项目十六 红外遥控数码管

数码管，常见的用来显示数字的，比如像计算器。在本实验之前我们先来了解一下数码管是如何工作的。数码管，其实也算是 LED 中的一种。数码管的每一段，都是一个独立的 LED，通过数字引脚来控制相应段的亮灭就能达到显示数字的效果。下面让我们通过实验的方式来感受一下数码管的神奇之处吧！

所需材料

- 1× 八段数码管
- 8× 220 欧电阻 

硬件连接

按下图连线图连接，注意数码管各段所对应的引脚。右边引脚说明图上为什么画这么几个箭头呢？个人觉得，这样看起来更方便。可以给你作为参考。我们从上面一排看，红色箭头的方向，从右往左，b → a → f → g 的顺序正好对应，下面红色箭头逆时针顺序 b → a → f → g。蓝色箭头也是表达的同样的意思。

我还特意在连接图上，对数码管所连接的引脚做了标示。这样就能更清楚的知道哪个引脚控制哪一段了。这 8 个电阻同样是起限流的作用。

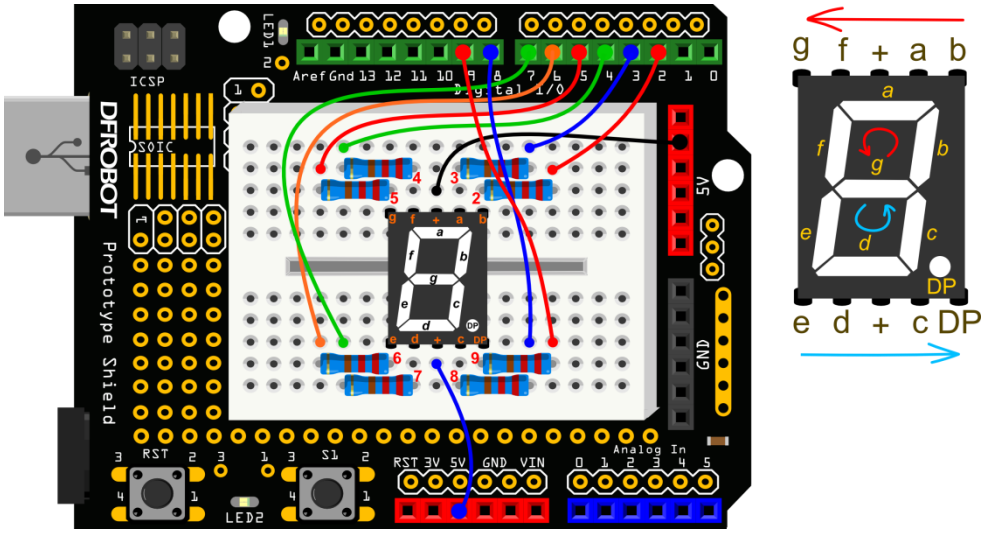

图 15-1 数码管显示连线图

输入代码

样例代码 15-1：

```
//项目 15 - 数码管显示
void setup(){
  for(int pin = 2 ; pin <= 9 ; pin++){        // 设置数字引脚 2~9 为输出模式
      pinMode(pin, OUTPUT);
      digitalWrite(pin, HIGH);
  }
}

void loop() {
     // 显示数字 0
    int n0[8]={0,0,0,1,0,0,0,1};
    //数字引脚 2~9 依次按数组 n0[8]中的数据显示
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n0[pin-2]);
    }
    delay(500);

    // 显示数字 1
    int n1[8]={0,1,1,1,1,1,0,1};
    // 数字引脚 2~9 依次按数组 n1[8]中的数据显示
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n1[pin-2]);
    }
    delay(500);

    // 显示数字 2
    int n2[8]={0,0,1,0,0,0,1,1};
    // 数字引脚 2~9 依次按数组 n2[8]中的数据显示
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n2[pin-2]);
    }
    delay(500);
```

```
// 显示数字 3
int n3[8]={0,0,1,0,1,0,0,1};
// 数字引脚 2~9 依次按数组 n3[8]中的数据显示
for(int pin = 2; pin <= 9 ; pin++){
   digitalWrite(pin,n3[pin-2]);
}
delay(500);

// 显示数字 4
int n4[8]={0,1,0,0,1,1,0,1};
// 数字引脚 2~9 依次按数组 n4[8]中的数据显示
for(int pin = 2; pin <= 9 ; pin++){
   digitalWrite(pin,n4[pin-2]);
}
delay(500);

// 显示数字 5
int n5[8]={1,0,0,0,1,0,0,1};
// 数字引脚 2~9 依次按数组 n5[8]中的数据显示
for(int pin = 2; pin <= 9 ; pin++){
   digitalWrite(pin,n5[pin-2]);
}
delay(500);

// 显示数字 6
int n6[8]={1,0,0,0,0,0,0,1};
// 数字引脚 2~9 依次按数组 n6[8]中的数据显示
for(int pin = 2; pin <= 9 ; pin++){
   digitalWrite(pin,n6[pin-2]);
}
delay(500);

// 显示数字 7
int n7[8]={0,0,1,1,1,1,0,1};
// 数字引脚 2~9 依次按数组 n7[8]中的数据显示
for(int pin = 2; pin <= 9 ; pin++){
   digitalWrite(pin,n7[pin-2]);
```

```
    }
    delay(500);

    // 显示数字 8
    int n8[8]={0,0,0,0,0,0,0,1};
    // 数字引脚 2~9 依次按数组 n8[8]中的数据显示
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n8[pin-2]);
    }
    delay(500);

    // 显示数字 9
    int n9[8]={0,0,0,0,1,1,0,1};
    // 数字引脚 2~9 依次按数组 n9[8]中的数据显示
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n9[pin-2]);
    }
    delay(500);
}
```

