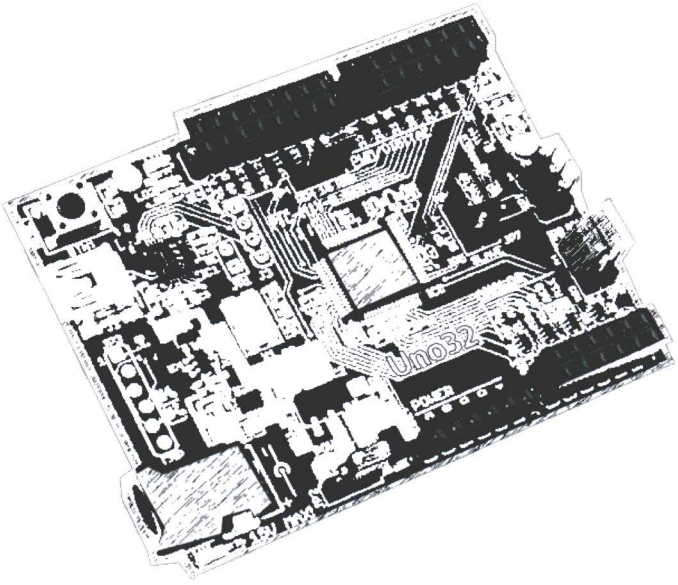
HZ BOOKS
华章科技

相比其他Arduino入门书籍，本书主要针对专业开发者。
本书不但教会读者如何使用和移植Arduino社区的开源库，还教会读者如何写自己的库。
本书结合著名的开源项目，使读者能够迅速完成从普通读者到系统开发者的升级。

电子与嵌入式系统
设计丛书



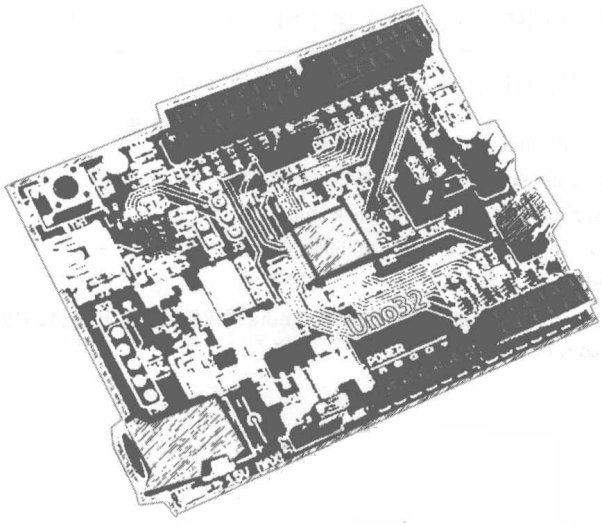
Pro Arduino

深入理解Arduino
移植和高级开发

［美］ 里克·安德森（Rick Anderson） 丹·塞尔沃（Dan Cervo） 著
程晨 译

机械工业出版社
China Machine Press

电子与嵌入式系统
设计丛书



Pro Arduino

深入理解Arduino

移植和高级开发

[美] 里克·安德森（Rick Anderson） 丹·塞尔沃（Dan Cervo） 著
程晨 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入理解 Arduino：移植和高级开发 /（美）里克·安德森（Rick Anderson），（美）丹·塞尔沃（Dan Cervo）著；程晨译 . —北京：机械工业出版社，2016.6
（电子与嵌入式系统设计丛书）
书文原名：Pro Arduino

ISBN 978-7-111-54140-0

I. 深… II. ① 里… ② 丹… ③ 程… III. 移动终端－应用程序－程序设计 IV. TN929.53

中国版本图书馆 CIP 数据核字（2016）第 144882 号

本书版权登记号：图字：01-2013-8509

深入理解 Arduino：移植和高级开发

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：陈佳媛　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2016 年 7 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：17.25
书　　号：ISBN 978-7-111-54140-0　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

