Аналогично входы могут быть цифровыми (например, определяющими факт нажатия кнопки) или аналоговыми (например, подключенные к фотоэлементу).

В книге, по существу описывающей приемы программирования, а не аппаратные средства, мы попробуем избежать сползания в обсуждение специфики электроники. Однако вы лучше поймете рассматриваемые здесь проблемы, если вооружитесь мультиметром и коротким куском провода в изоляции.

Желающим поближе познакомиться с электроникой я могу порекомендовать мою книгу Hacking Electronics (TAB/McGraw-Hill, 2013)[1].

Цифровые выходы

В предыдущих главах использовался светодиод, подключенный к контакту **13** на плате Arduino. Например, в главе 5 этот светодиод использовался как источник сигналов азбуки Морзе. На плате Arduino есть целая коллекция контактов для вывода цифровых сигналов.

Давайте поэкспериментируем с одним из таких контактов. Возьмем, к примеру, контакт **4** и посмотрим, как он действует, а чтобы вы могли наблюдать за происходящим, подключите свой мультиметр к плате Arduino. Как это сделать, показано на рис. 6.1. Если в комплект мультиметра входят зажимы типа «крокодил», зачистите изоляцию на концах отрезков провода и с одной стороны зафиксируйте концы проводов в зажимах, а другие их концы вставьте в гнезда на плате Arduino. Если таких зажимов у вас нет, просто накрутите концы проводов на щупы мультиметра.

Мультиметр должен быть настроен на измерение постоянного напряжения в диапазоне 0...20 В. Отрицательный провод (черный) следует подключить к «земле» (контакт **GND**), а положительный — к контакту **D3**. Провода одним концом должны быть подключены к щупам мультиметра, а другим вставлены в гнезда на плате Arduino.

---

[1] *Монк С.* Практическая электроника: иллюстрированное руководство для радиолюбителей. М.: Вильямс, 2016. — *Примеч. пер.*

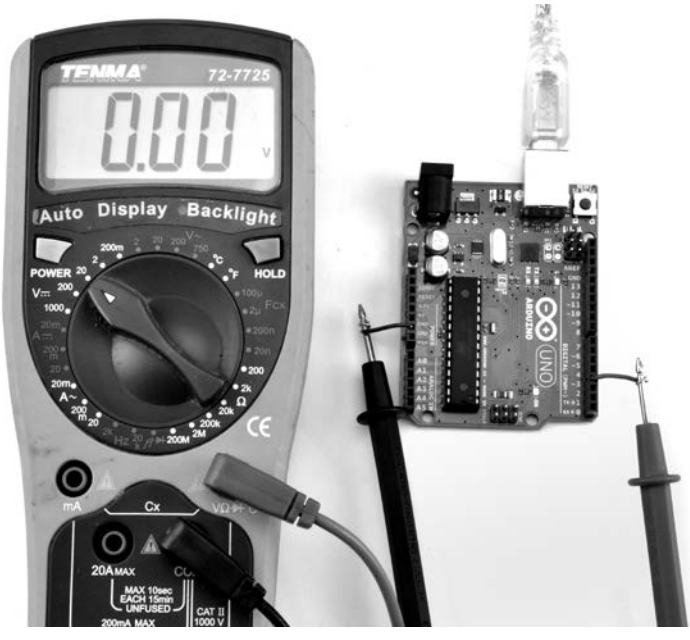

Рис. 6.1. Измерение напряжения на выходе с помощью мультиметра

Загрузите скетч 6-01:

```
//sketch 6-01
const int outPin = 4;

void setup()
{
  pinMode(outPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Enter 1 or 0");
}

void loop()
{
  if (Serial.available() > 0)
  {
    char ch = Serial.read();
    if (ch == '1')
    {
      digitalWrite(outPin, HIGH);
    }
```

```
    else if (ch == '0')
    {
      digitalWrite(outPin, LOW);
    }
  }
}
```

В начале скетча можно видеть команду `pinMode`. Она должна использоваться для каждого контакта, с которым предполагается работать, чтобы Arduino могла настроиться на работу с электроникой, подключенной к этим контактам, и использовать их как входы или выходы:

```
pinMode(outPin, OUTPUT);
```

Как нетрудно догадаться, `pinMode` — это встроенная функция. Ее первый аргумент — номер контакта (значение типа `int`), а второй аргумент — режим работы, `INPUT` (вход), `INPUT_PULLUP` (вход с подтягивающим резистором) или `OUTPUT` (выход). Обратите внимание на то, что имена режимов должны записываться только прописными буквами.

