第 3 章

Arduino 程序设计基础

3.1 Arduino 编程简介

Arduino 程序非常简单易用，尤其是对于有 C 语言编程基础的使用者。本章将介绍 Arduino 程序的语法和常用功能。打开 Arduino IDE，将自动给每个新项目的程序文件中添加两个函数：setup()和 loop()。

Arduino 程序不需要写 main()函数，但必须有 setup()和 loop()函数，这就是 Arduino 程序最基本的结构。用户关注 setup()和 loop()函数的编程设计即可，Arduino IDE 内部会自动生成 main()函数，具体细节可以参考第 2 章中的实战内容。

3.2 Arduino 常用数据类型

Arduino 常用数据类型如表 3.1 所示。

表 3.1 Arduino 常用数据类型

数据类型	字节数	取值范围	存储类型
int	2	−32 768～32 767	正或负的整数
unsigned int	2	0～65 535	正的整数
long	4	−2 147 483 648 ～2 147 483 648	大范围的整数
unsigned long	4	0～4 294 967 295	大范围的正整数
float	4	−3.402 823 5E+38 ～3.402 823 5E+38	浮点数

续表

数据类型	字节数	取值范围	存储类型
double	4	−3.402 823 5E+38 ～3.402 823 5E+38	在 ATmega 系列的 8 位微处理器中，double 和 float 都是 4 字节
Boolean	1	true，false	只能是 true 或 false
char	1	−128～127	通常用来表示字符，也可以表示范围内的整数
Byte	1	0～255	小范围的正整数

与 C 语言相似，变量类型的选择与实际要求相关，在定义一个变量时就必须声明这个变量的类型，如

```
float x = 0.1;      //声明一个值为 0.1，变量名为 x 的浮点型变量
```

3.2.1 浮点数精度的问题

需要注意的是，Arduino 在计算、存储浮点数时，会产生精度误差。接下来用一个例子说明。

首先，在 setup()函数中声明一个浮点数。

```
void setup() {
   Serial.begin(9600);                    //初始化串口，波特率为 9600
   float x = 0.1;
}
void loop(){
   x = x - 0.1;
   if(x == 0){                            //如果 x 等于 0
        Serial.println("x 与 0 相等");     //打印语句
   }else {                                //否则，如果 x 小于 0
        Serial.println("x 不等于 0");      //打印另一个语句
   }
}
```

运行上述程序，串口监视器将会持续输出“x 不等于 0”。一般而言，大家都会觉得程序输出应该是第一句“x 与 0 相等”，但输出却是“x 不等于 0”。这是为什么呢？原因是，程序里写的十进制小数在计算机内部只能用二进制的小数近似表示，所以无法精确地表征。对于二进制小数，小数点右边能表达的值是 1/2，1/4，1/8，1/16，1/32，1/64，1/128，…，$1/2^n$。所有这些小数都是一点一点地拼凑出来的一个近似的数值，所以才会有不准确的现象发生。

由上可知，如果需要对浮点型的数字进行大小判断，不能直接使用==、>或<，而要使用相对差值，可以用一个程序实现浮点数的比较。

```
boolean almostEqual(float a, float b){
    const float DELTA = .00001; //如果两个值相差小于此值，视为相等
    if(a == 0) return fabs(b) <= DELTA;
    if(b == 0) return fabs(a) <= DELTA;
    return fabs((a - b) / max(fabs(a), fabs(b))) <= DELTA;
}
```

3.2.2　变量作用域

在 Arduino 中声明的变量，有一个名为作用域(Scope)的属性。简单来说，在某个函数内部声明的变量，只在其函数内部可用。这样，当程序规模很大时，可以防止很多难以预料的错误。

例外的是，在 for 循环的括号中声明并初始化的变量，只在 for 循环中可用，举例如下。

```
int x;              // 任何函数都可以调用此变量
void setup(){
}
void loop(){
   int i;           // i 只在 loop()函数内可用
   float f;         // f 只在 loop()函数内可用
   // ...
   for (int j = 0; j<100; j++){
    //变量 j 只能在循环括号内访问
   }
}
```