完成下载后，数码管就会循环显示 0~9 的数字。由于要看懂代码的话，首先需要了解数码管的构造，所以我们这回先说硬件部分。

硬件回顾

数码管

数码管其实就是一个前面介绍的 led 的组合体，这个组合体包含 8 个 led，所以也称之为八段数码管。说白了就八个灯。哪八段？不用多说了吧！a 到 g 以及小数点 DP。其实用法和前面说的 LED 也是一样的，每段都是一个发光二极管，分别用 8 个数字口来控制它们的亮灭，通过不同段的显示，就能组成 0~9 的数字。比如，我们让 b，a，f，e，d，c 亮起的话，就能显示一个数字“0”了。

下图 11-2 是个引脚说明图，不陌生了吧！在前面硬件连接的时候，已经看到过一次了。

这里，b → a → f → g → e → d → c → DP 分别连接到 Arduino 数字引脚 2~9。

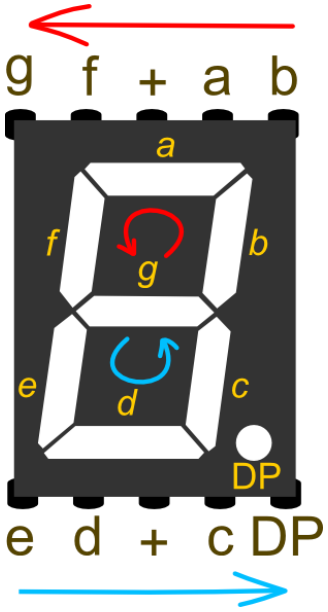

图 11-2 引脚说明图

数码管一共有 10 个引脚。a~DP 这 8 个引脚接到数字口，那还有两个引脚呢？这是公共端，LED 只有一端是不能被点亮的。我们在 RGB 灯那章讲到过共阴共阳的问题，数码管也存在共阴共阳问题。所谓共阳就是公共端接+5V，共阴则是公共端接 GND。

数码管的共阴共阳在使用上有什么区别

共阳数码管，它们公共端接 5V，那在代码中，控制另一端的数字引脚为 LOW，这样才能让数码管点亮。如果是共阴数码管，公共端接 GND，在代码中中，控制另一端数字引脚为 HIGH，才让数码管点亮。

所以，共阴共阳只是在代码上要稍作修改。我们这里选用的是共阳数码管。硬件有了了解，我们来看看软件部分。

代码回顾

硬件部分我们已经说过，数码管需要接到 8 个数字引脚，所以在一开始，需要定义 8 个数字引脚作为输出。这次我们用一个 for 循环来完成这 8 个数字引脚的设置。数码管 b，a，f，g，e，d，c，DP 分别和 Arduino 数字引脚 2~9 对应。

```
for(int pin = 2 ; pin <= 9 ; pin++){
      pinMode(pin, OUTPUT);
      digitalWrite(pin, HIGH);
}
```

