JIYU C YUYAN DE WULIANWANG KAIFA JISHU

基于C语言的物联网开发技术

高 松 主编

北京邮电大学出版社
www.buptpress.com

高松，女，籍贯为山东省菏泽市。2001 年毕业于聊城大学物理系应用电子技术教育专业，获取学士学位。2002—2005 年在中国矿业大学攻读硕士研究生，获取硕士学位。2005—2008 年在中国矿业大学攻读博士研究生，获取博士学位，所学专业为控制理论与控制工程。2008 年 7 月至今，在北京政法职业学院信息技术系网络安全教研室担任专职教师，主要负责计算机网络技术专业网络组建与网络攻防方向、物联网方向专业课程的教学工作。已发表论文十几篇，主编或参编教材 4 本，参与市级课题 4 个，主持或参与校级课题 8 个。多次指导学生参加职业技能大赛，并帮助学生获得了北京市大赛的二等奖、三等奖和国赛二等奖，其中，2014 年指导学生在移动互联技术应用赛项中获得北京市二等奖，2015 年指导学生在计算机网络应用赛项中获得北京市二等奖，2018 年指导学生在计算机网络应用赛项中获得北京市三等奖、国赛二等奖。

JIYU C YUYAN DE WULIANWANG KAIFA JISHU

基于C语言的物联网开发技术

ISBN 978-7-5635-5870-4

9 787563 558704

定价: 33.00元

策 划 人：张向杰
责任编辑：徐振华　王小莹
封面设计：七星博纳

基于C语言的物联网开发技术

高　松　主编

北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书主要以C语言为基础，结合硬件设备CC2530开发板，介绍了物联网开发技术，主要内容包括C语言的数据类型、顺序结构程序设计、选择结构程序设计、循环结构程序设计、函数、数组等内容。本书将理论内容与实践相结合，为每章内容设计了一个实际项目，让学生在学习C语言程序设计理论知识的同时，能够将所学知识应用到物联网硬件设备中，如控制LED灯的亮灭、利用按键控制两灯、使用循环控制两灯亮灭的间隔时间、控制输出字符串的内容、求解条形码的校验码等。本书在保证了完整的C语言知识体系的基础上，用大量的例题帮助学生掌握程序设计的思想和学会程序设计的方法，并配有大量和实际相关的实践开发项目。同时，本书与物联网硬件设备相结合，将程序设计应用到实际中，注重原理与实践相结合，实用性强。

图书在版编目(CIP)数据

基于C语言的物联网开发技术 / 高松主编. -- 北京 : 北京邮电大学出版社，2019.9
ISBN 978-7-5635-5870-4

Ⅰ. ①基… Ⅱ. ①高… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2019)第195363号

书　　名：基于C语言的物联网开发技术
作　　者：高　松
责任编辑：徐振华　王小莹
出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路10号(邮编：100876)
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：北京玺诚印务有限公司
开　　本：787 mm×1 092 mm　1/16
印　　张：11
字　　数：255千字
版　　次：2019年9月第1版　2019年9月第1次印刷

ISBN 978-7-5635-5870-4　　定　价：33.00元

前　言

本书针对目前市面上适合高职学生的物联网开发技术教材短缺的情况，以物联网的开发软件C语言为主，结合硬件CC2530模块的实际应用开发项目，详细地介绍了基于C语言的物联网开发技术，并将每部分理论内容与实践相结合，让学生在学习理论的同时能够进行开发项目的实践。本书具体内容如下：

第1章是顺序结构程序设计，主要内容为数据类型、运算符、表达式和数据的输入输出、字符数据的输入与输出等，并结合了第6章的项目一“编程实现开发板上灯的点亮”，让学生熟悉模块程序的烧写。

第2章是选择结构程序设计，主要内容为if语句、if语句的嵌套、switch语句等，并结合了第6章的项目二“编程实现开发板上两个灯的亮灭”，让学生熟悉使用选择语句通过模块上的两键控制两灯的亮灭。

第3章是循环结构程序设计，主要内容为循环结构的概念、while语句、do-while语句、for语句，并结合了第6章的项目三“编程实现通过按键控制两灯的亮灭”，让学生熟悉使用循环来控制灯闪亮的时间间隔。

第4章是数组，主要内容为一维数组、字符数组、二维数组，并结合了第6章的项目四“编程实现通过指令控制两个灯的亮灭”，让学生熟悉使用数组的元素值作为指令控制灯的亮灭的方法。

