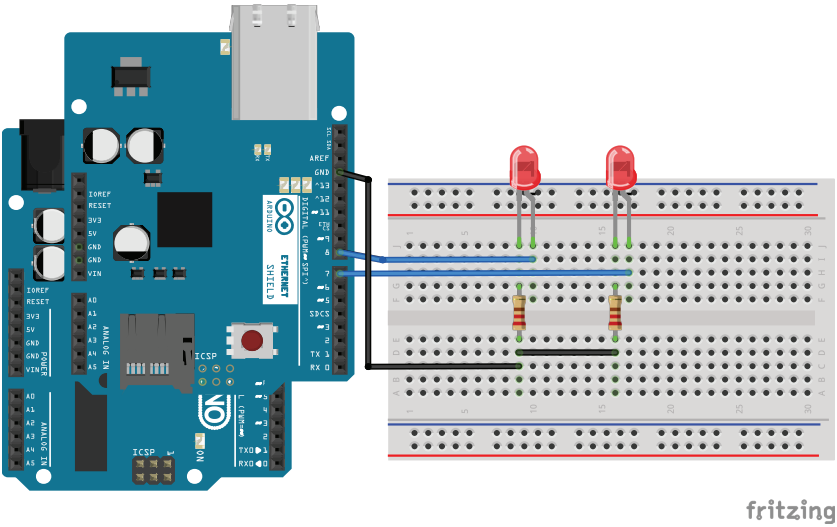

Рис. 29.1. Схема подключения модуля Ethernet shield и светодиодов

```
// (port 80 is default for HTTP):
EthernetServer server(80);
int pins[] = { 7, 8};          // Пины для светодиодов
int pinState[] = {0, 0};       // Состояние пинов
String getData="";
boolean startGet=false;
void setup()
   {
   Serial.begin(9600);
   for(int i=0;i<2;i++)
      {
      pinMode(pins[i],OUTPUT);    // контакты подключения светодиодов
      digitalWrite(i,LOW);        // выключить светодиоды
     }
   // инициализации библиотеки Ethernet server
   Ethernet.begin(mac, ip);
   server.begin();
   }

void loop()
   {
   // ожидание подключения клиентов
   EthernetClient client = server.available();
```

```
if (client)
   {
   boolean currentLineIsBlank = true;
   while (client.connected())
      {
      if (client.available())
         {
         char c = client.read();
         if(startGet==true)    // данные после '?'
             getData+=c;
         if(c == '?')          // начало сбора данных после '?'
            startGet=true;
         if (c == '\n' && currentLineIsBlank) // окончание получения
            {
            if(getData.length()<1)   // запрос без get-данных
               {
               pinState[0]=0;
               pinState[1]=0;
               }
            else
               {
               pinState[0]=int(getData[5])-48;
               pinState[1]=int(getData[12])-48;
              }
         // отправка заголовков клиенту
         client.println("HTTP/1.1 200 OK");
         client.println("Content-Type: text/html");
         client.println("Connection: close");
         client.println();
         // формирование страницы ответа
         client.println("<!DOCTYPE HTML>");
         client.println("<html>");
         client.println("<h3>Ethernet shield + LEDS</h3>");
         client.println("<form method='get'>");
         // светодиод 1
         client.print("<div>");
         client.print("led1 off<input type='radio' name='led1' value=0
             onclick='document.getElementById(\"submit\").click();' ");
         if (pinState[0] == 0)
             client.print("checked");
         client.println(">");
         client.print("<input type='radio' name='led1' value=1
             onclick='document.getElementById(\"submit\").click();' ");
```

```
            if (pinState[0] == 1)
               client.print("checked");
            client.println("> on");
            client.println("</div>");
            // светодиод 2
            client.print("<div>");
            client.print("led2 off<input type='radio' name='led2' value=0
               onclick='document.getElementById(\"submit\").click();' ");
            if (pinState[1] == 0)
            client.print("checked");
            client.println(">");
            client.print("<input type='radio' name='led2' value=1
               onclick='document.getElementById(\"submit\").click();' ");
            if (pinState[1] == 1)
               client.print("checked");
            client.println("> on");
            client.println("</div>");
            client.println("<input type='submit' id='submit'
                style='visibility:hidden;' value='Refresh'>");
            client.println("</form>");
            client.println("</html>");
            break;
            }
            if (c == '\n')
               {currentLineIsBlank = true;}
            else if (c != '\r')
               {currentLineIsBlank = false;}
            }
         }
      }
      // задержка для получения клиентом данных
      delay(1);
      // закрыть соединение
      client.stop();
      for(int i=0;i<2;i++)     // светодиоды включить или выключить
         {digitalWrite(pins[i],pinState[i]);}
      startGet=false;
      getData="";
}
```

Порядок подключения:

1. Подключаем Ethernet shield к плате Arduino, с помощью кабеля RJ45 подключаем Ethernet shield к сети.
2. Подключаем светодиоды по схеме на рис. 29.1.

3. Загружаем в плату Arduino скетч из листинга 29.1.
4. Открываем браузер на любом компьютере данной сети и в адресной строке набираем http://192.168.0.214 (тот адрес, который вы присваиваете Arduino в скетче).
5. На странице (рис. 29.2), изменяя статус элементов input radio, видим изменение состояния светодиодов, подключенных к плате Arduino.

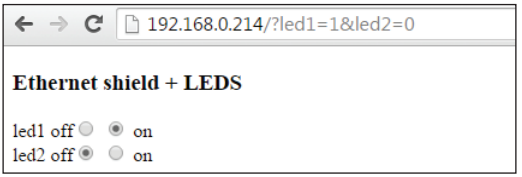

Рис. 29.2. Веб-страница, формируемая Arduino-сервером

На странице http://arduino-kit.ru/0029 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 29.1.

30 Беспроводная связь. Модуль Wi-Fi ESP8266

В этом эксперименте мы познакомимся с модулем ESP8266, с помощью которого можно подключить плату Arduini к сетям Wi-Fi, и напишем скетч для передачи данных датчика температуры на вебсервис Народный мониторинг.

Необходимые компоненты:

- контроллер Arduino UNO;
- модуль ESP8266 ESP-01;
- плата для прототипирования;
- датчик температуры LM335;
- резистор 2,2 кОм;
- провода папа-папа;
- блок питания +5 В 1 А;
- преобразователь напряжения 3–30 В.

После своего появления платы на базе Wifi чипа ESP8266 стали по-настоящему народными. Огромные возможности и минимальная цена сделали свое дело. Платы на ESP8266 – это не просто модули для связи по Wi-Fi. Чип, по сути, является микроконтроллером со своими интерфейсами SPI, UART, а также портами GPIO, а это значит, что модуль можно использовать автономно без Arduino и других плат с микроконтроллерами. Существует около 11 официальных модификаций платы. В нашем распоряжении самая простая плата – ESP01. Распиновка платы показана на рис. 30.1. Покажем, как использовать ее в качестве Wi-Fi модуля для Arduino.

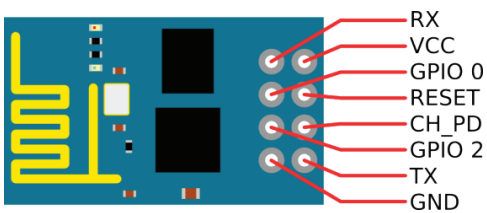

Рис. 30.1. Распиновка модуля ESP-01

Модуль ESP8266 рассчитан только на 3,3 В. Поэтому нам необходим источник питания 3,3 В. Схема подключения модуля ESP-01

к плате Arduino показана на рис. 30.2. Общение с модулем с помощью AT-команд. Список основных AT-команд показан в табл. 30.1. Загрузим на плату Arduino скетч, показанный в листинге 30.1, и будем отправлять в модуль ESP-01 AT-команды. Результат выполнения команд показан на рис. 30.3.

