07

第七章

避障机器人

DFROBOT
DRIVE THE FUTURE

前情回顾

在上个章节里，我们学习逻辑运算符、if...else if...else的用法。利用了超声波测距的原理实现了七音符音乐小车的功能。

本章内容

当MAX遇到障碍物实现向右、向左、原地掉头等避障功能，同时RGBled灯会分别亮蓝色、紫红色等颜色，原地掉头时RGBled灯不亮。

本章知识点

1. 随机函数random()的使用；
2. myMax.swerve函数的使用；
3. 学习switch...case语句；
4. if、if...else if、switch..case语句的区别；

一、 编写并下载程序

打开 ArduinoIDE，将下面的代码输入到编辑区中

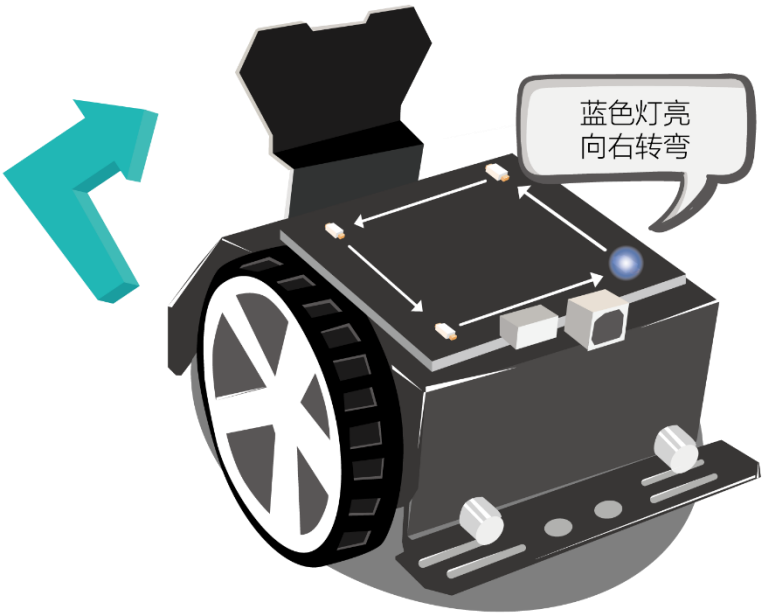
```
/*
程序功能：当 MAX 遇到障碍物实现向右、向左、原地掉头等避障功能，同时 RGBled 灯会分别亮蓝色、紫红色等颜色，原地掉头时 RGBled 灯不亮。
作　　者：DFRobot
*/
#include <DFRobot_MAX.h>     //载入 MAX 驱动库
DFRobot_MAX myMax;           //载入 MAX 驱动函数
int randomNum , getValue;    //定义两个变量

void setup() {
    myMax.begin();         //初始化 MAX
}

void loop() {
    getValue = myMax.distanceValue();     //读取超声波的值
    randomNum=random(1,5);                //随机生成一个 1 到 4 之间的随机数赋给 randomNum。
 if(getValue < 45 ) {                     //当距离<45 时
   myMax.playSound(9);                    //播放音效 9
   if(getValue <= 15) {                   //当距离<=15 时
      myMax.swerve(RETREAT_L, 0, RETREAT_R, 160); delay(1400);     //原地掉头
   }
   else {                                 //否则执行 switch
       switch(randomNum){
       case 1: myMax.swerve(ADVANCE_L, 140, RETREAT_R, 140);   //当 randomNum=1 时，右转
                myMax.rgbLed(0, 0, 0, 255);          //RGB 灯亮蓝色
                delay(600);
                myMax.rgbLed(0, 0, 0, 0);
                break;
      case 2: myMax.swerve(RETREAT_L, 140, ADVANCE_R, 140);   //当 randomNum=2 时，左转
               myMax.rgbLed(1, 255, 0, 255);       //RGB 灯亮紫红色
               delay(600);
               myMax.rgbLed(1, 0, 0, 0);
               break;
      case 3: myMax.swerve(RETREAT_L, 40, RETREAT_R, 140);     //当 randomNum=3 时，后退向右偏转
```

