海盗船 --- 手机遥控功能

在完成基础功能后，我们对机器人来个升级，给它增加手机蓝牙遥控。可通过手机轻松遥控小车，更可给小车增加炫酷的LED灯光效果。结合我们全新打造的手机端APP，相信你一定会对它爱不释手。并且我们对蓝牙的代码做了优化，更方便初学者使用。

**准备工作：**

* iPhone 或者 iPad × 1
* [走你APP](https://itunes.apple.com/cn/app/zou-ni-lan-ya4.0-kong-zhi-qi/id950937437?mt=8) × 1



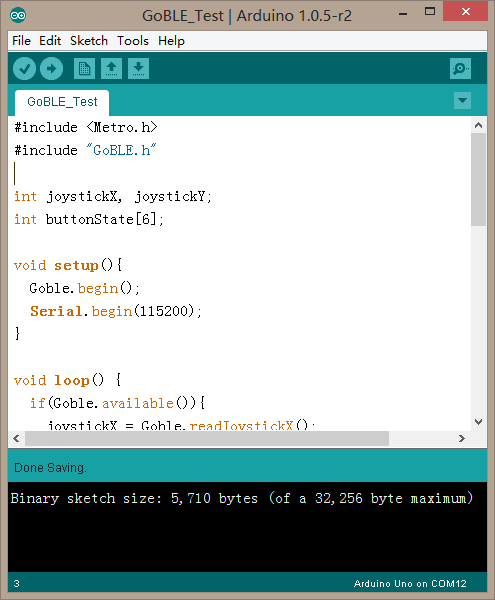
注意：需要 iOS 8.1 或更高版本。此App已针对iPhone 5、iPhone 6 和 iPhone 6 Plus 优化。

由于我们目前只开发了iOS的APP，所以安卓用户暂时没有可直接使用的APP。iPhone用户可在Apple Store下载GoBLE的APP。下载完成后，打开APP，如果呈现出如下所示的界面，说明你APP下载正确了！



**输入测试代码**

插上USB线，你可在软件包中找到GoBLE\_Test.ino的代码，这是一段测试代码。还不能遥控小车的，只是可以从电脑端看到手机发出来的信号。下载一下看了就知道了。下载代码前，不要忘记加载一下遥控的GoBLE的库。下载成功后，信息框会显示Done uploading。



**蓝牙调试**

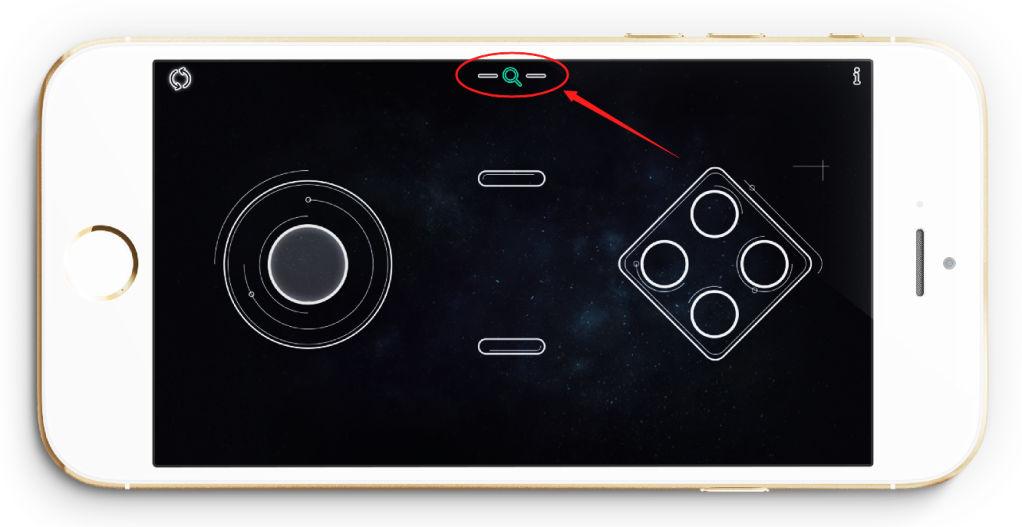
下载完代码，我们需要将手机和蓝牙进行配对。首先 , 记得打开手机蓝牙。然后，打开走你APP，会出现如下界面。