从引脚 2 开始，一直循环到引脚 9，都设为 OUTPUT 模式，初始化为 HIGH。前面说过，共阳的话，设置 HIGH，不被点亮，所以开始先不点亮数码管。（当然，你一个一个引脚分开设置输出模式也是不会错的，只是会让代码显得很冗长。）

好了，到了主函数，要分别显示 0~9 的数字。是不是觉得代码大部分都是相似的。所以，我们只要看明白如何显示数字 0，那整段代码就都迎刃而解了。

```
int n0[8]={0,0,0,1,0,0,0,1};
```

这里我们要引入一个数组的概念。数组是一个变量的集合，可以通过索引号来找到数组中的元素。在我们的程序中，声明了一个 int 型的数组。并取名为 n0。之后用 8 个数值来初始化这个数组。那如何获得数组中的元素呢？你只需要简单的指出这个元素的索引号。数组是从 0 开始索引的，这意味着数组中的第一个元素的索引号为 0 而不是 1，因此数组中的 8 个元素的索引号是 0~7。在这里元素 4，对应索引号为 3（n0[3]），值为 1。元素 8（索引号 7，n0[7]）的值为 1。

声明中 n0[8]的方括号中的 8 代表有 8 个元素。

定义完数组后，进入又一个 for 循环。

```
for(int pin = 2; pin <= 9 ; pin++){
      digitalWrite(pin,n0[pin-2]);
}
```

这个 for 循环是给 2~9 引脚写入状态值，也就是 HIGH 还是 LOW，digitalWrite 函数中写入 HIGH 的另一种形式就是写入“1”，LOW 则可以写为“0”。我们通过数组索引的方式给 2~9 引脚赋值。

比如当 pin=2，代入 n0[pin-2]中，对应为 n0[0]，n0[0]意思是获得数组的第一个元素，为 0。完成了引脚 2 置低（LOW）。我们前面说了，共阳的数码管，置低（LOW）的话，是被点亮，所以，b 端被点亮了。循环到 pin=3，a 段被点亮,。循环到 pin=4，f 段被点亮，依次类推……。

整个循环过程如下：

pin=2 → n0[0] =0 → digitalWrite(2,0) → b 段点亮

pin=3 → n0[1] =0 → digitalWrite(3,0) → a 段点亮

pin=4 → n0[2] =0 → digitalWrite(4,0) → f 段点亮

pin=5 → n0[3] =1 → digitalWrite(5,1) → g 段不点亮

pin=6 → n0[4] =0 → digitalWrite(6,0) → e 段点亮

pin=7 → n0[5] =0 → digitalWrite(7,0) → d 段点亮
pin=8 → n0[6] =0 → digitalWrite(8,0) → c 段点亮
pin=9 → n0[7] =1 → digitalWrite(9,1) → DP 段不点亮

这样就完成了显示数字“0”了。同样用数组的方法显示数字 1~9。自己动手画一下，哪几段亮，哪几段不亮就一目了然了。

如果代码 1 弄明白后，我们要教大家一种更简单的方法，那就是先输入代码 2。

输入代码 2

我们这里要教大家实现这个数码管 0~9 循环的代码另一种写法，上面我们说到了数组，通过创建 10 个数组用于显示 0~9。这里同样也还是用数组写，区别在于代码 1 中其实准确的说应该叫一维数组，我们这里用到二维数据。这样一来，可以让代码看起来更简洁。动手输一下下面这段代码吧，看看是不是同样的效果!

样例代码 15-2：

```
//项目 11 - 数码管数字显示
int number[10][8] =
{
  {0,0,0,1,0,0,0,1},    //显示 0
  {0,1,1,1,1,1,0,1},    //显示 1
  {0,0,1,0,0,0,1,1},    //显示 2
  {0,0,1,0,1,0,0,1},    //显示 3
  {0,1,0,0,1,1,0,1},    //显示 4
  {1,0,0,0,1,0,0,1},    //显示 5
  {1,0,0,0,0,0,0,1},    //显示 6
  {0,0,1,1,1,1,0,1},    //显示 7
  {0,0,0,0,0,0,0,1},    //显示 8
  {0,0,0,0,1,1,0,1}     //显示 9
};
void numberShow(int i){          //该该函数用来显示数字
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin, number[i][pin - 2]);
    }
}
```

```
void setup(){
  for(int pin = 2 ; pin <= 9 ; pin++){       // 设置数字引脚 2~9 为输出模式
      pinMode(pin, OUTPUT);
      digitalWrite(pin, HIGH);
  }
}

void loop() {
    for(int j = 0; j <= 9 ; j++){
          numberShow(j);        //调用 numberShow()函数，显示 0~9
          delay(500);
    }
}
```