3.3　Arduino 中的运算

Arduino 提供了一套丰富的运算编程功能。本书将根据用途的不同，将运算符分组进行介绍。

3.3.1　算术运算符

本节介绍 Arduino 中的一些常用算术运算符，如表 3.2 所示。实例假设整数变量 A 的值为 10，变量 B 的值为 20。

表 3.2 Arduino 常用算术运算符

运 算 符	描 述	例 子
+	加法	A+B=30
−	减法	A−B=−10
*	乘法	A * B=200
/	除法	B / A=2
%	取余,左操作数除以右操作数的余数	B%A =0
abs(x)	取绝对值	abs(−10) =10
min(x,y)	取较小值	min(A,B)=10
max(x,y)	取较大值	max(A,B)=20
pow(x,y)	计算 x 的 y 次方	pow(3,2)= 9
sqrt(x)	计算 x 的平方根	sqrt(4)= 2

3.3.2 关系运算符

表 3.3 所示为 Arduino 支持的常用关系运算符。实例假设整数变量 A 的值为 10,变量 B 的值为 20。

表 3.3 Arduino 常用关系运算符

运 算 符	描 述	例 子
==	等于	(A==B)为假(非真)
!=	不等于	(A!=B) 为真
>	大于	(A>B)为假
<	小于	(A<B)为真
>=	大于或等于	(A>=B)为假
<=	小于或等于	(A<=B)为真

3.3.3 位运算符

Arduino 定义了位运算符,位运算符作用在所有位上,并且按位运算。

假设变量 A=60,B=13,它们的二进制格式表示如下。

```
A = 0011 1100
B = 0000 1101
```

表 3.4 列出了 Arduino 常用位运算符。

表 3.4　Arduino 常用位运算符

运　算　符	描　　述	例　　子
&	按位与	(A&B) 得到 12，即 0000 1100
|	按位或	(A|B) 得到 61，即 0011 1101
^	按位异或	(A^B)得到 49，即 0011 0001
~	取反	(~A)得到－61，即 1100 0011
<<	按位左移	A << 2 得到 240，即 1111 0000
>>	按位右移	A >> 2 得到 15，即 0000 1111

3.3.4　逻辑运算符

表 3.5 列出了 Arduino 常用逻辑运算符，其中，假设布尔变量 A 为 true，变量 B 为 false。

表 3.5　Arduino 常用逻辑运算符

运　算　符	描　　述	例　　子
&&	逻辑与	(A&&B)为 false
||	逻辑或	(A||B)为 true
!	非	!A 为 false

3.3.5　赋值运算符

表 3.6 列出了 Arduino 常用赋值运算符。

表 3.6　Arduino 常用赋值运算符

运　算　符	描　　述	例　　子
+=	加和赋值	C+=A 等价于 C=C+A
-=	减和赋值	C-=A 等价于 C=C-A
*=	乘和赋值	C*=A 等价于 C=C*A
/=	除和赋值	C/=A 等价于 C=C/A
<<=	左移位赋值	C <<=2 等价于 C=C << 2
>>=	右移位赋值	C >>=2 等价于 C=C >> 2
&=	按位与赋值	C&=2 等价于 C=C&2
|=	按位或赋值	C|=2 等价于 C=C|2

3.3.6　限制取值范围

使用 constrain(x,min,max)函数可以确保返回值在 min～max 的范围内。

很多时候需要一些值必须被限制在某些范围内，如电压值过大的话会烧毁元件，过

小的话不能保证元件正常工作。在这种情况下就可以用 constrain()函数获取一个范围内的值。

令 min=200,max=220,则在不同的 x 值情况下,constrain(x,min,max) 的返回值的变化如表 3.7 所示。

表 3.7 constrain()函数举例

x	constrain(x,min,max) 的返回值
190	200
210	210
230	220

3.3.7 取整

对一个浮点数向上或向下取整是一个常用的操作。使用 floor(x)获取不大于 x 的最大整数;使用 ceil(x)获取不小于 x 的最小整数。

表 3.8 所示为在不同的 x 值情况下,取整函数举例。

表 3.8 取整函数举例

x	floor(x)的返回值	ceil(x)的返回值
1	1.00	1.00
1.5	1.00	2.00
−1.5	−2.00	−1.00

3.3.8 三角函数

为适应实际需要,Arduino 提供了丰富的三角函数功能。

```
double sin(double x);
double cos(double y);
double tan(double x);
double acos(double x);
double asin(double x);
double atan(double x);
```