前　言

Arduino 自发布以来，就已经不再只是一个开发平台，而已经成为一种文化。这种文化围绕开源和开放式硬件的理念，再造了计算机科学与教育。Arduino 开放了硬件开发，使上手更加容易，但同时又保留了真实世界应用的复杂性。这使得 Arduino 为在校学生、经验丰富的开发人员以及设计者提供了一个完美的环境。本书是一本详细介绍 Arduino 的书籍，给出了开发人员在高级设置下才能运用的技能和概念。除了项目之外，这本书还提供了例子用以演示一些概念。这些概念能够很容易地和许多其他项目集成，并为将来的项目提供灵感。本书旨在实现从中级到专业的过渡。

致谢

将最深切的感谢致以泰瑞、克雷格、道格、谢恩以及其他支持和帮助这个项目的家人和朋友。感谢克里夫·谢里尔在计算机科学方面提供了良好的基础。感谢米格尔、Ayars 博士以及 Adafruit、SparkFun 和 Arduino 的每一个人，所做出的贡献。感谢里克、米歇尔和 Apress 出版社的员工，给予我们实现这个项目的机会。

——丹·塞尔沃

首先也是最重要的，将饱含爱的感谢献给我的妻子克里斯汀·艾比。她让写作本书的过程成为我们生活的中心。其次，非常感谢合著者丹·塞尔沃。诚挚感谢所有为本书的完成提供帮助的人，尤其是瑞恩·奥斯特瑞格。我的朋友马克·斯普劳尔、安贾妮特·杨、安东尼·廖伊，以及编辑米歇尔·洛曼、布里吉德·达菲、克莉丝汀·里基茨和劳拉·乔赫斯也给予我很多支持。谢谢罗格斯大学创造了这样一个有利的环境。谢谢大卫·芬戈尔德和里奇·诺瓦克，还要感谢开源代码和开放的硬件社区，没有 Arduino，我的所有项目也将不复存在。最后，感谢 chipKIT 团队一直有求必应，并真诚地努力工作，去获得最好的开源代码支持和多平台 Arduino 图像。

——里克·安德森

目　录

第 1 章
Arduino1.0.4 的内核变化

如果你正在编写程序，创建自己的库，或制作兼容 Arduino 的电路板，你将受到 Arduino1.0.4 的变化的影响。为改善工作流程和客制化服务，人们对 Arduino IDE 做了优化。IDE 的改变包括除去未使用的按钮，将 Compile 和 Upload 按钮相邻安置。Arduino IDE 目前支持多种语言，你可以为编辑器自定义语言。这些只是可以看到的变化——有了这些升级，Arduino 团队借机实现重大的以及代码破译的改变，以便提供更加一致和完善的 Arduino API。Arduino Core API 的核心库也已被全面修订。除此之外的升级包括对自定义 Arduino 变量更好的支持，以及能够集成 Arduino 和可编程 USB 器件的功能。

本章将会一一讲述这些变化、它们有何含义以及如何影响你的代码。

这些变化可以分为以下几类：

- Arduino IDE
- 程序
- API Core
- 核心库
- Arduino 派生电路板的变量支持

1.1 Arduino IDE 的变化

Arduino 最初的文件扩展名为 .pde。这是 Processing 应用文件扩展名。如果你同时安装了这两个程序，那么 Arduino 文件将会在 Processing 程序中打开。现在，经过升级之后，Arduino 程序有了自己的扩展名：.ino。因此，mysketch.pde 现在命名为 mysketch.ino。双击文件名，Arduino 就会启动。你可以通过改变首选项用以支持原来的 PDE 扩展名，但是在默认的情况下，PDE 文件也可以简单地打开。除非你通过首选项进行设置，否则文件不会被重新命名为 .ino。

现在的 Arduino IDE 编辑窗口的左下角有了行数标号，如图 1-1 所示。Compile（编译）是第一个按钮，第二个按钮是 Upload（上传）。右下角显示了所选择的电路板及其连接的端口。这些变化使得人们能够通过识别代码行，确认正确的串行端口，并判断电路板是否连