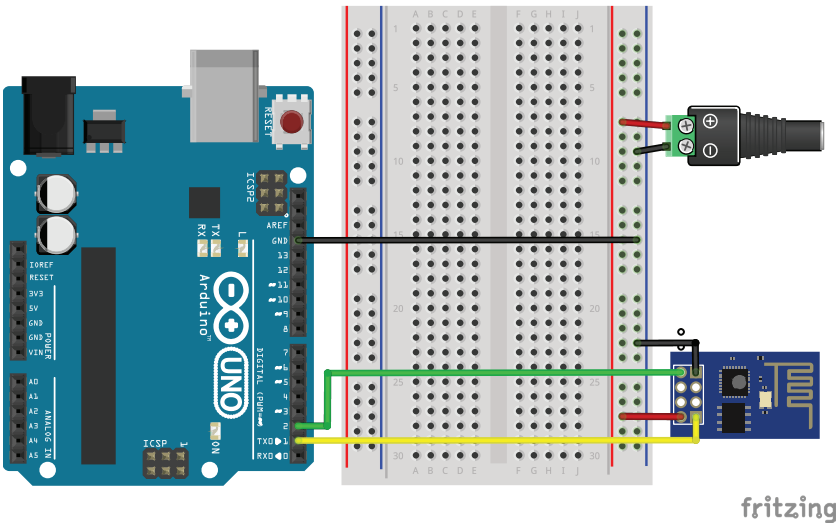

Рис. 30.2. Схема подключения модуля ESP-01 к Arduino

Листинг 30.1

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);  // указываем пины rx и tx

void setup()
   {
   pinMode(2,INPUT);
   pinMode(3,OUTPUT);
   Serial.begin(9600);
   mySerial.begin(9600);
   }

void loop()
   {
   if (mySerial.available())
      {
      int c = mySerial.read();  // читаем из software-порта
      Serial.write(c);          // пишем в hardware-порт
```

```
    }
  if (Serial.available())
    {
    int c = Serial.read();    // читаем из hardware-порта
    mySerial.write(c);        // пишем в software-порт
    }
  }
```

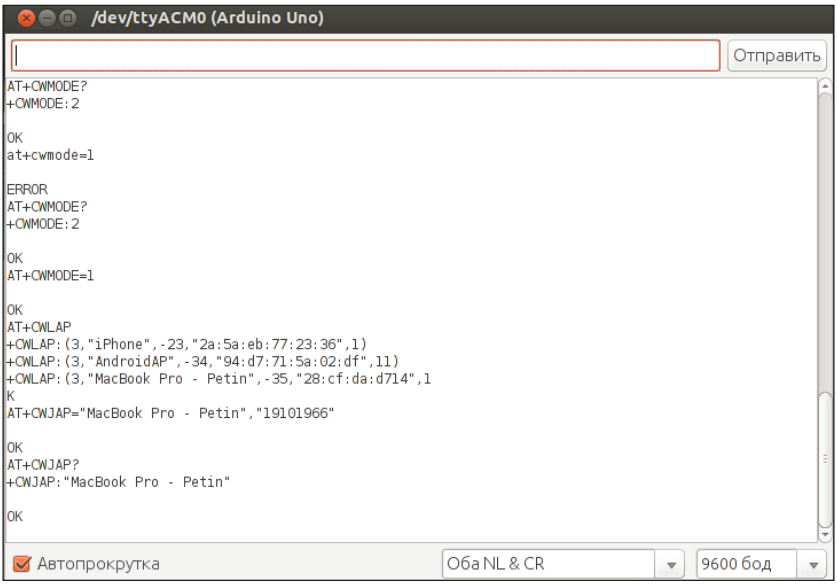

Рис. 30.3. Отправка AT-команд из Arduino IDE

Создадим скрипт отправки данных на сайт Народный мониторинг (http://narodmon.ru). Подключим к плате Arduino датчик температуры LM335 и каждые 10 минут будем отправлять данные. Схема соединений показана на рис. 30.4. Для отправки данных необходимо выполнить следующую последовательность действий:

1. Сброс ESP-01 и проверка готовности модуля (AT+RST).
2. Подключение к сети по Wi-Fi (`AT+CWJAP="<SSID>","<password>"`).
3. Выбор режима одиночного соединения (AT+CIPMUX=0).
4. Создание TCP-соединения (`AT_CIPSTART="TCP","92.39.235.156",8283`).
5. Отправка данных (`AT+CIPSEND=<length>` и сами данные `#<MAC>\n#<ID_sensor>\n#<value>\n##`).
6. Закрыть TCP-соединение (`AT+CIPCLOSE`).
7. Пауза 10 минут и переход к шагу 4.

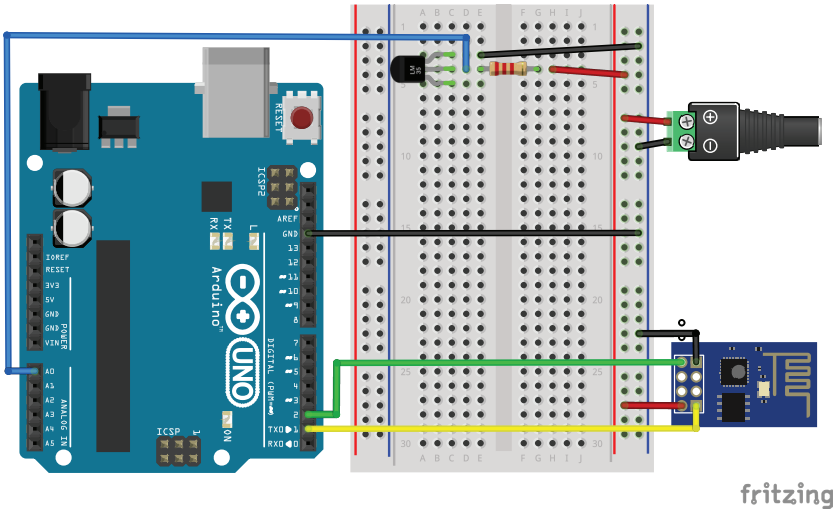

Рис. 30.4. Схема подключения модуля ESP-01 к Arduino

Таблица 30.1

Команда	Описание	Выполнение
AT	Проверка модуля. Если модуль успешно стартовал, то отвечает OK	AT
AT+RST	Перезапуск модуля	AT+RST
AT+RESTORE	Сбросить на заводские настройки	AT+RESTORE
AT+UART	Настройка последовательного интерфейса	AT+UART=baudrate,databits, stopbits,parity,flow control
AT+CWMODE	Переключение режима Wi-Fi. Для вступления в силу требуется перезапуск модуля командой AT+RST	AT+CWMODE? AT+CWMODE=1 (station) AT+CWMODE=2 (AP) AT+CWMODE=3 (station+AP)
AT+CWJAP	Подключение к AP (точке доступа)	AT+CWJAP=<идентификатор сети>, <пароль> AT+CWJAP?
AT+CWLAP	Отобразить список доступных AP	AT+CWLAP
AT+CWQAP	Отключение от AP	AT+CWQAP
AT+CIPSTART	Установить подключение TCP или UDP	AT+CIPSTART="<TCP/UDP>","<IP>", port
AT+CIPSEND	Отправить данные	AT+CIPSEND=? AT+CIPSEND=<длина> AT+CIPSEND=<идентификатор><длина>
AT+CIPCLOSE	Закрыть подключение TCP или UDP	AT+CIPCLOSE

Содержимое скетча показано в листинге 30.2.

Листинг 30.2

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);      // RX, TX
const int LM335=A0;   // контакт подключения датчика LM335

#define SSID "MacBook Pro - Petin"  // введите ваш SSID
#define PASS "19101966"             // введите ваш пароль
#define DST_IP "92.39.235.156"      // naronmon.ru

void setup()
   {
   Serial.begin(9600);              // для отладки
   mySerial.begin(9600);
   delay(2000);
   Serial.println("Init");
   mySerial.println("AT+RST");      // сброс и проверка, если модуль готов
   delay(1000);
   if(mySerial.find("ready"))
      {Serial.println("WiFi - Module is ready");}
   else
      {Serial.println("Module dosn't respond.");
       while(1);
       }
   delay(1000);
   // соединение по wifi
   boolean connected=false;
   for(int i=0;i<5;i++)
      {
      if(connectWiFi())
         {connected = true;
          mySerial.println("Connected to Wi-Fi...");
          break;
         }
      }
   if (!connected)
      {
      mySerial.println("Coudn't connect to Wi-Fi.");
      while(1);
      }
   delay(5000);
   mySerial.println("AT+CIPMUX=0");  // режим одиночного соединения
   }
void loop()
   {
   String cmd = "AT+CIPSTART=\"TCP\",\"";
```

```
   cmd += DST_IP;
   cmd += "\",8283";
   Serial.println(cmd);
   mySerial.println(cmd);
   if(mySerial.find("Error"))
      return;
   double val = analogRead(LM335);          // чтение показаний LM335
   double voltage = val*5.0/1024;           // перевод в вольты
   double temp = voltage*100 - 273.15;      // в градусы Цельсия
   cmd = "#A0:F3:C1:70:AA:94\n#2881C4BA0200003B1#"+String(temp)+"\n##";
   delay(3000);
   mySerial.print("AT+CIPSEND=");
   mySerial.println(cmd.length());
   delay(1000);
   Serial.println(">");
   mySerial.print(cmd);
   Serial.println(cmd);
   delay(3000);
   mySerial.println("AT+CIPCLOSE");
   delay(600000);
   }
// процедура установки Wi-Fi-соединения
boolean connectWiFi()
   {
   String cmd="AT+CWJAP=\"";
   cmd+=SSID;
   cmd+="\",\"";
   cmd+=PASS;
   cmd+="\"";
   mySerial.println(cmd);
   Serial.println(cmd);
   delay(2000);
   if(mySerial.find("OK"))
      {
      Serial.println("OK, Connected to Wi-Fi.");
      return true;
      }
   else
      {
      Serial.println("Can not connect to the Wi-Fi.");
      return false;
      }
   }
```