Функция `loop` ожидает команды `1` или `0` из монитора последовательного порта. Если будет получена команда `1`, она включит контакт `3`, если `0` — выключит.

Выгрузите скетч в плату Arduino и откройте монитор последовательного порта, как показано на рис. 6.2.

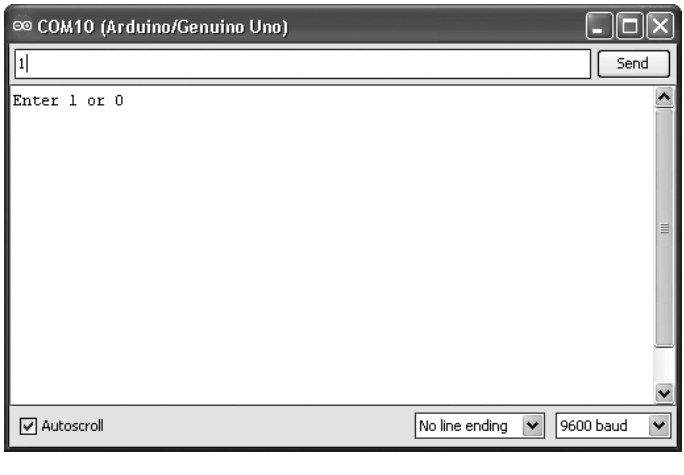

Рис. 6.2. Монитор последовательного порта

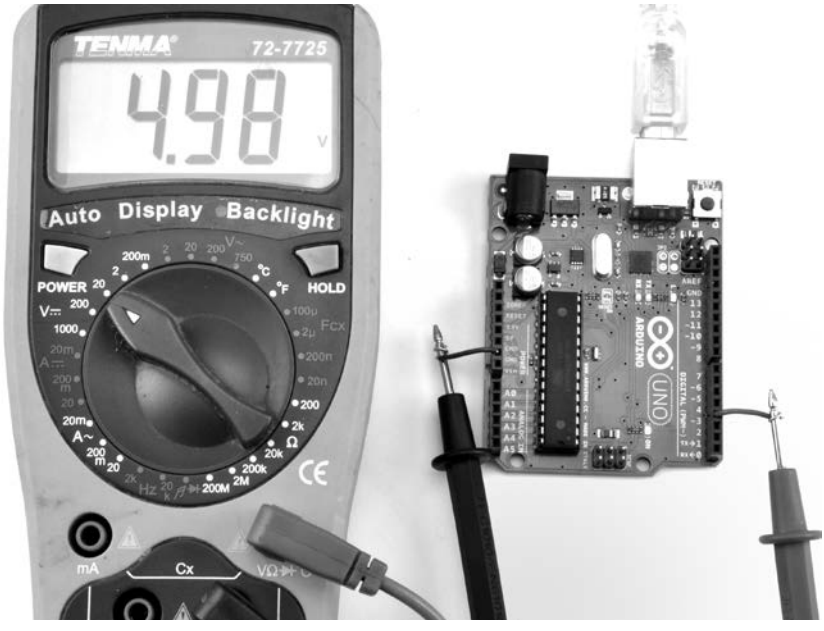

Рис. 6.3. На выход подан высокий уровень напряжения

Если мультиметр соединен с контактами на плате Arduino и включен, вы должны увидеть, как изменяется напряжение между значениями 0 и 5 В (приблизительно) при отправке команд из монитора последовательного порта, нажимая клавиши **1** и затем **Return** или **0** и затем **Return**. На рис. 6.3 изображены показания мультиметра после отправки команды **1**.