接等信息，从而快速地调试简单的错误。

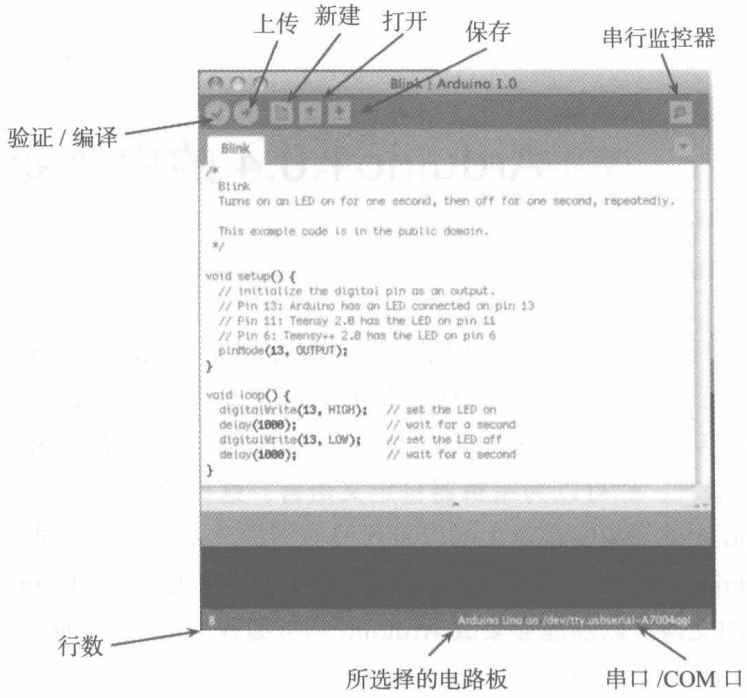


图 1-1 Arduino1.0.x 环境下更新后的主窗口

现在我们来看 Preferences 面板（File → Preferences），如图 1-2 所示。在使用编译进行查找错误的过程中，我经常使用详细的输出。verbose-output 特征之前可以通过按 Shift+Compile 组合键来触发，而现在已经被移到了 Preferences 面板。Preferences 面板现在能够调节编译输出窗口的大小，以便于阅读。

文件 preferences.txt 的位置在 Preferences 对话框里。明确这一点，因为你可能需要编辑这个文件。

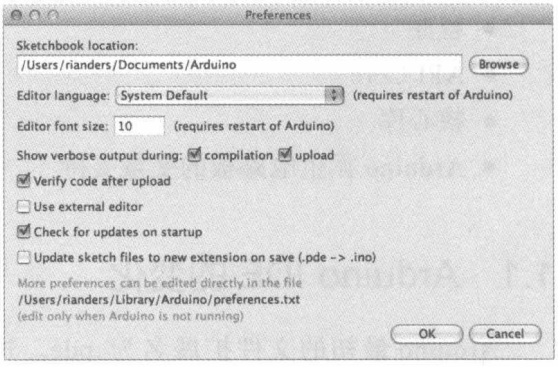


图 1-2 Arduino1.0.x 环境下更新后的 Preferences 面板

1.2 程序的变化

你编写 Arduino 程序时运用的是总可以访问的核心函数以及一系列对象的集合，不需要在程序中包含外部的库。例如，Serial 可以不用声明就直接使用。Arduino IDE 在编译之前会对 Arduino 程序进行预处理。这个过程包含了来自核心的 Arduino.h 文件。一些来自核心

的文件需要进行手动添加，就像 Ethernet 所实现的功能那样。Arduino Ethernet 电路板需要 Ethernet 的核心功能。但是因为并不是所有的 Arduino 电路板都有 Ethernet，这些文件是可用的，但不是自动包含的。

Arduino 通过程序的预处理和自动生成基本函数集而实现了简化。因此，除非需要构建自定义 Arduino 库，你不必为程序中需要包括 Arduino.h 文件并创建头文件而担心。Arduino 库需要用标准的 C/C++ 语言编写，这些将在第 12 章介绍。

