10

Создание своих библиотек для Arduino

В этой главе вы:

- познакомитесь с устройством библиотек для Arduino;
- создадите простую библиотеку для повторного применения;
- узнаете, как установить свою библиотеку в Arduino IDE;
- создадите библиотеку, принимающую значения для выполнения функции;
- создадите библиотеку, обрабатывающую данные с датчика и возвращающую готовые для использования значения.

Вспомните проект 22 из главы 7, где мы установили библиотеку с функциями для сохранения данных на SD-карту. Использование библиотеки помогло сократить время на разработку скетча — нам не пришлось писать функции для выполнения операций с модулем карты.

В дальнейшем, создавая скетчи для решения ваших задач, вы будете неоднократно использовать одни и те же функции, написанные вами. Для таких случаев разумно создать свою библиотеку, которую можно было бы установить и использовать в скетчах.

В этой главе вы узнаете, как организовать функции в библиотеку. Примеры отсюда покажут вам все, что нужно знать для создания собственных библиотек. Итак, приступим.

Создание первой библиотеки для Arduino

Для начала рассмотрим листинг 10.1. Он содержит две функции, `blinkSlow()` и `blinkFast()`, которые используются для медленного и быстрого мигания встроенным светодиодом Arduino.

Листинг 10.1. Мигание встроенным светодиодом Arduino

```
void setup()
{
  pinMode(13, OUTPUT); // Использовать встроенный светодиод
}

void blinkSlow()
{
  for (int i = 0; i < 5; i++)
  {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
  }
}

void blinkFast()
{
  for (int i = 0; i < 5; i++)
  {
    digitalWrite(13, HIGH);
    delay(250);
    digitalWrite(13, LOW);
    delay(250);
  }
}

void loop()
{
  blinkSlow();
  delay(1000);
  blinkFast();
  delay(1000);
}
```

Если в скетче вам понадобятся функции `blinkSlow()` и `blinkFast()`, а библиотеки у вас нет, вам придется снова вводить их вручную. Но если поместить код функций в библиотеку, то вы сможете вызывать их, просто подключив ее в начале скетча.

Устройство библиотеки для Arduino

Библиотека для Arduino состоит из трех файлов и нескольких необязательных скетчей с примерами, показывающими использование библиотеки. Вот три необходимых файла для каждой библиотеки Arduino:

`<`*`название_библиотеки`*`>.h` — заголовочный файл;

`<`*`название_библиотеки`*`>.cpp` — файл с исходным кодом;

`KEYWORDS.TXT` — определения ключевых слов.

В первых двух именах файлов замените `<`*`название_библиотеки`*`>` фактическим именем своей библиотеки. Давайте назовем нашу первую библиотеку *blinko*. Так первые два файла получат имена `blinko.h` и `blinko.cpp`.

Заголовочный файл

Файл `blinko.h` — это *заголовочный файл*. Так он называется потому, что содержит определения (заголовки) функций, переменных и других компонентов внутри библиотеки. Заголовочный файл для библиотеки blinko показан в листинге 10.2.

Листинг 10.2. Заголовочный файл библиотеки blinko

```
  /*
❶ blinko.h – библиотека с функциями для мигания встроенным светодиодом
             на плате Arduino, подключенным к выводу D13
  */
❷ #ifndef blinko_h
  #define blinko_h
❸ #include "Arduino.h" // Открывает библиотеке доступ к стандартным типам
                       // и константам Arduino
❹ class blinko         // Функции и переменные, определяемые библиотекой
  {
    public:
      blinko();
      void slow();
      void fast();
  };
❺ #endif
```

Этот файл немного похож на типичный скетч для Arduino, но есть и некоторые отличия. В ❶ находится комментарий, описывающий назначение библиотеки. Это не обязательно, но их лучше добавлять, чтобы облегчить использование библиотеки для других.

В строке ❷ код проверяет, была ли библиотека уже подключена в главном файле скетча. В строке ❸ подключается стандартная библиотека Arduino, позволяющая библиотеке blinko использовать стандартные функции, типы и константы Arduino.

В строке ❹ объявляется класс. Его можно представлять как коллекцию, объединяющую все переменные и функции библиотеки, включая имя самой библиотеки. Внутри класса могут быть общедоступные переменные и функции для использования в скетче, подключившем библиотеку, а также частные переменные и функции, доступные только внутри класса.

У каждого класса есть *конструктор*, который используется для создания экземпляра класса. Его имя совпадает с именем класса. Такая организация может показаться сложной. Но, просмотрев примеры в этой главе и создав несколько своих библиотек, вы поймете эти конструкции.

Внутри нашего класса есть конструктор `blinko()` и две библиотечные функции: `slow()` и `fast()`. Они следуют за ключевым словом `public:`. Это значит, что их сможет использовать любой скетч, подключивший библиотеку blinko.