На странице http://arduino-kit.ru/0030 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 30.1 и 30.2.

31 Беспроводная связь. Модуль Bluetooth HC-05

В этом эксперименте рассмотрим работу модуля Bluetooth HC-05, позволяющего плате Arduino установить беспроводную связь и обмениваться данными с другими устройствами по протоколу Bluetooth.

Необходимые компоненты:

- контроллер Arduino UNO;
- модуль Bluetooth HC-05;
- плата для прототипирования;
- провода папа-папа;
- телефон или планшет с OS Android.

Bluetooth (с англ. – «голубой зуб») – одна из технологий беспроводной передачи данных. Спецификация была разработана в 1998 г. компанией Ericsson, а позднее оформлена группой Bluetooth Special Interest Group (SIG), официально зарегистрированной 20 мая 1999 г. Bluetooth позволяет объединять в локальные сети любую технику: от мобильного телефона и компьютера до холодильника. При этом одним из немаловажных параметров новой технологии являются низкая стоимость устройства связи (в пределах 20 долларов), его небольшие размеры (ведь речь идет о мобильных устройствах) и, что немаловажно, совместимость, простота встраивания в различные устройства. Мы будем использовать недорогой модуль HC-05. В нем используется чип BC417 плюс Flash-память и выводы GPIO. Чип поддерживает спецификацию Bluetooth v2.0 + EDR, AT-команды, может работать в режиме Master или Slave, поддерживает скорость обмена от 2400 до 1 382 400. Напряжение питание модуля составляет 3,3 В, ток потребления ~50 мА, что позволяет питать его от вывода Arduino +3,3 В. Для программирования модуля с помощью AT-команд необходимо на вывод PIO11 подать +3,3 В. Не на всех версиях модулей HC-05 вывод PIO11 выведен на разъем. В некоторых случаях потребуется аккуратно подпаять провод к выводу прямо на плате (рис. 31.1). Подключим модуль к плате Arduino и рассмотрим простейшие AT-команды. Схема подключения показана на рис. 31.2.

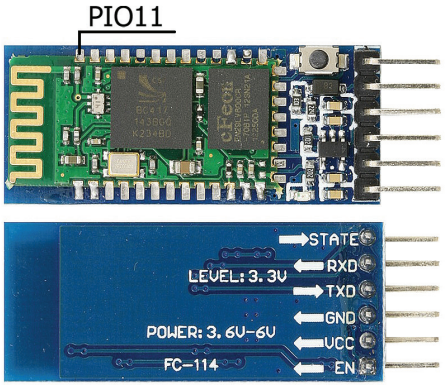

Рис. 31.1. Вид модуля Bluetooth HC-05

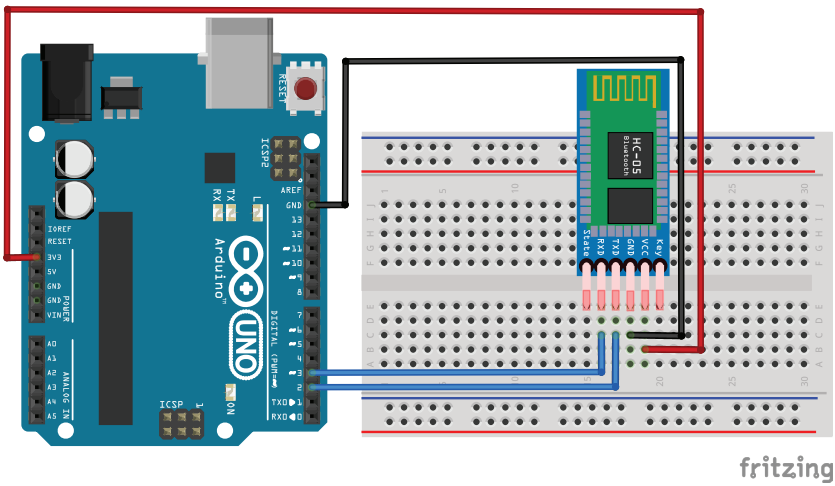

Рис. 31.2. Схема подключения модуля HC-05 к плате Arduino

Теперь на плату Arduino загрузим скетч, показанный в листинге 31.1.