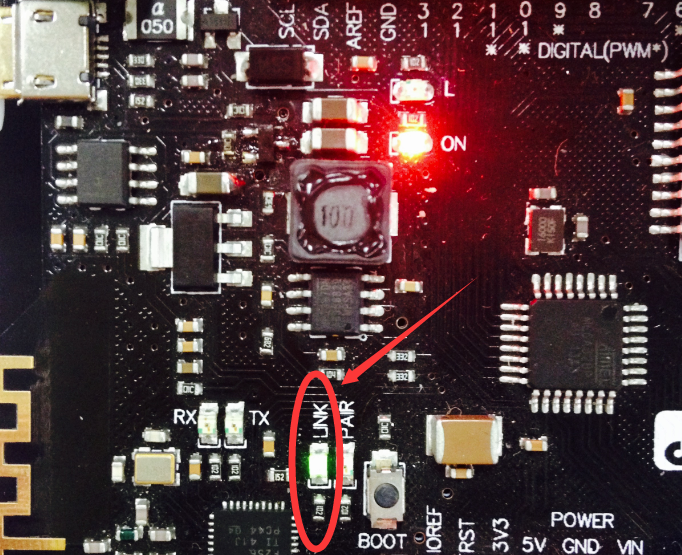
点击蓝牙搜索键（红色放大镜）。此时，会显示所有蓝牙设备，选择Romeo BLE对应的蓝牙设备名称，进行连接。



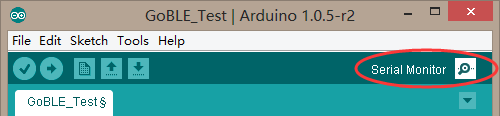
连接成功后，蓝牙搜索键的会由红色转变为绿色。如下图所示：



连接成功后，Roemo BLE上的绿色Link会点亮，此时说明你手机和控制器连接上了。



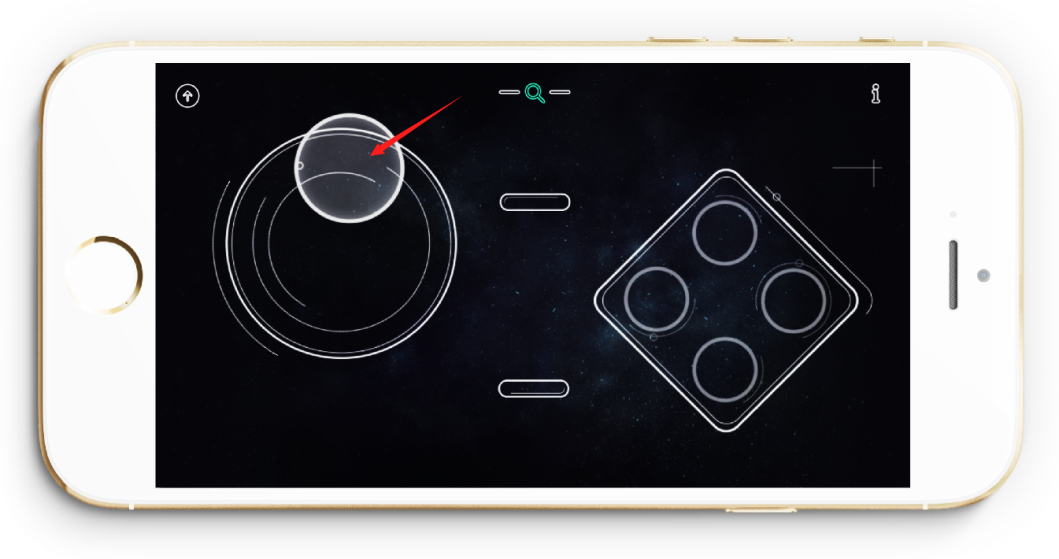
完成连接后，我们就可以测试控制器接收信号的情况了。回到Arduino IDE的界面，点击右上角Serial Monitor（串口监视器）。