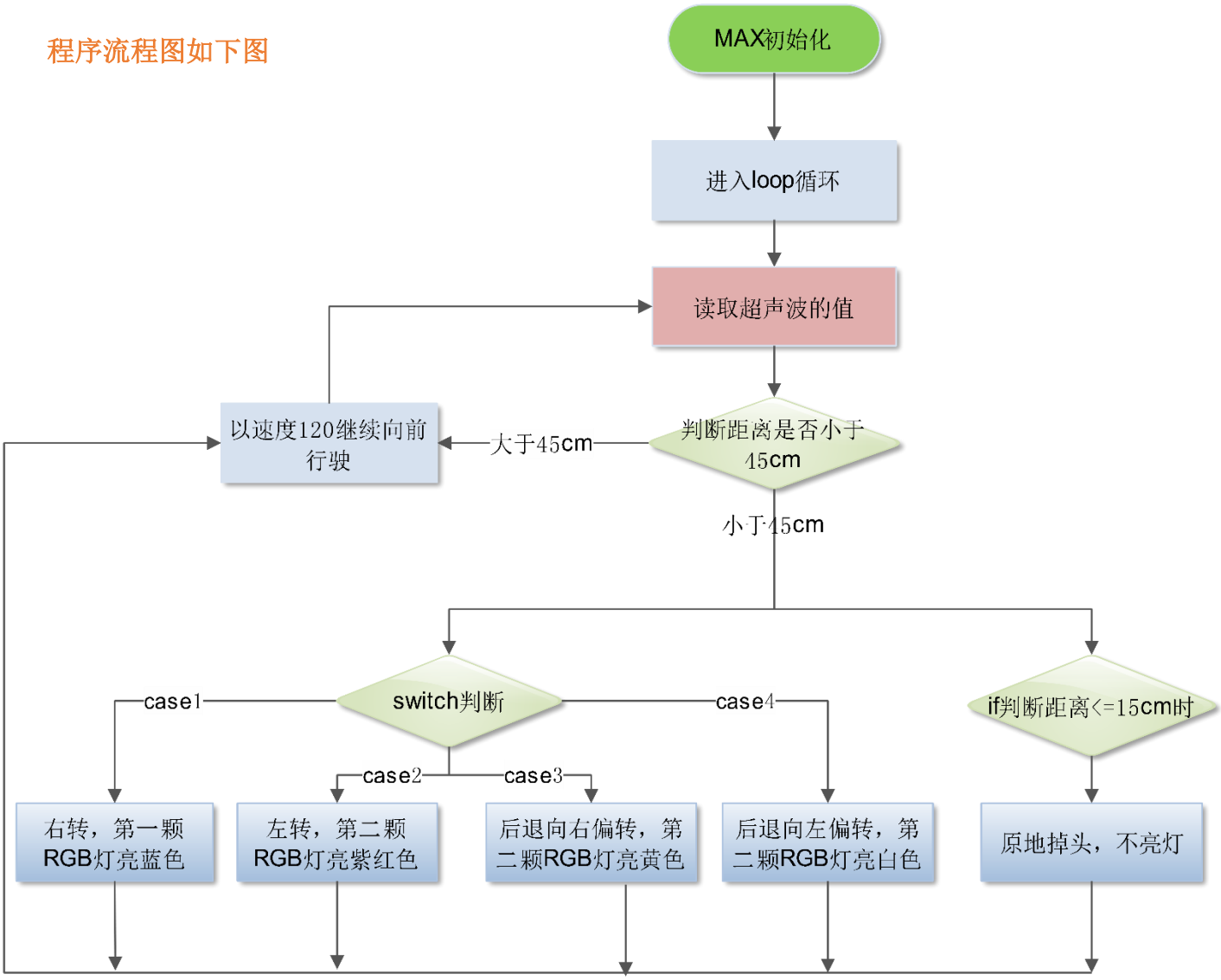
```
            myMax.rgbLed(2, 255, 255, 0);       //RGB 灯亮黄色
            delay(1000);
            myMax.rgbLed(2, 0, 0, 0);
            break;
    case 4: myMax.swerve(RETREAT_L, 140, RETREAT_R, 40);   //当 randomNum=4 时，后退向左偏转
            myMax.rgbLed(3, 255, 255, 255);     //RGB 灯亮白色
            delay(1000);
            myMax.rgbLed(3, 0, 0, 0);
            break;
    }
  }
}
else {                          //大于 45cm 就以 120 的速度前进
    myMax.forward(120, 120);
  }
}
```