第5章是函数，主要内容为函数的定义、函数调用、函数的嵌套调用、函数的递归调用等，并结合了第6章的项目五“编程实现四个灯的流水灯效果”，让学生熟悉用循环和调用延时函数来控制开发板上四个灯的依次亮灭。

第6章的项目六是针对应用广泛的条形码的程序设计，以帮助学生对程序开发进行综合实践。

在每个项目后还有项目拓展，让读者通过一个项目的开发，由易到难、不断深入地完成项目拓展。

书中每章在介绍程序开发理论的同时，结合大量的例题和非常详细的程序分析，内容充实易懂，言简意赅。并且，每章后都设置了大量各种类型的习题，以供学生巩固复习。

本书适合高职高专院校物联网专业及计算机相关专业的学生使用。本书用大量与生活息息相关的实例帮助读者理解程序开发的思想，学会程序设计的方法。并且，本书与物联网硬件设备相结合，将程序设计应用到实际系统开发中，注重理论与实践相结合，实用性强。

高　松
北京政法职业学院

目　录

第1章　顺序结构程序设计

1.1　数据类型

在C语言中，数据类型可以分为基本类型、构造类型、指针类型和空类型，其中，基本类型可以分为整型、实型、字符型，构造类型包括数组、结构体、枚举类型。整型可以分为短整型(short)、长整型(long)、整型(int)，实型可分为单精度型(float)、双精度型(double)、字符型(char)。

C语言有32个关键字，包括auto、break、case、char、const、continue、default、do、double、else、enum、extern、float、for、goto、if、int、long、register、return、short、signed、sizeof、static、struct、switch、typedef、unsigned、union、void、volatile、while。每个关键字都有特定的含义，例如，int用于声明整型变量或函数；float用于声明浮点型变量或函数；short用于声明短整型变量或函数；char用于声明字符型变量或函数返回值类型；long用于声明长整型变量或函数。

下面将详细介绍常量和变量。

常量是指在程序执行过程中，其值不可改变的量。例如，整型常量包括10、3、－5等；实型常量包括2.5、－6.3、0.15等；字符常量包括'1'、'k'、'*'等；字符串常量包括"987"、"hijst"、"I am a student"等。

除上述常量类型外，还有一种常量——符号常量。若在程序中用一个标识符来表示一个常量，这个标识符就是符号常量。符号常量名通常用大写字母或大写字母组合来表示。一般用下面的预处理命令的格式来定义符号常量：

```
#define 标识符(符号常量) 常量
```

一旦定义了符号常量，在整个程序执行过程中，该符号常量的值都保持不变。例如，指令#define A 100定义了A为符号常量，其值为100，因此，在程序运行中，A就表示常量100。

变量是指在程序运行期间其值可以改变的量，通常以标识符来命名。C语言规定标识符是由字母、数字或下划线三种字符组成，并且只能以字母或下划线开头，即起始字符不能为数字。在程序编译过程中，系统会为变量分配相应存储单元。例如，_a、cat、Year、

name_1 均为合法的变量名，而 4ab、a. b、%a、@、x<y 均为不合法变量名。因为，4ab 是数字开头的，a. b 包含了不合法的字符“.”，%a 包含了不合法的字符“%”，“@”不是三种合法字符中的字符，x<y 中包含了不合法的字符“<”。在定义变量时，不仅要使用字母、数字、下划线三种合法组成字符来定义，还要注意大小写字母是两个不同的标识符。

1. 整型常量和变量

(1) 整型常量

整型可以分为短整型(short)、长整形(long)、整型(int)。在C语言中，整型常量按进制可以分为十进制整型常量、八进制整型常量、十六进制整型常量。十进制整型常量包括 10、980 等；八进制整型常量以 0 开头，如 032、0671、－055 等；十六进制整型常量以 0x 开头，如 0x32、0x671、－0x55 等。

例 1-1 将下面的八进制数转换成十进制数：

032,0671,－055

分析：

032 表示八进制数 32，转换成十进制数为：

$$3\times8^1+2\times8^0=26$$

0671 表示八进制数 671，转换成十进制数为：

$$6\times8^2+7\times8^1+1\times8^0=6\times64+7\times8+1=384+56+1=441$$

－055 表示八进制数－55，转换成十进制数为：

$$-(5\times8^1+5\times8^0)=-45$$

例 1-2 将下面的十六进制数转换成十进制数：

0x32,0x671,－0x55

分析：

0x32 表示十六进制数 32，转换成十进制数为：

$$3\times16^1+2\times16^0=3\times16+2\times1=50$$

0x671 表示十六进制数 671，转换成十进制数为：

$$6\times16^2+7\times16^1+1\times16^0=6\times256+7\times16+1=1536+112+1=1649$$

－0x55 表示十六进制数－55，转换成十进制数为：

$$-(5\times16^1+5\times16^0)=-(80+5)=-85$$

(2) 整型变量

在不同的编译系统中对程序进行编译时，对于整型变量分配的存储单元大小是不同的，一般是分配 4 字节的存储单元，即 32 位大小的存储单元。例如，对于程序

```
int a = 1,b = 2,c = 3;
```