Листинг 31.1

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // указываем пины RX и TX

void setup()
  {
```

```
   pinMode(2,INPUT);
   pinMode(3,OUTPUT);
   Serial.begin(38400);
   mySerial.begin(9600);
   }
void loop()
   {
   if (mySerial.available())
      {
      int c = mySerial.read();  // читаем из software-порта
      Serial.write(c);          // пишем в hardware-порт
      }
   if (Serial.available())
      {
      int c = Serial.read();    // читаем из hardware-порта
      mySerial.write(c);        // пишем в software-порт
      }
   }
```

Порядок подключения:

1. Подключаем модуль HC-05 к плате Arduino по схеме на рис. 31.2. Отключаем провод, ведущий к выводу 34 модуля (PIO11), от 3,3 В.
2. Загружаем на плату Arduino скетч из листинга 31.2.
3. Светодиод на плате должен быстро мигать. Если не мигает или мигает иначе, отключаем питание модуля от 3,3 В, затем снова подключаем питание.
4. Подключаем провод, ведущий к выводу 34 модуля (PIO11), к 3,3 В.
5. Открываем монитор последовательного порта Arduino и набираем AT-команды из табл. 31.1. Смотрим результат выполнения команд (рис. 31.3).

Таблица 31.1

AT-команда	Параметры, ответ	Описание
AT	Ответ модуля: OK	Тестовая команда
AT+NAME?	Ответ модуля: +NAME:<name>	Получить имя модуля
AT+NAME=<name>	<name> – имя Bluetooth-модуля. Ответ модуля: OK (или FAIL)	Установить новое имя модуля
AT+PSWD?	Ответ модуля: + PSWD:<password>	Получить код доступа к Bluetooth-модулю
AT+PSWD=<password>	<password> – код доступа к модулю. Ответ модуля: OK (или FAIL)	Установить код доступа к Bluetooth-модулю

Таблица 31.1 (окончание)

AT-команда	Параметры, ответ	Описание
AT+ROLE?	Ответ модуля: +ROLE:<role> – режим работы Bluetooth-модуля	Получить режим работы Bluetooth-модуля
AT+ROLE=<role>	<role> – режим работы: 0 – slave; 1 – master; 2 – slave-loop. Ответ модуля: OK (или FAIL)	Установить режим работы Bluetooth-модуля
AT+UART?	Ответ модуля: +UART:<p1>,<p2>,<p3> <p1> – скорость обмена; <p2> – стоп-бит; <p3> – бит паритета	Получить параметры обмена
AT+UART=<p1>,<p2>,<p3>	<p1> – скорость обмена (9600,19200,38400,57600,115200); <p2> – стоп-бит; <p3> – бит паритета. Ответ модуля: OK (или FAIL)	Установить параметры обмена
AT+ADDR?	Ответ модуля: +ADDR:<addr> <addr> – адрес Bluetooth-модуля NAP: UAP : LAP	Получить адрес модуля

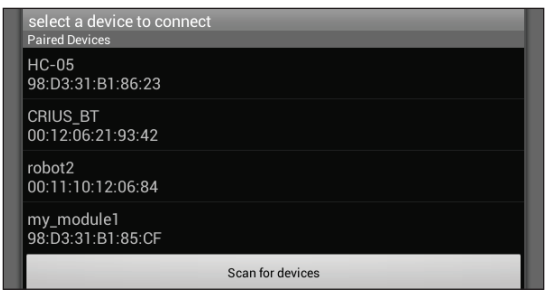

Рис. 31.3. Установка связи с модулем HC-05

Следующий шаг – двунаправленная передача данных между телефоном с OS Android и платой Arduino с модулем HC-05. Отсоединим контакт 34 Bluetooth-модуля от 3,3 В. Загрузим и установим на телефон из Play Market приложение Bluetooth Terminal (https://play.google.com/store/apps/details?id=Qwerty.BluetoothTerminal&hl=ru). Запустим программу и установим соединение с нашим модулем (см. рис. 31.3). Передаем Arduino и получаем (через монитор последовательного порта) из Arduino сообщения (см. рис. 31.4).

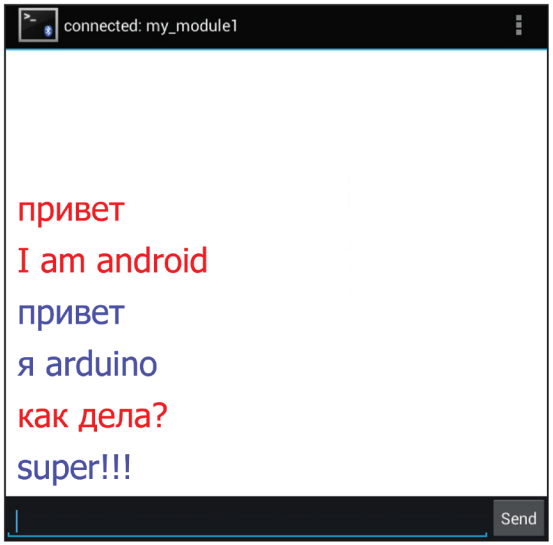

Рис. 31.4. Обмен сообщениями между телефоном и платой Arduino

На странице http://arduino-kit.ru/0031 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 32.1, и файл с полным списком AT-команд для модуля HC-05.

32 Беспроводная связь. Модуль GSM/GPRS SIM900

В этом эксперименте рассмотрим работу модуля GSM/GPRS shield – платы расширения, позволяющей Arduino работать в сетях сотовой связи по технологиям GSM/GPRS для приема и передачи данных, SMS и голосовой связи.

Необходимые компоненты:

- контроллер Arduino UNO;
- GSM/GPRS shield;
- работающая SIM-карта любого оператора;
- плата для прототипирования;
- датчик температуры LM335;
- резистор 2,2 кОм;
- провода папа-папа;
- блок питания +12 В 2 А.

GSM/GPRS shield на базе модуля SIMCom SIM900 выпускают несколько производителей, и платы имеют незначительные отличия. Также на некоторых платах расположены: слот для SIM-карты, стандартные 3,5 мм джек для аудиовхода и выхода и разъем для внешней антенны. На плате GSM/GPRS shild имеется несколько перемычек, позволяющих выбрать тип serial-соединения (hardware или software). GSM/GPRS shield имеет два способа включения – аппаратный (кратковременное нажатие кнопки PWRKEY) и программный (используется один из выходов Arduino). Рассмотрим пример отправки и получения SMS-сообщений с помощью GSM/GPRS shield. Каждые 30 минут будем отправлять на определенный номер показания аналогового датчика температуры LM335, подсоединенного к выводу A0. Схема соединений эксперимента показана на рис. 32.1.

Содержимое скетча для отправки SMS показано в листинге 32.1.

Листинг 32.1

```
// подключение библиотеки SoftwareSerial
#include <SoftwareSerial.h>
// номер телефона для отправки sms (поменяйте на свой)
```

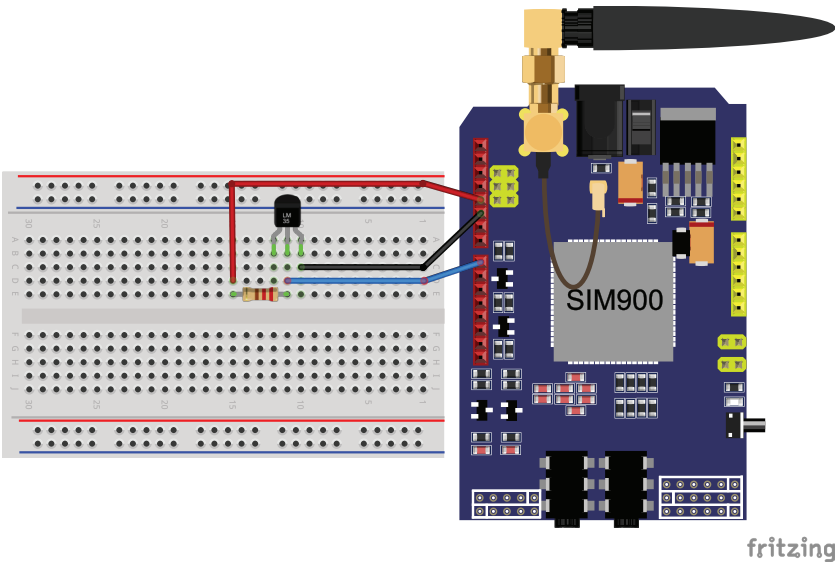

Рис. 32.1. Схема подключения модуля GSM/GPRS shield и датчика LM335. Шилд SIM900 установлен на Arduino UNO