代码回顾 2

对比一下代码 1，能发现明显的区别在哪里了吗？代码 1 中，我们创建了 10 个一维数组，代码 2 只需要创建一个二维数组就全部搞定了。不要被什么一维、二维数组的名字给吓唬到，其实用法一样的。

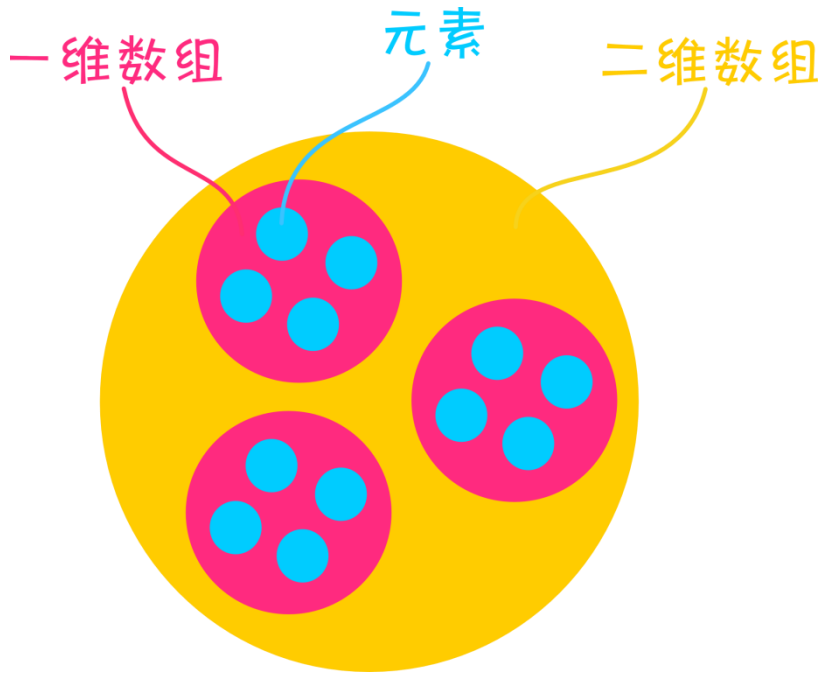

图 11-3 数组关系图

通过上面这个图，元素、一维数组、二维数组之间的关系就一目了然了。一维数组由元素组成，而二维数组则是由一个个一维数组组成的，关系就是那么简单。

代码 1 中，分别用 10 个数组，每个数组有 8 个元素，每个元素依次对应到数码管 b~DP 引脚状态值，这样就能在数码管上反映为 0~9 的数字显示。

而我们现在则是把前面散开的 10 个一维数组整合到一起，变为一个二维数组。同样通过索引的方式来找到这些元素。但，还是不要忘了索引号也是从 0 开始的！前面的方括号写入的是只是元素个数。看一下代码：

number[0][0]

```
int number[10][8] =
{
  {0,0,0,1,0,0,0,1},    //显示 0
  {0,1,1,1,1,1,0,1},    //显示 1
  {0,0,1,0,0,0,1,1},    //显示 2
  {0,0,1,0,1,0,0,1},    //显示 3
  {0,1,0,0,1,1,0,1},    //显示 4
  {1,0,0,0,1,0,0,1},    //显示 5
  {1,0,0,0,0,0,0,1},    //显示 6
  {0,0,1,1,1,1,0,1},    //显示 7
  {0,0,0,0,0,0,0,1},    //显示 8
  {0,0,0,0,1,1,0,1}     //显示 9
};
```

number[9][7]

这就是一个二维数组。索引号从 0 开始，如果让你找 number[0][0]，能找到是哪个数吗？就是二维数组中第 1 行的第 1 个数，为 0。number[9][7]也就是第 10 行的第 8 个数，为 1。

```
void numberShow(int i){          //该该函数用来显示数字
    for(int pin = 2; pin <= 9 ; pin++)
       digitalWrite(pin, number[i][pin - 2]);
}

void loop() {
    for(int j = 0; j <= 9 ; j++){
          numberShow(j);         //调用 numberShow()函数，显示 0~9
          delay(500);
    }
}
```