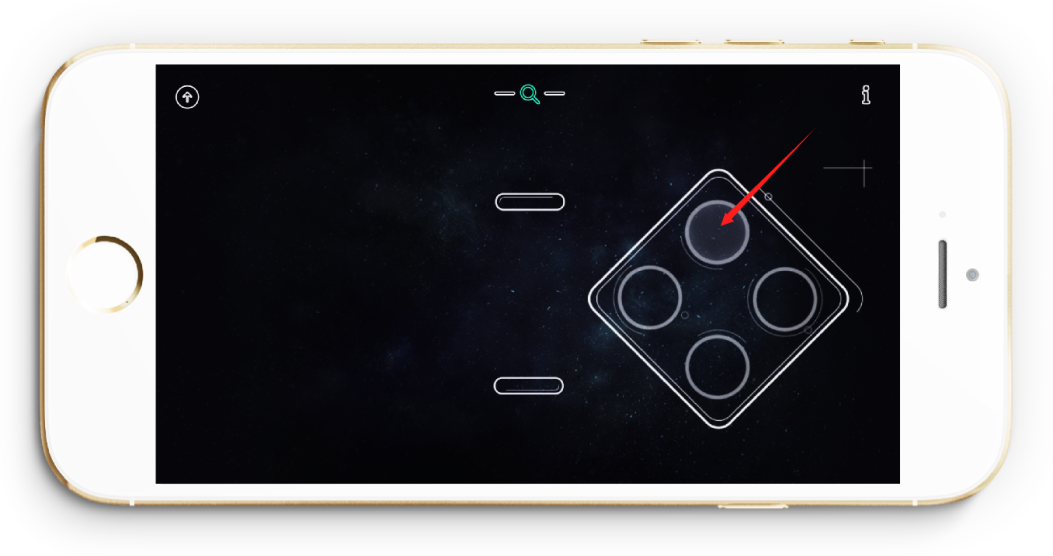
会弹出如下的对话框，先将串口的波特率设置为115200，就是下图红色圈出的部分。

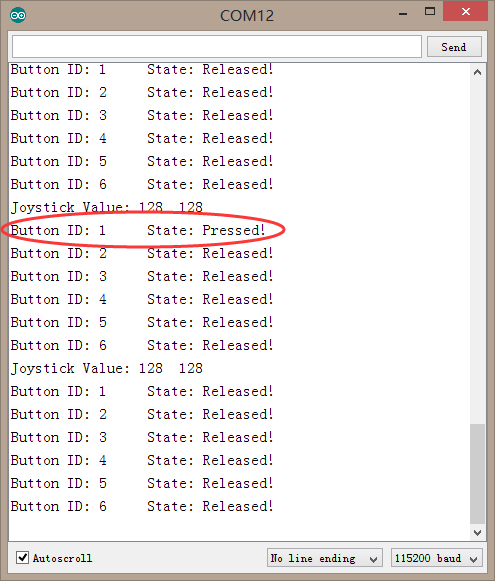


此时，拿出你的手机，任意的滑动摇杆，或者点击按钮。不出意外的话，你可以从串口监视器中看到，对应的数值，Joystick Value代表的是摇杆的数值。你会发现摇杆上下滑动时，Joystick Value的值随之发生变化，并且向上，值越大，向下，值越小。左右也是同样的道理。有发现吗？Button ID则对应显示你按下的哪个按键。