```
#define PHONE "+79031111111"
// Выводы для SoftwareSerial (у вас могут быть 7,8)
SoftwareSerial Sim900Serial(2, 3);
const int lm335=A0;                  // для подключения LM335
unsigned long millis1;

void setup()
   {
   Sim900Serial(19200);              // the Hardware serial rate
   }
void loop()
   {
   if (millis()-millis1>30*60*1000)  // прошло 30 минут?
      {
      SendTextMessage();             // отправить sms
      millis1=millis();
      }
   }
// подпрограмма отправки sms
void SendTextMessage()
   {
```

```
// AT-команда установки text mode
Sim900Serial.print("AT+CMGF=1\r");
delay(100);
// номер телефона получателя
Sim900Serial.println("AT + CMGS = \"");
Sim900Serial.println(PHONE);
Sim900Serial.println("\"");
delay(100);
// сообщение - данные температуры
double val = analogRead(lm335);        // чтение
double voltage = val*5.0/1024;         // перевод в вольты
double temp = voltage*100 - 273.15;    // в градусы Цельсия

Sim900Serial.println(temp);
delay(100);
// ASCII код ctrl+z - окончание передачи
Sim900Serial.println((char)26);
delay(100);
Sim900Serial.println();
}
```

Порядок подключения:

1. Установим SIM-карту на GSM/GPRS Shield, а GSM/GPRS Shield – на Arduino. С помощью джамперов соединим контакты для работы через SoftwareSerial-эмуляцию.
2. Собираем схему, согласно рис. 32.1.
3. Загружаем в плату Arduino скетч из листинга 32.1.
4. На телефон, указанный в скетче, раз в 30 минут должны приходить sms-сообщения с данными температуры.

Теперь изменим скетч таким образом, чтобы Arduino отправляла sms-сообщение с данными температуры только при получении приходящего сообщения с текстом «temp». Содержимое скетча показано в листинге 32.2.

Листинг 32.2

```
#include <SoftwareSerial.h>
SoftwareSerial Sim900Serial(2, 3);
String currStr = "";     //
String phone = "";       //
// True, если текущая строка является sms-сообщением
boolean isStringMessage = false;

void setup()
  {
```

```
   Serial.begin(19200);
   Sim900Serial.begin(19200);
   // Настраиваем прием сообщений с других устройств
   Sim900Serial.print("AT+CMGF=1\r");
   delay(300);
   Sim900Serial.print("AT+IFC=1, 1\r");
   delay(300);
   Sim900Serial.print("AT+CPBS=\"SM\"\r");
   delay(300);
   Sim900Serial.print("AT+CNMI=1,2,2,1,0\r");
   delay(500);
   }

void loop()
   {
   if (!Sim900Serial.available())
      return;
   char currSymb = Sim900Serial.read();
   if ('\r' == currSymb)
      {
      if (isStringMessage)   // текущая строка - sms-сообщение,
         {
         if (!currStr.compareTo("temp"))   // текст sms - temp
            {
            // отправить sms на приходящий номер
            Sim900Serial.print("AT+CMGF=1\r");
            delay(100);
            Sim900Serial.print("AT + CMGS = \"");
            Sim900Serial.print(phone);
            Sim900Serial.println("\"");
            delay(100);
            double val = analogRead(A0);           // чтение
            double voltage = val*5.0/1024;         // перевод в вольты
            double temp = voltage*100 - 273.15;   // в градусы Цельсия
            Serial.println(temp);
            Sim900Serial.println(temp);
            delay(100);
            Sim900Serial.println((char)26);
            delay(100);
            Sim900Serial.println();
            }
         Serial.println(currStr);
         isStringMessage = false;
         }
      else
```

```
        {
        if (currStr.startsWith("+CMT")) {
            Serial.println(currStr);
        // выделить из сообщения номер телефона
        phone=currStr.substring(7,19);
        Serial.println(phone);
        // если текущая строка начинается с "+CMT",
        // то следующая строка является сообщением
        isStringMessage = true;
        }
    }
    currStr = "";
    }
  else if ('\n' != currSymb)
    {
    currStr += String(currSymb);
    }
}
```

На странице http://arduino-kit.ru/0032 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 32.1 и 32.2.

33 Модуль GPS. Принцип работы, подключение, примеры

В этом эксперименте рассмотрим работу модуля GPS-приемника, позволяющего определять наше местоположение с помощью глобальной системы GPS, и подключение данного приемника к плате Arduino.

Необходимые компоненты:

- контроллер Arduino UNO;
- GPS-приемник V.KEL VK16E;
- плата для прототипирования;
- провода папа-папа.

GPS (Global Positioning System) – это система, позволяющая с точностью не хуже 100 м определить местоположение объекта, то есть определить его широту, долготу и высоту над уровнем моря, а также направление и скорость его движения. Кроме того, с помощью GPS можно определить время с точностью до 1 наносекунды.

GPS состоит из совокупности определенного количества искусственных спутников Земли (спутниковой системы NAVSTAR) и наземных станций слежения, объединенных в общую сеть. В качестве абонентского оборудования служат индивидуальные GPS-приемники, способные принимать сигналы со спутников и по принятой информации вычислять свое местоположение. Мы используем GPS-приемник V.KEL VK16E (см. рис. 33.1).

Рис. 33.1. GPS-приемник V.KEL VK16E

Назначение выводов:

- BOOT – зарезервировано;
- GND – общий вывод;
- RX – вход для данных в последовательном формате UART;
- TX – выход для данных в последовательном формате UART;
- VCC – вход напряжения питания от 3,3 до 5 В;
- PPS – выход импульсов времени.

Холодный старт происходит, когда GPS-приемник был выключен длительное время, перемещался в выключенном состоянии на значительное расстояние или его часы не совпадают с данными спутника. При холодном старте со спутников скачивается альманах. Время обновления альманаха – от 5 до 15 минут в зависимости от условий приема и количества видимых спутников. Особенность – приемник в это время должен быть неподвижен. Теплый старт происходит, когда приемник был выключен более 30 мин. При этом происходит прием уточняющих данных. Занимает 0,5–1,5 мин. Горячий старт происходит, когда приемник был отключен непродолжительное время. Данные считаются свежими, и приемник просто находит спутники (опираясь на данные альманаха). Для проверки работоспособности подключим модуль по последовательному порту к компьютеру с ОС Windows, используя USB-TTL-адаптер, и запустим программу MiniGPS_v1.7.1.exe. Программа показывает количество найденных приемником спутников и в случае их достаточного числа показывает наше местоположение – географические широту и долготу (см. рис. 33.2). Мигание зеленого светодиода сигнализирует о достаточном для определения положения количестве спутников.

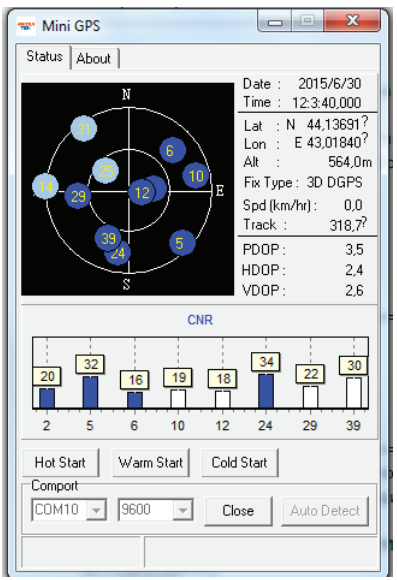

Рис. 33.2. Просмотр данных GPS-приемника в программе MiniGPS

Теперь подключим GPS-приемник к плате Arduino и будем выводить данные о местоположении – географические широту и долготу в монитор последовательного порта Arduino. Схема подключения показана на рис. 33.3.

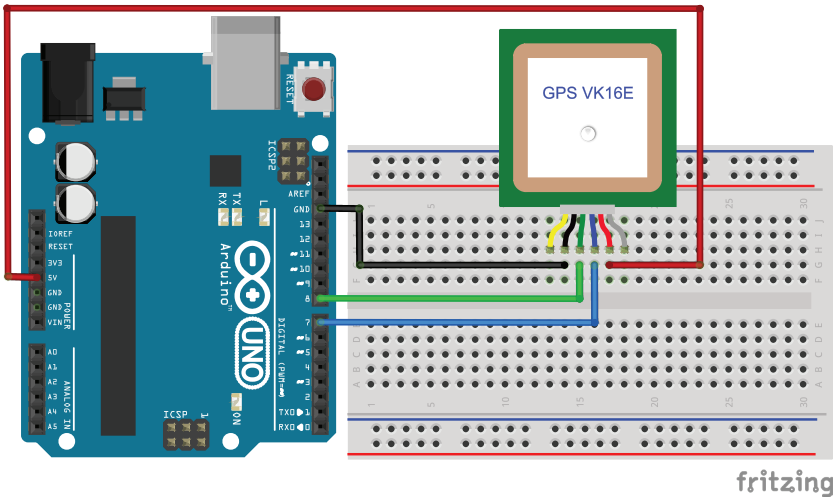

Рис. 33.3. Схема подключения GPS-приемника к Arduino

Приступим к написанию скетча. Будем использовать библиотеки SoftwareSerial и TinyGPS, позволяющую выделять нужные данные из всего потока, передаваемого приемником. Содержимое скетча показано в листинге 33.1.

Листинг 33.1

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
TinyGPS gps;
SoftwareSerial gpsSerial(7, 8);
bool newdata = false;
unsigned long start;
long lat, lon;
unsigned long time, date;


void setup()
   {
   gpsSerial.begin(9600);     // скорость обмена с GPS-приемником
   Serial.begin(9600);
   Serial.println("Waiting data of GPS...");
   }
void loop()
   {
```

```
   // задержка в секунду между обновлениями координат

   if (millis() - start > 1000)
     {
      newdata = readgps();
      if (newdata)
         {
         start = millis();
         gps.get_position(&lat, &lon);
         gps.get_datetime(&date, &time);
         Serial.print("Lat: "); Serial.print(lat);
         Serial.print(" Long: "); Serial.print(lon);
         Serial.print(" Date: "); Serial.print(date);
         Serial.print(" Time: "); Serial.println(time);
         }
      }
   }
// проверка наличия данных
bool readgps()
   {
   while (gpsSerial.available())
      {
       int b = gpsSerial.read();
//в TinyGPS есть ошибка: не обрабатываются данные с \r и \n
   if('\r' != b)
         {
         if (gps.encode(b))
         return true;
         }
      }
   return false;
   }
```