上面这两段代码我们整合在一起看，loop()主函数中，for 循环让变量 j 在 0~9 循环，j 每赋一次值，numberShow()函数就要运行一次。

numberShow()函数整个运行过程如下：

程序一开始 j=0，numberShow(j)为 numberShow(0),跳回到上面的 numberShow()函数，i 现在的值就为 0 了，pin 初始值为 2，所以 digitalWrite()现在值为 digitalWrite(2,number[0][0])，回到数组 number[10][8]中找到 number[0][0]对应的值，为 0。此时，digitalWrite(2,0)，代表引脚 2 被置 LOW，引脚 2 对应的 b 段点亮（共阳数码管置 LOW 才被点亮）。之后再是循环 pin=3，pin=4，……，一直到 pin=9 整个 for 循环才结束，也代表数组的第一行的 8 个元素全被运行了一遍，最终显示一个数字“0”。

回顾一下代码 1 是如何显示一个数字“0”的：

```
// 显示数字 0
int n0[8]={0,0,0,1,0,0,0,1};
//数字引脚 2~9 依次按数组 n0[8]中的数据显示
  for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n0[pin-2]);
  }
```

原理是一样的，通过给引脚 2~9 循环赋值，控制数码管 b~DP 段亮灭，就能显示出一个我们想要的数字。

numberShow(0)循环完后，再次回到 loop()中的 for 函数：

j=1 → numberShow(1) → i=1 → number[1][pin-2] → 显示数字 1

j=2 → numberShow(2) → i=2 → number[2][pin-2] → 显示数字 2

j=3 → numberShow(3) → i=3 → number[3][pin-2] → 显示数字 3

……

j=3 → numberShow(9) → i=9 → number[9][pin-2] → 显示数字 9

好了，这就是整段代码的分析过程，好好体会一下一维数组和二维数组的区别，以及整段代码如何巧妙运行的。

了解了数码管和红外接收管各自的工作原理后，我们需要把这两者结合起来，看看红外接收管和数码管结合能迸发出怎样的火花呢？想到了吗？遥控数码管！Arduino 控制器把