这里你需要了解默认核心函数库所发生的变化。稍后本章将介绍这些变化如何影响 Arduino 的默认库文件。

上述默认库文件被具有新特征的新变量所替代，同样，WProgram.h 被 Arduino.h 所替代。

1.3　API 的升级

这部分将会讨论 API 的变化。

1.3.1　pinMode

pinMode 已经升级以支持 INPUT_PULLUP。这为创建按钮和开关增加了新的支持。这些按钮和开关在默认情况下为高电平有效，当启动时被拉低。代码清单 1-1 给出了一个例子。

代码清单 1-1　pinMode INPUT_PULLUP 电阻特性

```
setup()
{
    Serial.begin(9600);
    pinMode(10, INPUT);
    digitalWrite(10, HIGH);
    int val = digitalRead(10);
    Serial.print(val);
}
```

在 Arduino 1.0.x 环境下，你可以按以下方式编写：

```
setup()
{
    Serial.begin(9600);
    pinMode(10, INPUT_PULLUP);
    int val = digitalRead(10);
    Serial.print(val);
}
```

这种方法的好处就是使得 pinMode 能够按需要来设定默认值。此外，使用内部的上拉电阻避免了使用外部上拉电阻的需求，这样使项目得到简化。

1.3.2 返回类型

返回类型已经升级并使用 size_t 来返回数据的大小，这是一个与平台相关的无符号整数。size_t 包含在 print.h 标头 stdio.h。它返回一个用于数据打印的大小类型。

你可以使用上述返回值来检查返回用于迭代的数据量。当编写自定义库用于打印自定义数据时，你可以用 size_t 作为返回值。

1.3.3 uint_8

现在一些函数采用并返回 uint_8，这是一个具有跨平台兼容性的通用 8 位整数。

1.4 Arduino API Core 1.0.4

接下来我们看看 Arduino API Core 的变化。

1.4.1 Arduino.h

如果你正在使用标准 AVR GCC 系统库或者编写自定义库，那么了解 Arduino 库是非常重要的。Arduino.h 现包含了来自 wiring.h 的所有值。如果你已经在使用 C/C++ 编程，那么你最好了解哪个函数已经是可用的，这样你就不用再次将其添加到库中。

Arduino.h 包含了代码清单 1-2 中所示的库，所以你不用将它们再添加到你的程序中。

代码清单 1-2 Arduino.h 中自动包含的新的头文件

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "binary.h"
#include "WCharacter.h"
#include "WString.h"
#include "HardwareSerial.h"
#include "pins_arduino.h"
```

你不用将这些库复制到你的程序中。它们已自动包含在其中供你使用。

预处理过程会对 Arduino.h 编译，然后将架构与 main.cpp 文件结合。这个文件包含了 void setup () 和 void loop () 的实现。实际上，这个文件很短，如代码清单 1-3 所示。

代码清单 1-3 新版本的 main.cpp

```
#include <Arduino.h>
int main(void)
{
```

```
        init();
#if defined(USBCON)
        USBDevice.attach();
#endif
        setup();
        for (;;) {
                loop();
                if (serialEventRun) serialEventRun();
        }
        return 0;
}
```

通过观察源文件，我们可以发现两件有趣的事情。第一，main.cpp 现用来观察一个 USB 是否被定义和连接。第二，void loop() 代码运行，之后检查串口事件。如果发现事件，那么代码就会运行。

1.4.2　串行对象的升级

现在从串行口发送数据是非同步的。串行对象依赖于一个叫作 stream 的父对象。因此，它自动地包含在你主程序中的 HardwareSerial.h 中。

1.4.3　升级后的 Stream 类

Stream 类已经升级。这是串行对象的一部分，并提供 HardwareSerial 对象所使用的搜索、查询以及价值解析等功能。

1. 构造函数

构造函数（constructor）可以简单地设置串行口超时，默认值为 1000 毫秒。