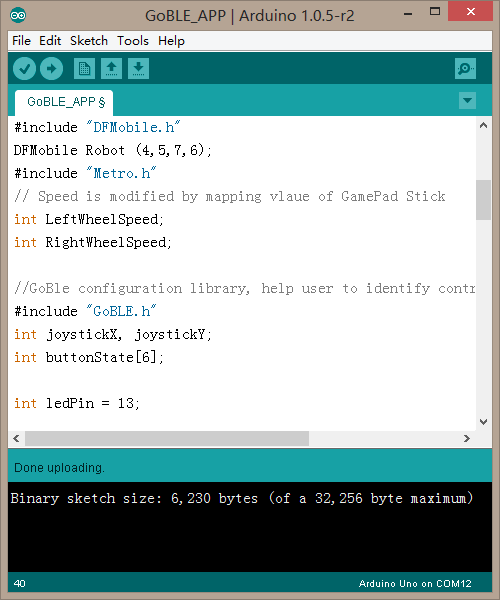


**输入遥控代码**

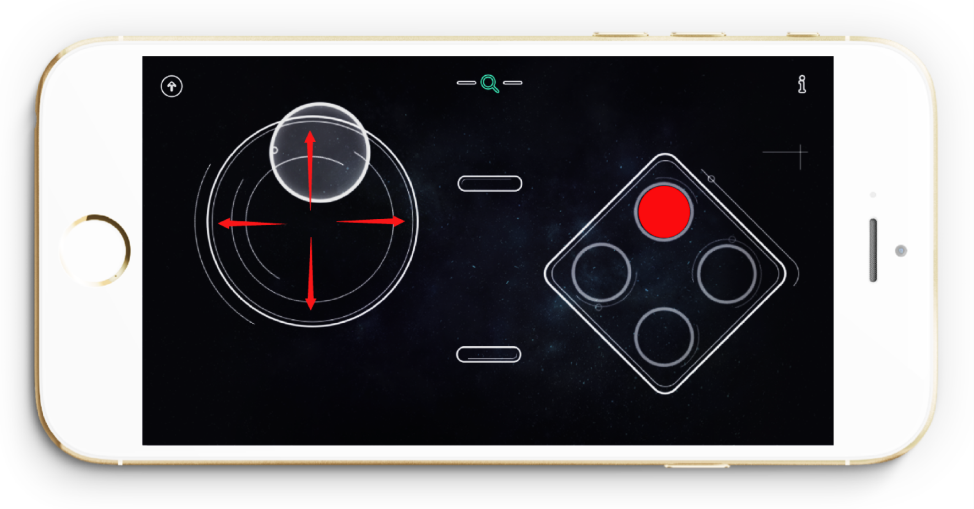
如果前面都很顺利的话，我们就可以进行最后的遥控代码的下载了。这里下载遥控代码时，有个很关键的一点需要注意。因为我们前面在测试阶段，已经将手机和Romeo BLE建立连接了。**蓝牙在连接时，是无法下载程序的。所以，需要将蓝牙断开连接后，方可下载重新下载程序。这点需要牢记！**

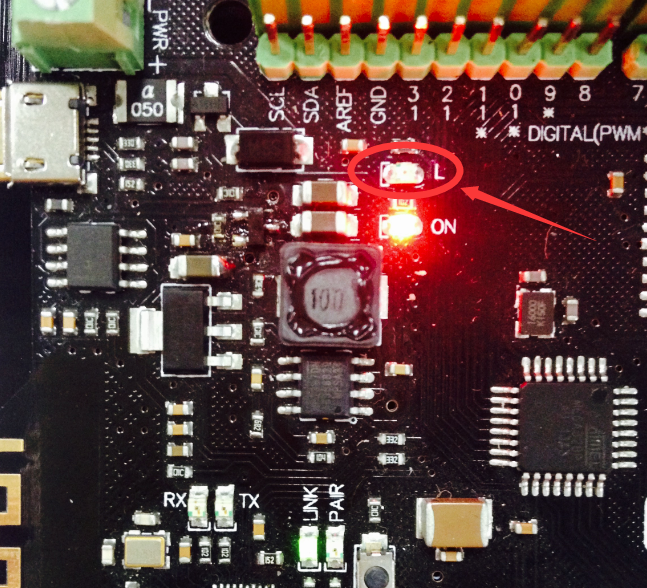
断开蓝牙连接的方法是：可以给Romeo BLE断电再重新上电，或者关掉手机蓝牙。

断掉蓝牙后，我们重新下载下面这段遥控代码，可在代码包中找到GoBLE\_APP.ino。



下载成功后，按下图箭头方向前后左右滑动摇杆，就可以通过手机来操控你的机器人。还可按下图中标出的按键，用来控制板载的13号引脚的LED的开关。





**代码回顾**

小车我们玩过了，现在就来简单学习下代码如何使用了。GoBLE\_Test测试代码我们就不单独说了，直接来看下GoBLE\_APP。

基础功能重复的就不多说了，直接说蓝牙遥控的部分。首先，这里用到两个库，Metro和GoBLE库。

#include "Metro.h"

#include "GoBLE.h"