红外接收管从 Mini 遥控器那儿接到的信号，经处理传达给数码管。让 Mini 遥控器上 0~9 对应在数码管上显示 0~9，除此之外，还有递减，递增的功能。

所需材料

- 1× 红外接收管 
- 1× Mini 遥控器 
- 1× 八段数码管 
- 8× 220 欧电阻 

硬件连接

你会发现在硬件连接上同样就是把项目十四和十五结合在一起，并没有什么太大变化，如果在连接数码管时候有些不明白，可以回看一下项目十五。

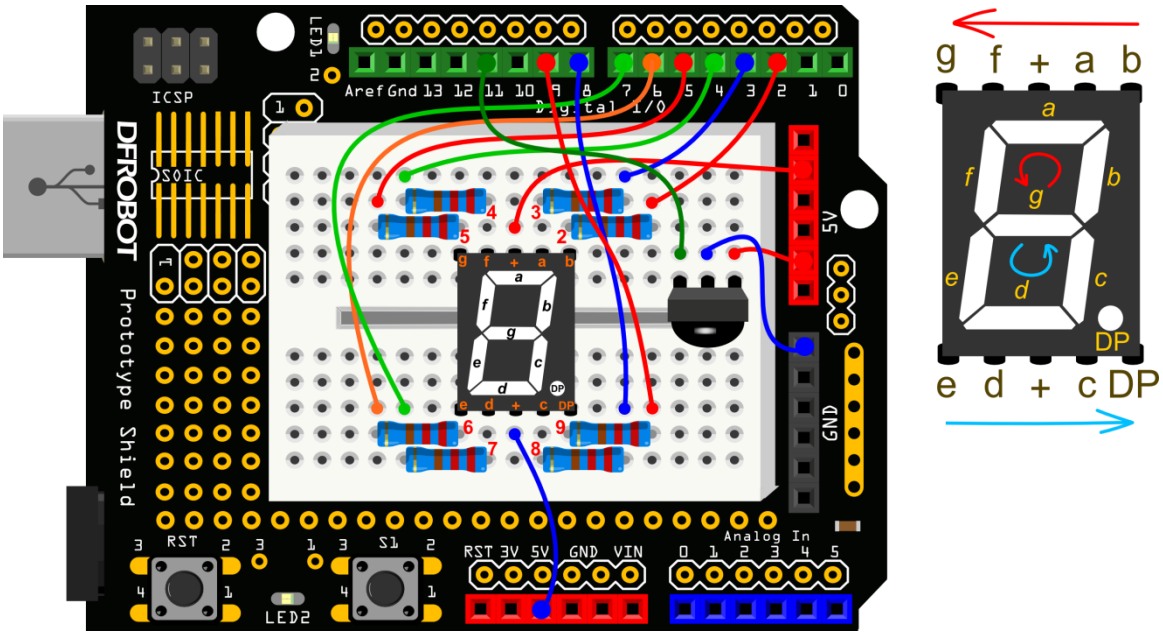

图 16-1 红外遥控数码管连线图

输入代码

在输入代码的过程中，结合前面的项目十四和十五，看看整段代码是如何把这两者整合的，程序中是如何处理把红外接收管接到的信号，再转变为数码管的显示的。

样例代码 13-1：

```
//项目十三 – 红外遥控数码管
#include <IRremote.h>       //调用 IRremote.h 库
int RECV_PIN = 11;          //定义 RECV_PIN 变量为 11
IRrecv irrecv(RECV_PIN);    //设置 RECV_PIN (也就是 11 引脚) 为红外接收端
decode_results results;     //定义 results 变量为红外结果存放位置
int currentNumber = 0;      //该变量用于存放当前数字

long codes[12]=             //该数组用来存放红外遥控器发出的红外码
{
   0xFD30CF,0xFD08F7,                 // 0 ,1
```

```
    0xFD8877,0xFD48B7,                  // 2 ,3
    0xFD28D7,0xFDA857,                  // 4 ,5
    0xFD6897,0xFD18E7,                  // 6 ,7
    0xFD9867,0xFD58A7,                  // 8 ,9
    0xFD20DF,0xFD609F,                  // + ,-
};

int number[10][8] =             //该数组用来存放数码管显示的数字
{
  {0,0,0,1,0,0,0,1},//0
  {0,1,1,1,1,1,0,1},//1
  {0,0,1,0,0,0,1,1},//2
  {0,0,1,0,1,0,0,1},//3
  {0,1,0,0,1,1,0,1},//4
  {1,0,0,0,1,0,0,1},//5
  {1,0,0,0,0,0,0,1},//6
  {0,0,1,1,1,1,0,1},//7
  {0,0,0,0,0,0,0,1},//8
  {0,0,0,0,1,1,0,1} //9
};

void numberShow(int i) {                    //该函数用来让数码管显示数字
      for(int pin = 2; pin <= 9 ; pin++){
         digitalWrite(pin, number[i][pin - 2]);
      }
}

void setup(){
      Serial.begin(9600);             //设置波特率为 9600
      irrecv.enableIRIn();            //启动红外解码

      for(int pin = 2 ; pin <= 9 ; pin++){    //设置数字引脚 2~9 为输出模式
          pinMode(pin, OUTPUT);
          digitalWrite(pin, HIGH);
      }
}
```

```
void loop() {
  //判断是否接收到解码数据,把接收到的数据存储在变量 results 中
  if (irrecv.decode(&results)) {
      for(int i = 0; i <= 11; i++){
        //判断是否接收到 0~9 按键的红外码
        if(results.value == codes[i]&& i <= 9){
              numberShow(i);  //在数码管上对应显示 0~9
              currentNumber = i;  //把当前显示的值赋给变量 currentNumber
              Serial.println(i);
              break;
        }

        // 判断是否接收到递减的红外码, 并且当前值不为 0
        else if(results.value == codes[10]&& currentNumber != 0){
              currentNumber--;              //当前值递减
              numberShow(currentNumber);    //数码管显示递减后的值
              Serial.println(currentNumber); //串口输出递减后的值
              break;
        }

        //判断是否接收到递增的红外码, 并且当前值不为 9
        else if(results.value == codes[11]&& currentNumber != 9){
              currentNumber++;              //当前值递增
              numberShow(currentNumber);    //数码管显示递增后的值
              Serial.println(currentNumber); //串口输出递增后的值
              break;
        }
     }

    Serial.println(results.value, HEX);  //串口监视器查看红外码
    irrecv.resume();   //等待接收下一个信号
  }
}
```