```
Stream() {_timeout=1000;}
```

2. 成员函数

表 1-1 中给出了成员函数。

表 1-1　Stream 成员函数

函　数	描　述
Void setTimeout(unsigned long timeout);	为 stream 函数设置超时值。如果处理过程太长，那么它将返回。默认的时间设置为 1000 毫秒，也就是 1 秒。这个值由构造函数设置
bool find (char *target);	在 stream 中搜索目标字符，如果找到则返回 true，否则，将返回 false。此外，发生超时时也会返回 false
bool find (char *target, char, size_t length);	读取 stream 直到找到特定长度的目标字符串
bool findUntil(char *target, char * terminator);	根据与 find() 相同的逻辑工作，但是当终止符被找到时，返回 true
bool findUntil(char *target, size_t targetlen, char *terminate.size_t termlen);	在一个特定的缓冲区和长度内，如果找到终止字符串或者达到长度则返回 true

（续）

函　数	描　述
long parseInt( );	从当前位置搜索第一个有效的（长）整数。初始字符不是数字（0 到 9）或为负数将被跳过；一旦找到非数字，值将会被返回
Float parseFloat();	从当前位置搜索第一个有效的浮点类型，忽略非数字或者负号。一旦找到无周期的非数字（.），值将被返回
Size_t readBytes (char *buffer, size_t	从 stream 中读取字符串到 buffer 中，如果达到指定长度或者超时，函数要么返回 0（如果没有数据找到），要么返回 buffer 中的字符数目
length); size_t readBytesUntil(char terminator, char *buffer, size_t length);	从 stream 中读取字符串到 buffer 中，如果读到终止字符串、到达指定长度，或者达到超时，那么函数将返回 0（没有找到数据）或者 buffer 中的字符数目。
long parseInt(char skipChar);	允许跳过整数的分解和字符（例如，逗号）
float parseFloat (char skipChar);	功能与 parseFloat() 相似，但是忽略跳过字符

1.4.4 Print 类

Print 类也已经升级。这直接影响 Client 和 Stream 类。包含这些的类别也会受到影响。HardwareSerial 和 UDP 类使用 Stream。因此，你不用特意将 Print 包含到你的主 Arduino 程序中。表 1-2 给出了一些更重要的升级的公共方法。

表 1-2　升级的 Print 公共方法

方　法	描　述
size_t write(const char *str) {return write((const uint8_t *)str, strlen(str));}	打印位于指针位置的字符串。该函数自动找到字符串的长度，返回打印的字符数目
virtual size_t write(const uint8_t *buffer, size_t size);	将一个 uint8_t 常量指针写入大小为 size_t 的缓冲区中。打印一定长度的字节，返回打印字符的数目
size_t print(const_FlashStringHelper *);	打印闪存中的常量字符串，返回打印的字符数目
size_t print(const String &);	打印通过的常量字符串对象作为参考，返回打印字符的数目
size_t print(const char[ ]);	打印常量字符阵列，返回打印字符的数目
size_t print(char);	打印字符，返回打印字符的数目
size_t print(unsigned char, int=DEC);	以十进制形式打印无符号字符，返回打印字符的数目
size_t print(int, int=DEC);	以默认十进制形式打印整型，返回打印字符的数目
size_t print(unsigned int, int=DEC);	以默认十进制形式打印无符号整型，返回打印字符的数目
size_t print(long, int=DEC);	以默认的十进制形式打印长整型，返回打印字符的数目
size_t print(unsigned long, int=DEC);	以默认十进制形式打印无符号整型，返回打印字符的数目
size_t print(double, int=2);	打印双精度型，取小数点后两位，返回打印字符的数目
size_t print(const Printabble&)	打印通过的可打印的对象作为参考，返回打印字符的数目
size_t println(const_FlashStringHelper *);	换一行打印闪存中的常量字符串，返回打印字符的数目
size_t println(const String &s);	换一行打印通过的常量字符串作为参考，返回打印字符的数目
size_t println(char[ ]);	换一行打印常量字符阵列，返回打印字符的数目

（续）

方　法	描　述
size_t pirntln(char);	换一行打印字符，返回打印字符的数目
size_t println(unsigned char, int=DEC);	换一行以默认的十进制形式打印无符号字符，返回打印字符的数目
size_t println(int, int=DEC);	换一行以默认的十进制形式打印整型，返回打印字符的数目
size_t println(unsigned int, int=DEC);	换一行以默认的十进制形式打印无符号整型，返回打印字符的数目
size_t println(long,int=DEC);	换一行以十进制打印长整型，返回打印字符的数目
size_t println(unsigned long, int=DEC);	换一行以十进制打印无符号长整型，返回打印字符的数目。
size_t println(double, int=2);	换一行打印双精度型，取小数点后两位，返回打印字符的数目
size_t println(const Printable&);	给定一个可打印的对象，换一行打印
size_t println(void);	打印一个换行符。返回打印字符的数目

1.4.5　新型的可打印类

创建一种新型的可打印类，用来定义新的对象如何被打印。代码清单 1-4 给出了一个例子。

代码清单 1-4　写入字节的例子

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

void loop( )
{
  byte bb = B101101;
  int bytesSent = Serial.print("byte: println: ");
  Serial.print(bytesSent);
  Serial.print(" : ");
  Serial.print(bb);
  Serial.print(" write: ");
  Serial.write(bb);
  Serial.print("");
  Serial.write(45); // send a byte with the value 45
  Serial.println("");
  bytesSent = Serial.write("A");
}
```