Порядок подключения:

1. Собираем схему, согласно рис. 33.3.
2. Загружаем в плату Arduino скетч из листинга 33.1.
3. Ждем мигания зеленого светодиода на приемнике GPS, сигнализирующего о наличии данных о местоположении.
4. Смотрим в мониторе последовательного порта Arduino вывод данных широты и долготы. Также получаем текущую дату и время по Гринвичу (рис. 33.4).

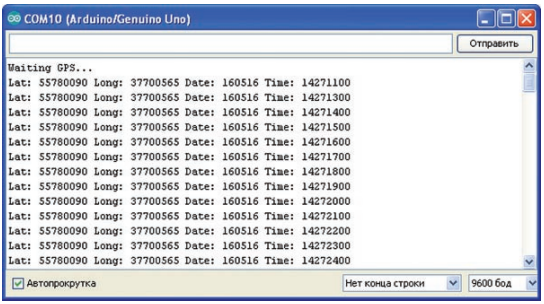

Рис. 33.4. Вывод данных GPS в монитор последовательного порта Arduino

На странице http://arduino-kit.ru/0033 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 33.1, Arduino-библиотеку TinyGPS и программу MiniGPS.

Встроенные функции языка Arduino

Программирование микроконтроллеров Arduino осуществляется на специальном языке программирования, основанном на C/C ++. Этот язык считается сложным и имеет высокий порог вхождения. Но для программирования Arduino используется упрощенная версия этого языка программирования. Также для упрощения разработки прошивок существует множество функций, классов, методов и библиотек. Благодаря этому работать с этими микроконтроллерами очень удобно и легко.

Рассмотрим основные встроенные функции языка Arduino.

В первую очередь рассмотрим функции для работы с портами микроконтроллеров Arduino.

Функция pinMode()

Функция pinMode() конфигурирует режим работы указанного вывода: как вход либо как выход.

Синтаксис
pinMode(pin, mode)

Параметры
- pin: номер вывода, режим работы которого будет конфигурироваться.
- mode: принимает значения INPUT, OUTPUT или INPUT_ PULLUP.

Возвращаемые значения
Нет.

Пример

```
void setup()
{
   // устанавливаем режим работы вывода 13 , OUTPUT - выход
   pinMode(13, OUTPUT);
}

void loop()
{;}
```

Функция digitalWrite()

Функция digitalWrite() отправляет на цифровой вывод значение HIGH или LOW.

Если функцией pinMode() вывод сконфигурирован как выход (OUTPUT), то при выполнении функции digitalWrite() его напряжение будет изменено на соответствующее значение: 5 В (либо 3,3 В для плат, работающих от 3,3 В) при отправке HIGH, 0 В (земля) – при LOW.

Если вывод сконфигурирован как вход INPUT, то отправка функцией digitalWrite() значения HIGH приведет к подключению внутреннего подтягивающего резистора номиналом 20 кОм. Запись значения LOW приведет к отключению подтяжки. Внутренний подтягивающий резистор может обеспечить только тусклое свечение светодиода. Поэтому если светодиод горит, но очень тускло, наиболее вероятная причина этого – подтягивающий резистор. Для решения данной проблемы необходимо перевести соответствующий вывод в режим выхода с помощью функции pinMode().

Синтаксис
digitalWrite(pin, value)

Параметры
- pin: номер вывода.
- value: значение HIGH или LOW.

Возвращаемые значения
Нет.

Пример

```
// контакт подключения светодиода
int ledPin = 13;

void setup()
{
   // конфигурируем контакт подключения светодиода
   // как OUTPUT - выход
   pinMode(ledPin, OUTPUT);
}

void loop()
{
   digitalWrite(ledPin, HIGH);       // включаем светодиод
   delay(1000);                      // ждем 1 секунду
```

```
   digitalWrite(ledPin, LOW);        // выключаем светодиод
   delay(1000);                      // ждем 1 секунду
}
```

Программа устанавливает на выводе 13 высокий уровень HIGH, выдерживает паузу в 1 секунду, после чего возвращает вывод в низкий уровень LOW.

Функция digitalRead()

Функция digitalRead() считывает уровень сигнала HIGH или LOW с указанного цифрового вывода.

Синтаксис
digitalRead(pin)

Параметры
- pin: номер цифрового вывода, с которого необходимо считать значение (*int*).

Возвращаемые значения
HIGH или LOW.