上传成功后拨开 MAX 的开关，我们看见的效果就是当 MAX 遇见障碍物会自动向左、向右转，向右转时第一颗 RGBled 灯亮蓝色；向左转时第二颗 RGBled 灯亮紫红色，后退向右偏转时第三颗 RGBled 灯亮蓝色，后退向左偏转时第四颗 RGBled 灯亮白色。



二、 程序分析

程序流程图如下图



代码回顾和分析：

首先，定义两个变量

int randomNum , getValue;

其中变量 randomNum 是选择 MAX 避障的模式:

randomNum 等于 1 时，右转。

randomNum 等于 2 时，左转。

randomNum 等于 3 时，后退向右偏转。

randomNum 等于 4 时，后退向左偏转。

回到我们的程序中，randomNum 的值其实是利用函数 random()随机生成的，random()函数用于生成一个整型的随机数，min 是随机数的最小值，max 是随机数的最大值。

函数格式如下：

代码例程：

```
randomNum=random(1,5);      // 随机生成一个 1 到 4 之间的随机数赋给 randomNum。
```

要想 MAX 能完美的实现避障功能，就需要用到函数：myMax.swerve()，就是它控制 MAX 躲避障碍物。函数格式如下：

```
myMax.swerve(左轮前进或后退，左轮速度，右轮前进或后退，右轮速度);
```

代码例程：

```
myMax.swerve(ADVANCE_L, 140, RETREAT_R, 140);     //右转。
```

参数名称及范围:

左轮后退	RETREAT_L
左轮前进	ADVANCE_L
右轮后退	RETREAT_R
右轮前进	ADVANCE_R
左轮速度	范围 0~255
右轮速度	范围 0~255

MAX 的避障模式有好几种，可是用什么语句判断它是否应该向右转、左转呢？这里就要新学一个语句——switch case 语句。

switch case 语句和 if...else if...语句相似，都是用于多种条件分支判断。

switch 语句的基本形式：

```
switch(表达式){
  case 常量表达式 1: 语句 1;
  break;
  case 常量表达式 2: 语句 2;
  break;
    :
    :
    :
case 常量表达式 n: 语句 n;
  break;
  default: 语句 n+1;
}
```

switch 语句的语义

switch 后面的括号里的表达式可以是任何类型。

当表达式与某一个 case 后面的常量表达式相等时，就执行 case 后面的语句，若没有匹配的就执行 default 后面的语句；每一个 case 的常量表达式的值都不能相同，否则会出现编译错误。

下面就根据本章的例程做一个简单的说明，这样可以更容易理解。

代码例程：

```
switch(randomNum){
   case 1:
// 当 randomNum=1，执行这之间的语句
   break;
      .
      .
      .
   case4:
// 当 randomNum=4，执行这之间的语句
   break;
}
```

注意：

1、default 段可以看情况选择（本章就没有选用）。

2、case 和数字直接隔一个空格，其数字后是冒号，不是分号。

3、关键字 break 可用于退出 switch 语句，通常每条 case 语句都以 break 结尾。如果没有 break 语句，switch 语句将会一直执行接下来的语句直到遇见一个 break，或者直到 switch 语句结尾。

三、 if 语句、if…else if 语句和 switch case 语句的区别

if 语句、if…else if 语句和 switch case 语句都是属于控制语句，但是它们三者使用起来还是有一定的区别，所以我们在使用的时候要根据自己的程序内容选择合适的控制语句。

if 语句， 一般用于判断一个条件，当有多个分支条件的时候都会使用 if…else if 语句；如果还是使用 if 语句会导致程序的执行效率下降，因为多个 if 语句会对每一个判断条件进行判断。这样增加了判断次数。

if…else if 语句，适合判断多个分支条件，相比 if 语句，它会在任何一个环节满足条件时就终止判断，这样减少判断次数，提高了效率。但如果分支条件过多时，嵌套的 if 语句层就会越来越多，这样的程序可读性差，出现 bug 时，不是很容易找错。

switch case 语句，相比 if…else if 语句，它的可读性更好，整体的流程比较明确。执行效率高。但是 switch 语句，仅能够处理整型数值，如果遇到浮点型数据或大规模的连续数值比如：

if(a>=1 && a<=100) 这样的语句，要用 switch 则需要连续写 100 个 case，像这两种情况下，switch 语句就不适用。