1.4.6　字符串库的升级

通过 F() 命令可以很容易地实现存储字符串到闪存中并打印。无论是什么字符串，只要是放在引号里面，都会被存储到闪存中，这将会减少 RAM 的使用量。

```
Serial.println(F("store in Flash"));
```

1.4.7 有线库的升级

有线库（Wire Library）也使用 Stream，因此有着和 Serial 一样的特征。函数 Wire.send() 被 Wire.write() 替换，Wire.receive ( ) 也换成了 Wire.read( )。

1.4.8 硬件串口的升级

硬件串口现在默认支持 USB。

- Serial.begin() 支持无符号长整型的声明。
- Serial.write() 现在返回 size_t。
- Serial.SerialEvent() 也发生了变化。
- Serial.SerialEventRun() 实现检查多达 4 个定义的串行口 (Serial、Serial1、Serial2 和 Serial3)，并查找每个可用的串口数据。

1.5 物理电路板的升级和 USB 的兼容性

所有新的 Arduino 电路板都有 16u2 芯片，用于 USB 或者拥有内置的 USB 支持，就像 Arduino Leonardo 32u4 一样。核心部分包括 USB 串口、键盘和手柄。Arduino Leonardo 拥有一些优势，比如在 Arduino 程序中 USB 库是可以访问的，可以用新的 USB 库对 Arduino Leonardo 进行编程。但是，16u2 芯片不能使用相同的库。而且因为它们是独立的芯片，对其编程也必须独立进行。目前，开发最为广泛的 USB 支持库是来自 Paul Stoffregen 的 Teensy 和 Teensy++ 电路板。

1.5.1 Avrdude 的升级

Avrdude 是 Arduino 用来将程序图像上传到 Arduino 电路板的上传器。Arduino 带有的 Avrdude 版本已经升级到 5.11，以支持上传类型。过去是 stk500 上传类型。从版本 5.11 开始，所有官方的电路板都可通过这个 Arduino 上传类型进行编程。此外，自定义启动加载程序和固件也可以通过 Avrdude 下载到 Arduino 上。

你也可以利用这个特性使用 Arduino 启动加载程序来对微控制器编程，这样微控制器就可以运行 Arduino 程序。可编程的微控制器包括 ATtiny85、ATtiny45、chipKIT Uno32、chipKIT Fubarino SD，以及用户创造和设计的可兼容 Arduino 的微控制器。

1.5.2 新的 Arduino Leonardo 电路板

Arduino 修订版 3 已经完成配置和升级，定义了变量类型，上传类型被配置为 Arduino。Arduino Leonardo 基于 Atmel ATmega32u4 芯片。Leonardo 电路板有以下特性：