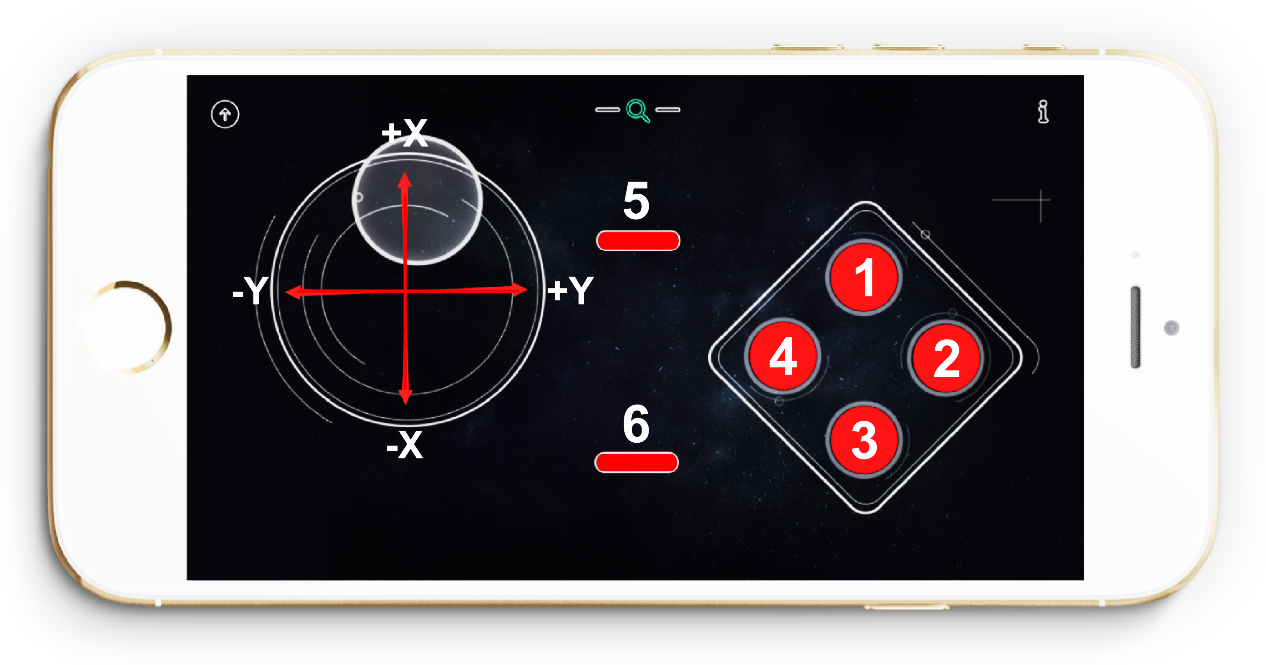
int joystickX, joystickY;

int buttonState[6];

joystickX，joystickY，buttonState[6]这三个变量是针对GoBLE库定义的。分别用来存储摇杆X轴,Y轴以及6个按键的状态值的。

下图就是各个按键的分布图。如果在测试代码中，有仔细看过数值变化过程的伙伴，也许能猜出左边摇杆图的中的+X,-X,+Y,-Y是什么意思了。

X,Y是摇杆的两个移动方向。“+，-”表示的数值的变化趋势。“+”代表数值递增的方向，“-”代表数值递减的方向。



Setup()函数中我们需要做一些初始化设置：

Goble.begin();

这句语句用来蓝牙初始化设置的。每次使用手机蓝牙的话，都必须用到这句语句。

Serial.begin(115200);

这个用法其实我们在前面测试阶段已经用到过了，是用来初始化串口的。如果我们想从串口监视器上看到数据，必须要在setup()函数中，begin一下Serial，并且设置一下它的速度(括号内的值)， **蓝牙波特率为115200。**

继续往下看，Goble.available()函数：

if(Goble.available()){

do something;

}

这个函数的含义是：如果接收到了蓝牙发过来的数值，那么执行些什么动作。

那么available()内部就是需要执行的动作了。首先，我们需要对接收到的数据做解析，下面两句就是读取X,Y轴上的数据。

joystickX = Goble.readJoystickX();

joystickY = Goble.readJoystickY();

// Serial.print("joystickX: ");

// Serial.print(joystickX);

// Serial.print("joystickY: ");

// Serial.println(joystickX);

下面的四句Serial.print()能猜出是什么意思呢？和串口有关。是用来串口打印出读到的数据，便于我们观察，调试。语句前面的”//”符号，表示这句话注释掉。代码在编译的时候，将不包含这四句话。简单的说 ，注释掉这四句话的话，我们无法从串口读到数据。

想了解更多，可查看[Arudino语法手册](http://wiki.dfrobot.com.cn/index.php/Arduino%E7%BC%96%E7%A8%8B%E5%8F%82%E8%80%83%E6%89%8B%E5%86%8C%EF%BC%88%E5%A4%9A%E9%A1%B5%E9%9D%A2%E7%89%88%EF%BC%89)。

buttonState[SWITCH\_UP] = Goble.readSwitchUp();

buttonState[SWITCH\_DOWN] = Goble.readSwitchDown();