程序编译过程中，系统会分配 4 字节的存储单元给变量 a，再分配 4 字节的存储单元给变量 b，最后分配 4 字节的存储单元给变量 c，这个顺序不能改变，因为程序是按指令一条一条执行的。

2. 实型常量和变量

实型数据用来表示具有小数点的实数，即以小数形式或指数形式出现的实数，可分为单精度型(float)、双精度型(double)。程序执行时，系统会为 float 型变量分配 4 字节，为 double 型变量分配 8 字节。例如，−0.33、56.2、10e3 都是实型常量。又例如，对于

```
float a = 3.5,b = 9.8;
```

变量 a 和 b 都是实型变量。

3. 字符常量和变量

用单引号包含的一个字符表示字符型常量，并且只能用单引号括起来一个字符。例如，'a'、'W'、'2'、'*'等都是字符常量。

还有一些字符前面加了一个“\”；这些字符称为转义字符，已不是字符原来表示的含义了。

\n 表示换行符。

\0 是字符串结束标志。

\t 即 Tab，表示横向跳格，每次跳 8 个字符的位置。

\ddd ddd 表示 1 到 3 位八进制数所代表的字符，且字符不能超过 3 位。例如，\101 表示八进制的 101，转换成十进制后为 65，ASCII 码为 65 的字符是 A，使用 printf 函数输出该字符：

```
printf("%c",'\101');
```

结果为：

```
A
```

\xhh hh 表示 1 到 2 位十六进制数所代表的字符。例如，\x41 表示十六进制的 41，转换成十进制后为 65，ASCII 码为 65 的字符是 A，使用 printf 函数输出该字符：

```
printf("%c",'\x41');
```

结果为：

```
A
```

数据类型为 char 的变量称为字符变量。一个字符变量在内存中占 1 字节的大小。例如，定义两个字符变量 c1 和 c2 时使用下面的定义方法：

```
char c1,c2;
```

对于字符的 ASCII 码表，读者可以自行去查看一下，十进制数从 0 到 255，对应了 256 个字符。需要记住大写字母 A～Z 的 ASCII 码为 65～90，小写字母 a～z 的 ASCII 码为 97～122。大写字母和对应的小写字母的 ASCII 码正好相差 32。

例 1-3　下列代码的运行结果是什么？

```
#include <stdio.h>
void main()
```

```
{
  char c1,c2;
  c1 = 67;
  c2 = 68;
  printf("%c,%d\n",c1,c1);
  printf("%c,%d\n",c2,c2);
}
```

运行结果为：

```
C,67
D,68
```

对于同一个字符变量，按照字符型输出就是对应的字符，按照整型输出就是ASCII表中字符对应的ASCII码。

1.2 运算符和表达式

C语言中包括算术运算符、赋值运算符、关系运算符、逻辑运算符、条件运算符等多种运算符，运算符不仅有优先级别，还有结合性。用算术运算符和括号将运算对象连接起来的、符合C语言语法规则的式子称为算术表达式。

1.2.1 算术运算符

基本的算术运算符包括加、减、乘、除、取余等。

＋ ：

① 加法运算符。其为双目运算符，如1＋2等，并且具有右结合性。

② 正号运算符。其为单目运算符，如＋1、＋0.5等。

－ ：

① 减法运算符。其为双目运算符，如1－2等，并且具有左结合性。

② 负号运算符。其为单目运算符，如－1、－0.5等。

＊ ：乘法运算符。其为双目运算符，如1＊2等，并且具有左结合性。

/ ：除法运算符。其为双目运算符，如1/2等，并且具有左结合性。

两个整型变量相除时，结果也为整型，若结果不是整型，则只取结果的整数部分，舍去小数。例如，10/6结果为1，得到这个结果并不是因为进行了四舍五入，而是只取商的整数部分，小数部分舍去了。

如果运算量中有一个是实型，则结果为双精度实型。例如，10.0/6结果为1.666667，

C 语言中对于实数的输出默认保留六位小数。

%:取余运算符。其为双目运算符,且具有左结合性,必须是两个整数才能进行取余运算。

除了这五种基本的算术运算符,还有两种算术运算符,分别为自增运算符、自减运算符。

++:自增运算符。其功能是使变量的值自增 1,例如,i++就是使 i 加上 1。

--:自减运算符。其功能是使变量的值自减 1,例如,i--就是使 i 减去 1。

对于自增和自减运算符,在使用时遵循两个原则:

① 若变量在前,符号在后,则先使用变量的值参与运算,再使变量自增或自减 1。

② 若符号在前,变量在后,则先使变量自增或自减 1,再参与运算。

如果变量没有参与其他运算,只有自增或自减构成一条指令,那么变量和符号的前后顺序不影响结果,都是变量自增或自减 1。例如,i++和++i 的结果都是变量 i 的值加 1;i--和-- i 的结果都是变量 i 的值减 1。如果变量参与了其他运算,结果就不同了。在后面的章节中将详细介绍该部分内容。

1.2.2　赋值运算符

=:赋值运算符。其表示将一个数据赋给一个变量。例如,"i=1"表示将 1 赋值给变量 i,那么变量 i 的值就是 1。它也可以表示将一个表达式赋值给一个变量。例如,"i=i+1"表示将变量 i 与 1 进行加法运算后将和赋值给左边的变量 i。因为左右两边都有变量 i,也可以写成"i+=1",中间的运算符是由"+"和"="两个运算符构成的"+=",其运算规则是运算符的左右两边先进行算术运算符"+"的运算,然后将结果赋值给左侧的变量。这种运算符称为复合运算符,类似的运算符还有"-=""*=""/=""%="等,它们的运算规则是一样的,即左右两边先进行算术运算,再将结果赋值给左侧的变量。例如,"i-=1"相当于"i=i-1";"i*=2"相当于"i=i*2";"i/=2"相当于"i=i/2";"i%=2"相当于"i=i%2"。但复合运算符不能将表达式赋值给表达式。例如,使用"i+j=t"将变量 t 赋值给"i+j",这样的赋值方式是错误的。

1.2.3　关系运算符

关系运算符包括以下六种:

<:小于。

<=:小于或等于。

>:大于。

>=:大于或等于。

==:等于。

!=:不等于。

其优先级别是前四种运算符“<”“<=”“>”“>=”的优先级别较高,后两种运算符“==”“!=”的优先级别较低。关系运算符的优先级低于算术运算符,但关系运算符的优先级高于赋值运算符。

用关系运算符将两个常量或变量或两个表达式(可以是算术表达式或关系表达式、赋值表达式、字符表达式等)连接起来的式子称为关系表达式。例如,以下这些式子 x>y,x+z>y+z,(x=1)>(y=2),'x'<'y',(a>b)>(a<c),1!=0,2==2 都是关系表达式,表示的是关系运算符两边的数值或表达式大小的关系。

关系表达式的结果是个逻辑值:真或假。如果关系成立,即为真,用 1 表示;如果关系不成立,即为假,用 0 表示。因此关系表达式的结果只有 0 和 1 两个值。例如,对于 3>5,此关系不成立,结果为 0。

对于两个字符的比较,需要通过 ASCII 码对照表查看两个字符的 ASCII 码。例如,字符'x'的 ASCII 码为 120,'y'的 ASCII 码为 121,因此,'x'<'y'的值为真,也就是关系表达式的结果为 1。对于(x=1)>(y=2),变量 x 赋值为 1,变量 y 赋值为 2,因此,左边为 1,右边为 2,关系表达式不成立,为假,即关系表达式的结果为 0。对于 1!=0,由于 1 不等于 0 是对的,因此,1!=0 的结果为真,即关系表达式的结果为 1。对于 2==2,由于 2 和 2 相等,因此,2==2 的结果为真,即关系表达式的结果为 1。

1.2.4 逻辑运算符

逻辑运算符包括三种:

&&:逻辑与。

||:逻辑或。

!:逻辑非。

逻辑运算符中的“&&”和“||”的运算优先级低于关系运算符的,“!”的运算优先级高于算术运算符的。用逻辑运算符将两个常量、变量、逻辑量或关系表达式连接起来的式子称作逻辑表达式。逻辑与运算的规则如下:

1&&1=1;

1&&0=0;

0&&1=0;

0&&0=0。

逻辑与运算的两边都为 1 时,结果才为 1,否则结果就为 0。

逻辑或运算规则如下:

1||1=1;

1||0=1;

0||1=1;

0||0=0。

逻辑或运算的两边只要有一边为 1,结果就为 1。两边都为 0 时,结果才为假。

逻辑非运算规则如下:

!1=0;