Если для нужд проекта окажется недостаточно контактов в группе с меткой **D**, для вывода цифровых сигналов можно дополнительно использовать контакты из группы с меткой **A** (analog — аналоговые). Для этого достаточно добавить букву A перед номером аналогового контакта, чтобы получилось, например, **A0**. Чтобы опробовать такую возможность, измените первую строку в скетче 6-01, подставив вместо номера контакта его имя `A0`, и подключите положительный провод мультиметра к контакту **A0** на плате Arduino.

Это все, что можно рассказать о цифровых выходах, поэтому теперь перейдем к цифровым входам.

Цифровые входы

Цифровые входы часто используются для определения момента, когда будет включен или выключен некоторый выключатель. Цифровой вход может иметь только одно из двух состояний: включено или выключено. Если напряжение на входе меньше 2,5 В (половина от 5 В), считается, что вход имеет состояние 0 (выключено), а если больше 2,5 В — состояние 1 (включено).

Отсоедините мультиметр и выгрузите в плату скетч 6-02:

```
//sketch 6-02
const int inputPin = 5;

void setup()
{
  pinMode(inputPin, INPUT);
  Serial.begin(9600);
}

void loop()
{
  int reading = digitalRead(inputPin);
  Serial.println(reading);
  delay(1000);
}
```

Как и при использовании выходов, нужно в функции `setup` сообщить плате Arduino, что некоторый контакт будет использоваться как цифровой вход. Получить состояние цифрового входа можно с помощью функции `digitalRead`. Она возвращает 0 или 1.

Нагрузочные резисторы

Скетч один раз в секунду читает состояние контакта цифрового входа и выводит полученное значение в монитор последовательного порта. Поэтому не забудьте открыть монитор после выгрузки скетча. Вы должны увидеть, как один раз в секунду появляется новое значение.

Вставьте один конец провода в гнездо **D5** и зажмите между пальцами другой его конец, как показано на рис. 6.4.

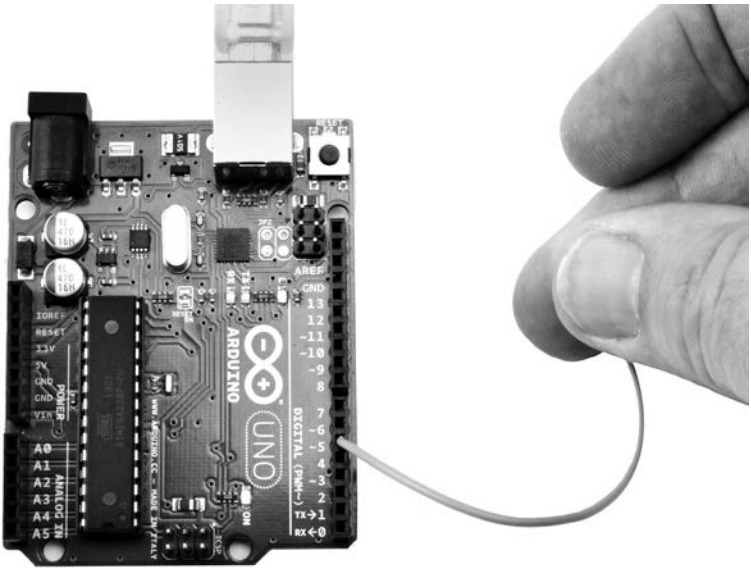

Рис. 6.4. Цифровой вход и человек как антенна

Продолжайте удерживать конец провода пальцами и следите за появлением значений в окне монитора последовательного порта. Вы должны увидеть смесь единиц и нулей. Причина такого поведения заключается в том, что входы на плате Arduino очень чувствительны. Ваше тело выступает в качестве антенны, передавая на провод наведенное электричество.

Теперь вставьте конец провода, который зажимали между пальцами, в гнездо с подписью **+5V**, как показано на рис. 6.5. На мониторе должны отображаться только единицы.