需要注意的是,三角函数的默认输入变量为弧度(rad)。如果输入变量是用角度表示的,务必转换为弧度单位。

3.3.9　随机数

Arduino 提供了方便的函数实现随机数的获取。

```
random(min, max);      //返回范围为[min, max -1]的一个随机数
random(max);           //当 min 省略时默认为 0,返回范围为[0,max-1]的一个随机数
```

3.4　Arduino 的循环语句

顺序结构的程序语句只能被执行一次。如果想要执行多次同样的操作,就需要使用循环结构。下面将分别介绍 Arduino 中主要的循环结构。

3.4.1　while 循环

while 是最基本的循环,它的结构为

```
while(布尔表达式) {
  //循环内容
}
```

只要布尔表达式为 true,循环就会一直执行下去。例如,下列程序会循环 100 次。

```
int x = 100;                //设置循环次数(全局变量)
void setup() {
    Serial.begin(9600);     //初始化串口,波特率为 9600
}
void loop(){
    while( x > 0 ){
      //需要循环的语句
      x = x - 1;
    }
}
```

3.4.2　do…while 循环

对于 while 语句,如果不满足条件,则不能进入循环。但有时程序需要即使不满足条件,也至少执行一次。

do…while 循环和 while 循环相似，不同的是，do…while 循环至少会执行一次，其结构为

```
do {
        //代码语句
}while(布尔表达式);
```

举例如下。

```
do
{
    delay(1000);        //等待温度传感器稳定
    x = readTemp();     //获取温度传感器的值,readTemp()为自定义的函数

} while (x < 100);      //温度小于 100 则重复上述语句,大于 100 则执行后续语句
```

3.4.3 for 循环

虽然所有循环结构都可以用 while 或 do…while 表示，但 Arduino 提供了另一种语句——for 循环，使一些循环结构变得更加简单可控。

for 循环执行的次数是在执行前就确定的。语法格式如下。

```
for(初始化; 布尔表达式; 更新) {
    //代码语句
}
```

Arduino 的 for 循环与 C 语言完全相同，如

```
for(int x = 0; x < 10; x = x + 1) {
    //需要执行 10 次的语句
}
```

当 for 循环直接写在 loop()函数中时，会不断触发。

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

```
    for(i = 0;i < 10;i++)
    {
      Serial.println(i);
    }
}
```

输出结果如图 3.1 所示。

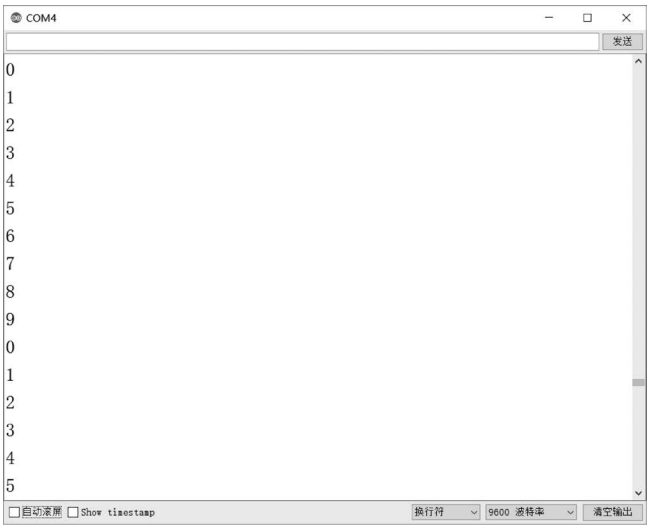

图 3.1　for 循环输出结果

3.4.4　循环的跳出

很多时候程序需要在满足一定条件下跳出循环，与 C 语言相似，Arduino 提供了 break 关键字。break 可以使程序跳出最里层的循环，并且继续执行该循环下面的语句。

```
for(int x = 0; x < 10; x = x + 1) {
    //x = 5 则跳出循环
    if( x == 5 ){
        break;
    }
}
```

除了 break 之外，Arduino 还提供 continue 关键字。

continue 适用于任何循环控制结构中，作用是让程序立刻跳转到下一次循环的迭代。

在 for 循环中，continue 关键字使程序立即跳转到更新语句。

在 while 或 do…while 循环中，程序立即跳转到布尔表达式的判断语句。

```
for(int x = 0; x < 3; x = x + 1) {
    //x = 1 则立刻跳转到下一次循环，不进行之后的语句处理
    if( x == 1 ){
        continue;
    }
    Serial.println(x);
}
```