Пример
Программа устанавливает на выводе 13 тот же уровень сигнала, что и на выводе 7.

```
// контакт подключения светодиода
int ledPin = 13;
// контакт подключения кнопки
int inPin = 7;
// переменная для хранения считанного значения
int val = 0;

void setup()
{
   // конфигурируем цифровой вывод  как выход
   pinMode(ledPin, OUTPUT);
   // конфигурируем цифровой вывод  как вход
   pinMode(inPin, INPUT);
}

void loop()
{
   // считываем значение с кнопки
```

```
   val = digitalRead(inPin);
   // выводим на светодиод уровень сигнала на кнопке
   digitalWrite(ledPin, val);
}
```

Функция analogRead()

Функция digitalRead() считывает величину напряжения с указанного аналогового вывода. В составе Ардуино есть 6-канальный (8-канальный – в Mini и Nano, 16 – в Mega) 10-битный аналогово-цифровой преобразователь, который преобразовывает входное напряжение из диапазона 0–5 В в целочисленные значения в пределах от 0 до 1023 соответственно. Разрешающая способность АЦП составляет: 5 В/1024 значения или 0,0049 В (4,9 мВ) на одно значение. Входной диапазон и разрешающая способность могут меняться с помощью функции analogReference().

Для считывания значения с аналогового входа требуется около 100 микросекунд (0,0001 с), поэтому максимальная частота опроса вывода приблизительно равна 10 000 раз в секунду.

Синтаксис
analogRead(pin)

Параметры

- pin: номер вывода, с которого будет считываться напряжение (0–5 для большинства плат, 0–7 для Mini и Nano, 0–15 для Mega).

Возвращаемые значения
Целое число int (от 0 до 1023).

Пример

```
// Выход подключения потенциометра
int analogPin = A0;
// переменная для хранения считанного значения
int val = 0;

void setup()
{
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop()
{
```

```
  // считываем напряжение с аналогового входа
  val = analogRead(analogPin);
  // отправляем значение в последовательный порт
  Serial.println(val);
}
```

Функция analogWrite()

Функция analogWrite() формирует заданное аналоговое напряжение на выводе в виде ШИМ-сигнала. Может использоваться для варьирования яркости свечения светодиода или управления скоростью вращения двигателя. После вызова analogWrite() на выводе будет непрерывно генерироваться ШИМ-сигнал с заданным коэффициентом заполнения до следующего вызова функции analogWrite() (либо до момента вызова digitalRead() или digitalWrite(), взаимодействующих с этим же выводом). Частота ШИМ составляет приблизительно 490 Гц.

На большинстве плат Arduino (на базе микроконтроллеров ATmega168 или ATmega328) функция analogWrite() работает с выводами 3, 5, 6, 9, 10 и 11. На Arduino Mega функция работает с выводами со 2 по 13. На более старых версиях Arduino (на базе микроконтроллера ATmega8) функция analogWrite() работает только с выводами 9, 10 и 11.

Arduino Due поддерживает функцию analogWrite() для выводов со 2 по 13, а также для выводов DAC0 и DAC1. В отличие от ШИМ-выводов, DAC0 и DAC1 являются выводами цифроаналоговых преобразователей, поэтому при вызове analogWrite() ведут себя как обычные аналоговые выходы.

При работе с analogWrite() предварительный вызов функции pinMode() для переключения выводов в режим «выход» не требуется.

Синтаксис
analogWrite(pin, value)

Параметры
- pin: вывод, на котором будет формироваться напряжение.
- value: коэффициент заполнения – лежит в пределах от 0 (всегда выключен) до 255 (всегда включен).

Возвращаемые значения
Нет.

Пример
Формирование на выводе, управляющем светодиодом, напряжения, пропорционального напряжению на потенциометре.

```
// контакт подключения светодиода
int ledPin = 9;
// контакт подключения потенциометра
int analogPin = A0;
// переменная для хранения считанного значения
int val = 0;

void setup()
{
   // конфигурируем цифровой вывод  в режим OUTPUT
   pinMode(ledPin, OUTPUT);
}

void loop()
{
   // считываем значение с аналогового входа
   val = analogRead(analogPin);
   // перевод значения из интервала 0-1023
   // в интервал 0-255
   analogWrite(ledPin, map(val,0,1023,0,255));
}
```

Функция tone()

Функция tone() генерирует на выводе прямоугольный сигнал заданной частоты (с коэффициентом заполнения 50%). Функция также позволяет задавать длительность сигнала. Однако если длительность сигнала не указана, он будет генерироваться до тех пор, пока не будет вызвана функция noTone(). Для воспроизведения звука вывод можно подключить к зуммеру или динамику.

В каждый момент времени может генерироваться только один сигнал заданной частоты. Если сигнал уже генерируется на каком-либо выводе, то использование функции tone() для этого вывода просто приведет к изменению частоты этого сигнала. В то же время вызов функции tone() для другого вывода не будет иметь никакого эффекта.

Использование функции tone() может влиять на ШИМ-сигнал на выводах 3 и 11 (на всех платах, кроме Mega).

Синтаксис
tone(pin, frequency)
tone(pin, frequency, duration)

Параметры
- pin: вывод, на котором будет генерироваться сигнал.
- frequency: частота сигнала в герцах (тип unsigned int).
- duration: длительность сигнала в миллисекундах (опционально) (тип unsigned long).

Возвращаемые значения
Нет.

Пример
Проигрывание на динамике мелодии до-ре-ми-фа-соль-ля-си.

```
// контакт подключения динамика
int speakerPin = 9;

void setup()
{
  // конфигурируем цифровой вывод  в режим OUTPUT
  pinMode(speakerPin, OUTPUT);
}

void loop()
{
  // проигрываем последовательно ноты
  tone(speakerPin,261,900);              // до
  delay(1000);
  tone(speakerPin,293,900);              // ре
  delay(1000);
  tone(speakerPin,329,900);              // ми
  delay(1000);
  tone(speakerPin,349,900);              // фа
  delay(1000);
  tone(speakerPin,392,900);              // соль
  delay(1000);
  tone(speakerPin,440,900);              // ля
  delay(1000);
  tone(speakerPin,494,900);              // си
  delay(1000);
}
```

Функция noTone()

Функция noTone() прекращает генерирование прямоугольного сигнала после использования функции tone(). Если сигнал не генерируется, функция ни к чему не приводит.

Синтаксис
noTone(pin)

Параметры
- pin: вывод, на котором следует прекратить генерирование сигнала.

Возвращаемые значения
Нет.

Пример
Проигрывание на динамике мелодии до-ре-ми-фа-соль-ля-си с использованием функции noTone()

```
// контакт подключения динамика
int speakerPin = 9;

// массив со значениями частот
int freq[] = {261,293,329,349,392,440,494};

void setup()
{
   // конфигурируем цифровой вывод  в режим OUTPUT
   pinMode(speakerPin, OUTPUT);
}

void loop()
{
   // проигрываем последовательно ноты
   // без указания длительности
   for(int i=0;i<7;i++)
   {
       tone(speakerPin,freq[i]);     // играем ноту
       delay(900);
       noTone(speakerPin);           // выключить ноту
       delay(100);
   }
```

Функция shiftOut()

Функция shiftOut() осуществляет побитовый сдвиг и вывод байта данных, начиная с самого старшего (левого) или младшего (правого) значащего бита. Функция поочередно отправляет каждый бит на указанный вывод данных, после чего формирует импульс (высокий уровень, затем низкий) на тактовом выводе, сообщая внешнему устройству о поступлении нового бита. Функция является программной реализацией протокола SPI.