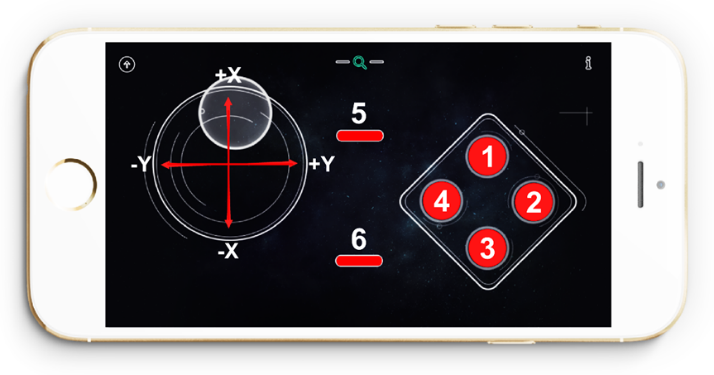
buttonState[SWITCH\_LEFT] = Goble.readSwitchLeft();

buttonState[SWITCH\_RIGHT] = Goble.readSwitchRight();

buttonState[SWITCH\_SELECT] = Goble.readSwitchSelect();

buttonState[SWITCH\_START] = Goble.readSwitchStart();

上面这一大串是用来读取按键的状态的。依次对应按键的分布图：



SWITCH\_UP (上键) -- 1

SWITCH\_RIGHT (右键) -- 2

SWITCH\_DOWN （下键） -- 3

SWITCH\_LEFT （左键） -- 4

SWITCH\_SELECT （选择键） -- 5

SWITCH\_START （开始键） -- 6

读完所有的数据以后，我们不能直接拿来用，需要做一些处理。

首先，需要将手柄读到的数据映射到我们小车的轮子转速上，所以就有了下面这两句。将读到的原始数据0~255映射为小车速度-255~255。

int SpeedX=2\*joystickX-256;

int SpeedY=2\*joystickY-256;

Serial.print("Speed: ");

Serial.print(SpeedX);

Serial.print(" ");

Serial.println(SpeedY);

同样是串口打印出速度，便于你调试使用，如果不需要的话，你也可以用”//”注释掉。

数据处理完之后，就需要让小车开始执行动作了。

如果 ( 摇杆向上 或者(||) 摇杆向下 ){

小车前进或者后退;

}

如果 ( 摇杆向右 或者(||) 摇杆向左){

小车右转或者左转;

}

如果 (摇杆X轴没变化 并且(&&) 摇杆Y轴没变化){

小车停止;

}

“翻译”成对应的代码就是下面这段：

if (SpeedX>200 || SpeedX<-200){

LeftWheelSpeed=SpeedX;

RightWheelSpeed=SpeedX;

Robot.Speed (LeftWheelSpeed,RightWheelSpeed);

}

else if (SpeedY>200 || SpeedY<-200){

LeftWheelSpeed=SpeedY-80;

RightWheelSpeed=-SpeedY-80;

Robot.Speed(LeftWheelSpeed,RightWheelSpeed);

}

else if (SpeedX==0 && SpeedY==0){

Robot.Speed(0,0);

}

最后一段是用来控制板载13号引脚的LED。

int ledPin = 13;

代码开始还是需要定义下13号引脚。

pinMode(ledPin,OUTPUT);

在setup()中将LED设置为输出模式，具体可以查看[pinMode()函数](http://wiki.dfrobot.com.cn/index.php/PinMode())的使用说明。

下面这段就是对按键读到的状态进行判断，因为我们只用到了按键1(上键)，如果被按下了(PRESSED)，点亮LED,也就是将LED引脚设置为HIGH。

if (buttonState[1] == PRESSED){

digitalWrite(ledPin,HIGH);

}

同样如果按键被释放(RELEASED),熄灭LED，也就是将LED引脚设置为LOW。

if (buttonState[1] == RELEASED){

digitalWrite(ledPin,LOW);

}

好了，代码解释就到这里了。这样一看，是不是代码也不是很难理解呀？可是尝试按照你的意愿来修改代码，比如换一个按键来控制小灯啦，或者给你小车装个手动转向灯等等。好玩的很多，当看到小车按你的要求在跑的话，一定会觉得非常有乐趣的！