串口处将输出

```
0
2
```

3.5 Arduino 的条件判断语句

3.5.1 if 语句

一个 if 语句包含一个布尔表达式和一条或多条语句。如果布尔表达式的值为 true，则执行 if 语句后的代码块，否则跳过 if 语句后的代码块。

```
int x = 1;
if( x == 1 ){     //x 值为 1 则执行花括号中的语句
    Serial.println("x 的值为 1");
}
else{             //否则执行 else 后的代码块
    Serial.println("x 的值不为 1");
}
Serial.println("程序结束");
```

串口处将输出

```
x 的值为 1
程序结束
```

3.5.2 switch case 语句

switch case 语句判断一个变量与一系列值中某个值是否相等，每个值称为一个分

支。尤其需要注意，不要遗忘每个 case 之后的 break 关键字，否则程序会继续执行后面的 case 语句和 default 关键字后面的语句。示例如下。

```
switch (value) {
    case 1:
      //当 value 等于 1 时执行此处语句
      break;
    case 2:
      //当 value 等于 2 时执行此处语句
      break;
    default:
      // 如果没有匹配项或 break，则执行此段
      // default 段是可选的
}
```

3.6　Arduino 数组与字符串

3.6.1　Arduino 中的数组

Arduino 中的数组与 C 语言基本相同，是一组变量的集合。数组的声明如下。

```
int list1[] = {1, 2, 3, 4};   //声明整型数组的同时初始化每个元素
float list2[5];               //声明一个长度为 5 的浮点型数组，但不初始化元素
Int list1[5] = {1,3,5};       //声明整型数组同时初始化前 3 个元素，未初始化的元素默认为 0
```

数组的索引是从 0 开始的，即

```
list[0] = 1;
list[3] = 4;
```

使用数组时一定要注意不能超出数组的索引范围，否则会出现无法预料的错误，而且编译器不会报警。

3.6.2　Arduino 中的字符串

Arduino 中最常用的数据类型之一是字符串，用来存储字符。与数组不同，字符串由一系列字符和用来表示结束的空字符组成。尤其注意分配空间和操纵字符串时不要忘记末尾的空字符。字符串的声明如下，末尾空字符不占空间，也不算在字符长度中。

```
char string0[6];              //声明长度为 6 的字符串
char string1[5] = "Hello";    //声明并初始化长度为 5 的字符串
char string2[12] = "Hello"; /* 声明一个长度为 12 的字符串数组,并初始化前 5 位,之后 7 位
                               没有初始化 */
char string3[] = "Hello";     //由编译器初始化字符串并确定大小
```

3.6.3 字符串的常用操作

Arduino 预置了一些函数方便用户进行字符串的常用操作。字符串的常用操作如表 3.9 所示,示例中的字符串为 3.6.2 节声明的字符串。

表 3.9 字符串的常用操作

函数形式	功能	示例
strlen(str)	获取字符串 str 的长度	strlen(string2); 结果为 12
strncpy(y,x)	将字符串 x 复制到 y 中	strncpy(string0,string1); 将 string1 复制到 string0 中,string0 现在为 Hello
strncpy(y,x,n)	将字符串 x 的前 n 个字符复制到 y 中	strncpy(string0,string2,5); 将 string2 的前 5 个字符复制到 string0 中,string0 现在为 Hello
strncpy(y,x)	将字符串 x 添加到 y 的末尾	strncpy(string2,string1); 将 string1 添加到 string2 的末尾,string2 现在为 HelloHello
strcmp(y,x)	比较字符串 x 和 y,如果相等就返回 0,如果不相等则返回第 1 个不同字母的 ASCII 码差值	strcmp(string1,“Hello”); 结果为 0 strcmp(string1,“hello”); 结果为-32 strcmp(string1,“hfllo”); 结果仍然为-32

3.7 String 及其成员函数

除了简单的字符串数组外,Arduino 还提供了类似于 C++语言中的 String 字符串对象,实现更加复杂的字符串操作。注意 String 对象的首字母是大写的 S,与 string 字符串不同。