Синтаксис
shiftOut(dataPin, clockPin, bitOrder, value)

Параметры
- dataPin: вывод, которому будет отправляться каждый бит из сдвигаемого байта данных.
- clockPin: тактовый вывод, который будет переключаться каждый раз, когда на выводе dataPin устанавливается корректное значение.
- bitOrder: характеризует порядок, в котором будут сдвигаться и выводиться биты; может принимать значения MSBFIRST или LSBFIRST (Most Significant Bit First – старший значащий бит первым или Least Significant Bit – младший значащий бит первым).
- value: сдвигаемый байт данных (тип byte).

Возвращаемые значения
Нет.

Пример
Бегущий огонь на 8 светодиодах, подключенных к сдвиговому регистру 74HC595.

```
// Вывод подключения ST_CP микросхемы 74HC595
int latchPin = 8;
// Вывод подключения SH_CP микросхемы 74HC595
int clockPin = 12;
// Вывод подключения DS микросхемы 74HC595
int dataPin = 11;
// данные для бегущего огня
byte data[]={1,2,4,8,16,32,64,128};

void setup() {
   // конфигурация контактов как OUTPUT
```

```
   pinMode(latchPin, OUTPUT);
   pinMode(clockPin, OUTPUT);
   pinMode(dataPin, OUTPUT);
}

void loop() {
   // запустить бегущий огонь на 8 светодиодах
   for (int j = 0; j < 8; j++) {
      // выставить LOW на latchPin
      digitalWrite(latchPin, LOW);
      shiftOut(dataPin, clockPin, LSBFIRST, data[j]);
      // вывести полученные данные на светодиоды
      digitalWrite(latchPin, HIGH);
      delay(1000);
   }
}
```

Функция shiftIn()

Функция shiftIn() осуществляет побитовый сдвиг и считывание байта данных, начиная с самого старшего (левого) или младшего (правого) значащего бита. Процесс считывания каждого бита заключается в следующем: тактовый вывод переводится в высокий уровень, считывается очередной бит из линии данных, после чего тактовый вывод сбрасывается в низкий уровень. Данная функция является программной реализацией протокола SPI.

Синтаксис
byte incoming = shiftIn(dataPin, clockPin, bitOrder)

Параметры
- dataPin: вывод, с которого будет считываться каждый бит (тип int).
- clockPin: тактовый вывод, который будет переключаться при считывании с dataPin.
- bitOrder: порядок, в котором будут сдвигаться и считываться биты; может принимать значения MSBFIRST или LSBFIRST (Most Significant Bit First – старший значащий бит первым или Least Significant Bit – младший значащий бит первым).

Возвращаемые значения
Считанное значение (тип byte).

Функция pulseIn()

Функция pulseIn() считывает длительность импульса (любого – HIGH или LOW) на выводе. Например, если заданное значение (**value**) – **HIGH**, то функция **PulseIn()** ожидает появления на выводе сигнала **HIGH**, затем засекает время и ожидает переключения вывода в состояние **LOW**, после чего останавливает отсчет времени. Функция возвращает длительность импульса в микросекундах либо 0 в случае отсутствия импульса в течение определенного тайм-аута. Функция работает с импульсами длительностью от 10 микросекунд до 3 минут.

Синтаксис
pulseIn(pin, value)
pulseIn(pin, value, timeout)

Параметры
- pin: номер вывода, с которого необходимо считыть импульс (тип int).
- value: тип считываемого импульса: HIGH или LOW (тип int).
- timeout (опционально): время ожидания импульса в микросекундах; значение по умолчанию – одна секунда (тип unsigned long).

Возвращаемые значения
Длительность импульса (в микросекундах) либо 0 в случае отсутствия импульса в течение тайм-аута (unsigned long).

Пример
```
int pin = 7;
unsigned long duration;

void setup()
{
   pinMode(pin, INPUT);
}

void loop()
{
   duration = pulseIn(pin, HIGH);
}
```

Прерывания – очень важный механизм Arduino, позволяющий внешним устройствам взаимодействовать с контроллером при возникновении разных событий.

Функция attachInterrupt()

Функция attachInterrupt() задает функцию, которую необходимо вызвать при возникновении внешнего прерывания. Заменяет предыдущую функцию, если таковая была ранее ассоциирована с прерыванием. В большинстве плат Ардуино существует два внешних прерывания: номер 0 (цифровой вывод 2) и 1 (цифровой вывод 3). Номера выводов для внешних прерываний, доступные в тех или иных платах Ардуино, приведены в таблице ниже:

Плата	0	1	2	3	4	5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due	Позволяет прикрепить функцию прерывания для всех доступных контактов					

Синтаксис
attachInterrupt(interrupt, function, mode)
attachInterrupt(pin, function, mode) (только для Arduino Due)

Параметры
- interrupt: номер прерывания.
- pin: номер цифрового порта (только для Arduino Due).
- function: функция, вызываемая прерыванием, функция должна быть без параметров и не возвращать значений.
- mode: задает режим обработки прерывания, допустимо использование следующих констанст:
  - LOW: вызывает прерывание, когда на порту LOW;
  - CHANGE: прерывание вызывается при смене значения на порту, с LOW на HIGH и наоборот;
  - RISING: прерывание вызывается только при смене значения на порту с LOW на HIGH;
  - FALLING: прерывание вызывается только при смене значения на порту с HIGH на LOW.

Возвращаемые значения
Нет.

Использование прерываний
Прерывания могут использоваться для того, чтобы заставить микроконтроллер выполнять определенные участки программы автоматически и позволяют решить проблемы с синхронизацией. Типичными

задачами, требующими использования прерываний, являются считывание сигнала энкодера (датчика вращения) либо мониторинг воздействий пользователя.

В программе, от которой требуется постоянно обрабатывать сигнал от датчика вращения (не пропуская при этом ни единого импульса), очень сложно реализовать еще какую-либо функциональность, помимо опроса. Это объясняется тем, что для своевременной обработки поступающих импульсов программа должна постоянно опрашивать выводы, к которым подключен энкодер. При работе с другими типами датчиков часто требуется подобный динамический интерфейс: например, при опросе звукового датчика с целью различить щелчок или при работе с инфракрасным щелевым датчиком (фотопрерывателем) для распознавания момента броска монеты. Во всех этих примерах использование прерываний позволит разгрузить микроконтроллер для выполнения другой работы, при этом не теряя контроль над поступающими сигналами.

Внутри функции-обработчика прерывания функция delay() не будет работать; значения, возвращаемые функцией millis(), не будут увеличиваться. Также будут потеряны данные, полученные по последовательному интерфейсу во время выполнения обработчика прерывания. Любые переменные, которые изменяются внутри функции обработчика, должны быть объявлены как volatile.

Пример

```
int pin = 13;
volatile int state = LOW;

void setup()
{
   pinMode(pin, OUTPUT);
   attachInterrupt(0, blink, CHANGE);
}

void loop()
{
   digitalWrite(pin, state);
}

void blink()
{
   state = !state;
}
```

Функция detachInterrupt()

Функция detachInterrupt() запрещает заданное прерывание.

Синтаксис

detachInterrupt(interrupt)
detachInterrupt(pin) (только для Arduino Due)

Параметры

- interrupt: номер прерывания, которое необходимо запретить.
- pin: номер вывода, соответствующее прерывание которого необходимо запретить (только для Arduino Due).

Петин Виктор Александрович
Биняковский Александр Анатольевич

Практическая энциклопедия Arduino

Издание второе, дополненное

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 13,49. Тираж 200 экз.

Веб-сайт издательства: www.дмк.рф