图 13-2 遥控数按键说明

下载完代码后，尝试按下上图 13-2 指出部分的按钮，看看是数码管是个怎样的变化。

代码回顾

程序一开始还是对红外接收管的一些常规定义，按项目十二原样搬过来就可以了。

```
#include <IRremote.h>       //调用 IRremote.h 库
int RECV_PIN = 11;          //定义 RECV_PIN 变量为 11
IRrecv irrecv(RECV_PIN);    //设置 RECV_PIN（也就是 11 引脚）为红外接收端
decode_results results;     //定义 results 变量为红外结果存放位置
int currentNumber = 0;      //该变量用于存放当前数字
```

在这里，我们多定义了一个变量 currentNumber，通过名字应该就可以看出来含义了吧。这个变量的作用是，用来存储当前的数字，数字递增递减是能找到对应的参照点。

同样用数组的方式来存放这些红外码，0x-表示是 16 进制。long 是变量的类型，如果你还想用遥控器上的其他按钮来控制做一些其他事情的话，把红外码替换掉就好了。

```
long codes[12]=              //该数组用来存放红外遥控器发出的红外码
{
  0xFD30CF,0xFD08F7,              // 0 ,1
  0xFD8877,0xFD48B7,              // 2 ,3
  0xFD28D7,0xFDA857,              // 4 ,5
```

```
    0xFD6897,0xFD18E7,          // 6 ,7
    0xFD9867,0xFD58A7,          // 8 ,9
    0xFD20DF,0xFD609F,          // 前进 ,后退
};
```

紧接着是，一个二维数组 number[10][8]的定义。我们在数码管那一章节已有说明了，通过调用数组的元素，把这些元素的值依次赋给数码管显示段的控制引脚，并在 numberShow()函数中得以实现数码管数字显示。

setup()函数中，仍然是波特率设置，启动红外解码，数字引脚模式设置等，这些常规设置。

到了主函数 loop()，一开始还是先判断是否接收到红外码，并把接收到的数据存储在变量 results 中。

```
if (irrecv.decode(&results))
```

一旦接收到数据后，程序就要做两件事。第一件事，判断是哪个红外码，也就能对应找到是哪个按键按下的。第二件事，找到对应按扭后，让数码管干什么事？让我们接着看看程序是如何完成这两件事的。

第一件事：

会有三种情况需要判断，第一种情况，按遥控器 0~9 时，数码管显示数字 0~9。第二种情况，每按下“后退”键，数字在原有基础上递减一位，直到减到 0 为止。第三种情况，每按下“前进”键，数字在原有基础上增一位，直到增到 9 为止。

对这三种情况进行判断，这里呢，同样用到了 if 语句，与以往有所不同的，我们选择用 if…else if。if…else 和 if…else if 的区别在哪儿？区别在于 else if 后面需要接判断表达式，else 不需要判断表达式。然而，不管是 else 还是 else if 都是依附于 if 语句存在的，不能独立使用。

回到代码中，这就是以下三种情况：

```
if(results.value == codes[i]&& i <= 9)
if(results.value == codes[10]&& currentNumber != 0)
if(results.value == codes[11]&& currentNumber != 9)
```

第一个 if 判断的是第一种情况，显示数字 0~9。判断条件就是接收到的数据 results.value 的值是不是数组中 codes[0]~codes[9]的红外码。

第二个 if 判断的是第二种情况，是否接到“后退”键指令，也就是 code[10]= 0xFD20DF，并且当前显示数字不为 0。

第三个 if 判断的是第三种情况，是否接到“前进”键指令，也就是 code[11]= 0xFDA857，并且当前显示数字不为 9。

还有一个问题——如何找到数组中的元素呢？所以，就需要在 if 判断前设置一个 for 循环，让变量 i 一直在 0~11 之间循环。

第一件事判断是哪个红外码完成后，开始执行第二件事。就是每个 if 语句后，都有相应的执行代码。就不一一详细说了。

整段代码就讲完了，这段代码应该是所以项目中最复杂的代码，可以一开始不能完全看明白，不过没关系，实践出真知，通过一遍遍不断的尝试，相信你总有一天能明白的。

课后作业

通过这个遥控项目，DIY 一个你的遥控作品吧！比如简单的会动的小人，结合我们前面的舵机，通过遥控器上不同的按键，让舵机转动不同的角度，感觉随你的控制转动，发挥你的想象做出更多 Arduino 作品吧！