String 对象的声明如下。

```
String String1 = "First String";        //声明一个 String 对象并初始化
String String2 = "Second String";       //声明一个 String 对象并初始化
String String3;                         //声明一个 String 对象,等待语句赋值
```

String 对象内置功能丰富的函数，如长度、比较等。其调用形式为“字符串对象名.函数名”。用刚刚定义的 String 作为例子：

```
int len = string1.length();             //length()为获取长度函数,len 的值为 12
```

String 的主要成员函数如表 3.10 所示。

表 3.10　String 的主要成员函数

函　数　名	功　　能
charAt(n)	返回 String 中的第 n 个字符
compareTo(String2)	与 String2 比较
concat(String2)	返回 String 与 String2 组合后的 String
endsWith(String2)	检查 String 是否以 String2 结尾
equals(String2)	检查 String 是否与 String2 相同
equalsIgnoreCase(String2)	检查 String 是否与 String2 相同(不区分大小写)
GetBytes(buffer,a)	复制字符 a 到提供的字节缓冲区
indexOf(a)	如若包含字符 a,则返回索引,否则返回－1
lastIndexOf(a)	与 indexOf()函数功能一样,但返回索引从结尾开始
length()	返回在 String 中的字符数量
replace(A,B)	将 String 中的所有实例 A 替换为 B
setCharAt(index,c)	存储字符 c 到 String 中给定的索引位置
startsWith(String2)	如果 String 以 String2 开头,则返回 true
substring(start,end)	返回 String 中从 start 开始到 end 前一个字符的字符串
toCharArray(buffer,len)	复制 String 中的 len 个字符到提供的缓冲区
toLowerCase()	所有字符都转化为小写
toUpperCase()	所有字符都转化为大写
trim()	去除开头和结尾的空格

特别地，使用＋运算符，可以实现与字符串连接函数 concat()相似的功能。

```
String String4 = String1 + String2;     //String4 将变为 First stringSecond string
```

String 对象虽然内置了功能丰富的函数，但会比 string 字符串消耗更多的资源。当

使用数量巨大的字符串时，应当使用 string 字符串，并小心操作确保不会越界。

3.8 Arduino 中的函数

函数是一组一起执行一个任务的语句。在日常编程中，代码经常划分到不同的函数中，增强程序的规范性、可读性和可重复利用性。

函数声明即告诉编译器函数的名称、返回类型和参数。函数定义提供了函数的实际主体。如表 3.11 所示，Arduino 中的函数定义的一般形式如下。

```
return_type function_name( parameter list )
{
   函数体
}
```

表 3.11 函数定义中各部分的解释

名　　称	中文名	解　　释
return_type	返回类型	一个函数可以返回一个值。return_type 是函数返回值的数据类型，如 int、float 等。有些函数执行所需的操作而不返回值，在这种情况下，return_type 是关键字 void
function_name	函数名称	函数的实际名称。函数名和参数列表一起构成了函数签名
parameter list	参数	参数就像是占位符，当函数被调用时，用户可以向参数传递一个值，这个值被称为实际参数。参数列表包括函数参数的类型、顺序、数量。参数是可选的，也就是说函数也可能不包含参数

下面给出一个函数的例子。

```
// 返回两个数中较大的那个数
int maxnumber(int num1, int num2)
{
   //局部变量声明
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
```

当调用这个函数时，就可以使用如下代码。

```
int num1 = 10;
int num2 = 20;
int max_one = maxnumber( num1, num2);
```

max_one 的值就是输入参数中的较大值 num2，即 20。

本章小结

本章通过一些实例介绍了 Arduino 编程的基础知识，包括常用的数据类型、运算符、循环语句、条件判断语句、数组与字符串、String 类型、函数等。后续的章节将应用这些编程技术，结合硬件资源实现形态各异的电子系统设计。

感兴趣的读者可以搜索资料，完成以下拓展练习，深入掌握 Arduino 编程的技巧。

拓展练习

（1）请列举 Arduino 编程与 C 语言编程的相似和不同之处。

（2）请列举 Arduino 编程与 C++语言编程的相似和不同之处。

（3）请列举 Arduino 编程与 Python 语言编程的相似和不同之处。

（4）请搜索资料，总结 Arduino 编程中指针的用法，并设计编程应用实例，用指针编程技术实现。