Наконец, в строке ❺ завершается определение заголовка. Обертывая определение заголовка директивами `#if`/`#endif`, мы гарантируем, что он не будет загружен дважды.

Файл с исходным кодом

Теперь заглянем в `blinko.cpp`. Файлы с расширением `.cpp` содержат исходный код, который будет выполняться при использовании библиотеки. Файл с исходным кодом библиотеки blinko приводится в листинге 10.3.

Листинг 10.3. Файл с исходным кодом библиотеки blinko

```
  /*
❶ blinko.cpp – библиотека с функциями для мигания встроенным светодиодом
               на плате Arduino, подключенным к выводу D13
  */
  ❷ #include "Arduino.h" // Открывает библиотеке доступ к стандартным типам
                         // и константам Arduino
  #include "blinko.h"

❸ blinko::blinko() // Выполняет операции в момент активации библиотеки
  {
    pinMode(13, OUTPUT);
  }

❹ void blinko::slow()
  {
    for (int i=0; i<5; i++)
    {
      digitalWrite(13, HIGH);
```

```
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
  }
}

❹ void blinko::fast()
{
  for (int i=0; i<5; i++)
  {
    digitalWrite(13, HIGH);
    delay(250);
    digitalWrite(13, LOW);
    delay(250);
  }
}
```

Файл с исходным кодом содержит реализации библиотечных функций, которые мы предполагаем использовать в разных скетчах. Здесь применяются и новые конструктивные элементы. В ❷ мы открываем для нашей библиотеки доступ к стандартным функциям, типам и константам Arduino и определениям в нашем заголовочном файле.

В ❸ находится определение функции-конструктора. Конструктор реализует операции, которые должны выполняться перед использованием библиотеки. В этом примере он настраивает цифровой контакт 13 на работу в качестве вывода, чтобы дать возможность управлять встроенным светодиодом Arduino.

Начиная с ❹, следуют определения функций, включенных в библиотеку. Они похожи на функции в обычном скетче с одним важным отличием: их определения начинаются с имени класса библиотеки и двух двоеточий. Например, вместо `void fast()` мы определили имя функции как `void blinko::fast()`.

Файл KEYWORDS.TXT

Теперь нужно создать файл `KEYWORDS.TXT`. Среда разработки Arduino IDE использует этот файл для определения ключевых слов в библиотеке и выделяет их в редакторе IDE. В листинге 10.4 показано содержимое файла `KEYWORDS.TXT` для нашей библиотеки blinko.

Листинг 10.4. Файл с ключевыми словами библиотеки blinko

```
blinko          KEYWORD1
slow            KEYWORD2
fast            KEYWORD2
```

Первая строка — это имя библиотеки, оно отмечено как `KEYWORD1`. Имена обеих функций обозначены как `KEYWORD2`. Обратите внимание, что пространство между

ключевыми словами и их обозначениями должно оформляться нажатием клавиши Tab, а не Пробел.

Сейчас у нас есть все три файла для создания библиотеки. Было бы неплохо включить и скетч с примерами, чтобы пользователи могли понять, как правильно применять функции. В листинге 10.5 показан наш скетч с примерами использования библиотеки blinko.

Листинг 10.5. Скетч с примерами использования библиотеки blinko

```
❶ #include <blinko.h>

❷ blinko ArduinoLED;
  void setup() {
  }

  void loop()
  {
❸   ArduinoLED.slow(); // Медленное мигание светодиодом,
                      // одна вспышка в секунду
    delay(1000);
❹   ArduinoLED.fast(); // Быстрое мигание светодиодом,
                      // четыре вспышки в секунду
    delay(1000);
}
```

Скетч очень простой. Он показывает, как использовать функции `slow()` и `fast()` в нашей библиотеке. Все, что конечному пользователю нужно сделать после установки библиотеки, — подключить ее ❶, создать экземпляр класса ❷ и вызвать любую функцию, как показано в ❸ и ❹.

Установка новой библиотеки

Теперь, чтобы поделиться новой библиотекой для Arduino с другими пользователями, нужно создать ZIP-файл. В дальнейшем с помощью этого файла они смогут установить библиотеку, как рассказывалось в главе 7.

Создание ZIP-файла в Windows версии 7 и выше

Чтобы создать ZIP-файл в Windows, следуйте инструкциям ниже.

Сначала поместите три файла библиотеки и скетч с примерами (хранящийся в отдельной папке, как и все скетчи) в одно место, как показано на рис. 10.1.

Выберите все файлы, щелкните правой кнопкой мыши на любом из них и найдите в контекстном меню пункт Send To ▸ Compressed (Zipped) Folder (Отправить ▸ Сжатая ZIP-папка), как показано на рис. 10.2.