Теперь извлеките конец провода из гнезда **+5V** и вставьте его в гнездо **GND**. Как можно догадаться, на мониторе теперь должна отображаться последовательность нулей.

Обычно цифровые входы используются для подключения выключателей. На рис. 6.6 показано, как, на первый взгляд, должно выглядеть такое подключение. Проблема здесь в том, что, если выклю-

Рис. 6.5. Соединение контакта 5 с контактом +5V

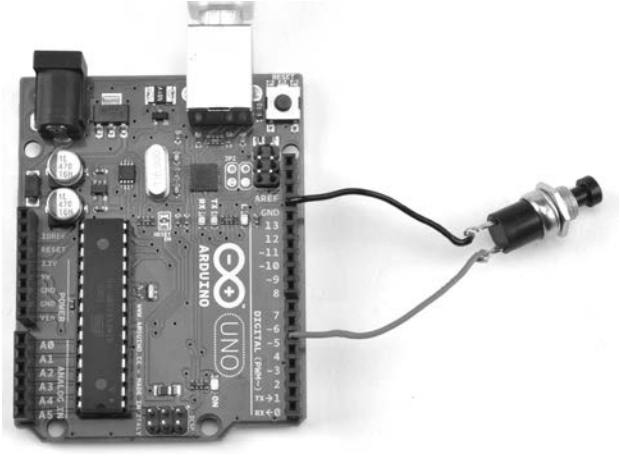

Рис. 6.6. Подключение выключателя к плате Arduino

чатель не замкнут, вход оказывается не соединен ни с чем. В этом случае говорят, что он висит в воздухе и легко может подавать ложные сигналы. Необходимо сделать поведение цифрового входа более предсказуемым, а добиться этого можно с помощью так называемого нагрузочного резистора. Далее вы узнаете, как задействовать внутренние нагрузочные резисторы, встроенные в Arduino, и избежать

необходимости подключения внешних резисторов. На рис. 6.7 показан стандартный способ подключения нагрузочного резистора. Когда выключатель не замкнут, через резистор подается напряжение 5 В на контакт, висящий в воздухе. Когда выключатель замыкается, цифровой вход соединяется непосредственно с «землей» и напряжение на нем падает до 0 В.

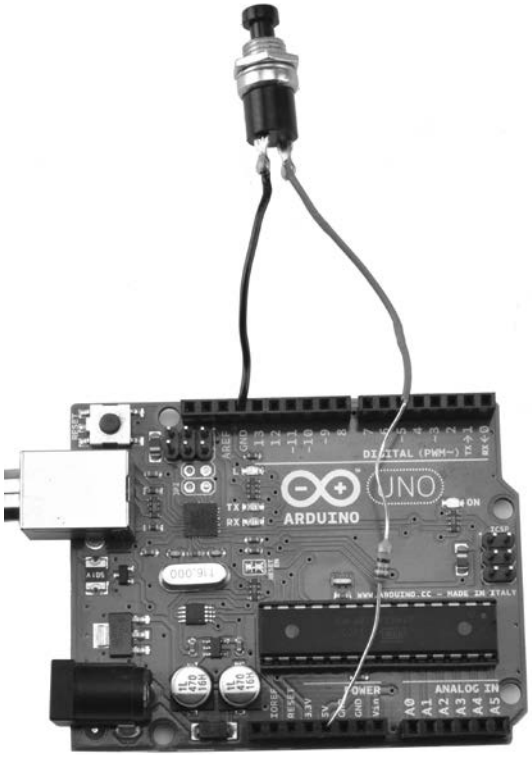

Рис. 6.7. Подключение с нагрузочным резистором

Один из побочных эффектов такой схемы подключения состоит в том, что, пока выключатель замкнут, источник напряжения 5 В оказывается соединен с «землей» через резистор, и возникает утечка тока. Поэтому сопротивление резистора должно быть довольно маленьким, чтобы гарантировать отсутствие влияния наведенных электрических помех, но достаточно высоким для того, чтобы предотвратить существенные утечки тока, когда выключатель замкнут.