!0=1。

逻辑非运算对于 0 的非运算结果为 1,所有不是 0 的情况做逻辑非操作的结果都是 0。

三种逻辑运算中的 1 代表所有为真的情况,0 代表所有为假的情况。

例如,若 a=3,b=2,则下面的几个逻辑运算结果分别如下。

!b:值为 0。因为 b 的值为 2,是真的,故逻辑运算结果为 0。也就是只有 0 是假的,其他数值都是真的,无论正负。

a>0&&b<0:值为 0。因为 a>0 为真,即为 1,而 b<0 假,即为 0,故整个逻辑运算的结果为 0。

a&&b:值为 1。因为 a 为 3,b 为 2,都为真,即都为 1,故整个逻辑运算结果为 1。

a||b:值为 1。因为 a、b 都为真,即都为 1,故整个逻辑运算结果为 1。

!a||b:值为 1。因为!a 为 0,而 b 为真,即为 1,所以逻辑或两边有一边为 1,整个逻辑运算结果即为 1。

4&&0||2:值为 1。因为 4&&0 的结果为 0,而 2 为真,故整个逻辑运算结果为 1。

当一个逻辑表达式中出现多个运算符时,一般是按从左向右的顺序来执行,但是要遵循各种运算之间的优先顺序。算术运算符、赋值运算符、关系运算符、逻辑运算符之间的优先顺序如表 1-1 所示。

表 1-1　运算符优先级表

运算符类别	具体名称	优先顺序
逻辑运算符	!	从高到低
算术运算符	+、-、*、/、%	
关系运算符	<、<=、>、>=	
关系运算符	==、!=	
逻辑运算符	&&、||	
赋值运算符	=	

例如,对于“6>2&&7<3-!0”,执行的顺序如下:!0 的结果为 1→3-1 的结果为 2→6>2 的结果为 1→7<2 的结果为 0→1&&0 的结果为 0。

但对于逻辑操作也有些情况要考虑。在逻辑表达式的运算过程中,并不是所有的逻辑运算符都要被执行,要遵循下面的两条规则:

① 逻辑与运算中，对于a&&b&&c，只有a为真时，才需要判断b的值；只有a和b都为真时，才需要判断c的值。如果a的值为假，整个表达式的值即为0，那么b和c不需要执行；如果a为真，b为假，整个表达式的值即为0，那么c不需要执行。

② 逻辑或运算中，对于a||b||c，只要a为真，整个表达式的值即为1，那么b和c不需要执行。如果a为假，才需要判断b；如果a和b都为假，才需要判断c。因此b、c能不能执行取决于a的值。

例如，对于“6>2&&7<3&&2<8”，执行的顺序如下：6>2结果为1→7<3结果为0→1&&0结果为0，故整个表达式值为0，2<8并没有执行。

例1-4 a=1，b=2，c=5，d=4，n=3，求解执行(a>b)&&(n=c>d)后n的值。

分析：

由于a>b不成立，值为0，整个表达式的值即为0，右侧的表达式并没有执行，因此n的值不变，仍为3。如果右边仍然去求解，那么由于c>d成立，值为1，n的值为1，结果就是错的。因此在逻辑运算中，一定要按照上述的两条规则来进行运算。

1.2.5 条件运算符

条件表达式的基本格式为：

表达式1?表达式2:表达式3

连接三个表达式的“?”和“:”是一个整体，不能分开。

执行的顺序为：先求解表达式1，若表达式1为真，则求解表达式2，而表达式2的值就是整个条件表达式的值。若表达式1的值为假，则求解表达式3，而表达式3的值就是整个条件表达式的值。例如，“3>2?6:7”的执行顺序为：先判断3>2结果为真，那么表达式2的值就是整个条件表达式的结果，即“3<2?6:7”的结果就是6。

也可以将条件表达式的值赋值给某一变量，条件运算符优先于赋值运算符，并且条件运算符的结合方向为自右至左。例如，i=3>2?6:7也就是将“3>2?6:7”的结果赋值给变量i，即i=6。

注意：条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值运算符。因此，3>2是关系表达式，大于号优先级别高于条件运算符，因此不需要将3>2用括号括起来。

如果在表达式中出现多个条件运算符，则采用自右向左的结合方式。例如，“b>a?b:c>d?c:d”的执行的顺序应该是先求解“c>d?c:d”的结果，再将该结果作为第三个表达式，跟左边的部分构成新的条件表达式去求解，相当于“b>a?b:(c>d?c:d)”。

例1-5 编程实现如下要求：输入两个整数，输出其中较小的那个数。

程序如下：

```
#include <stdio.h>
void main()
```