项目三　互动交通信号灯

有没有试着做上面那个课后作业呢？做出来的话，说明你已经基本掌握上面所学的东西了，如果不会也没关系，我相信，看完这个章节，前面那个问题就不攻自破了！我们这回就基于上面这个交通灯来进行一个拓展，增加一种行人按键请求通过马路的功能。当按钮被按下时，Arduino 会自动反应，改变交通灯的状态，让车停下，允许行人通过。

这个项目中，我们开始要实现 Arduino 的互动了，也会在代码学习到如何创建自己的函数。这次的代码相对长一点，耐下心来，等看完这一章，相信你能收获不少！

我们之后在所需元件中将不再重复罗列以下三样，UNO、扩展板+面包板、跳线。但是！每次都还是需要用到的。

所需元件

- 2×　5mm LED 灯 
- 2×　5mm LED 灯 
- 1×　5mm LED 灯 
- 6×　220 欧电阻* 
- 1×　按钮 

这里 5 个 LED 灯，为什么会用到了 6 个电阻呢？我们知道 5 个电阻是 LED 的限流电阻。还有一个电阻是给按钮的，它叫做下拉电阻（我们后面会解释）。

硬件连接

按图 3-1 的连线图连接你的电路。特别要注意的是，这次连线比较多，注意不要插错。下图中，面包板上标出淡绿色的不是跳线，只是为了说明纵向的孔导通，避免你插错。给 Arduino 上电前认真检查你的接线是否正确。在连线时，保持电源是断开的状态，也就是没有插 USB 线。

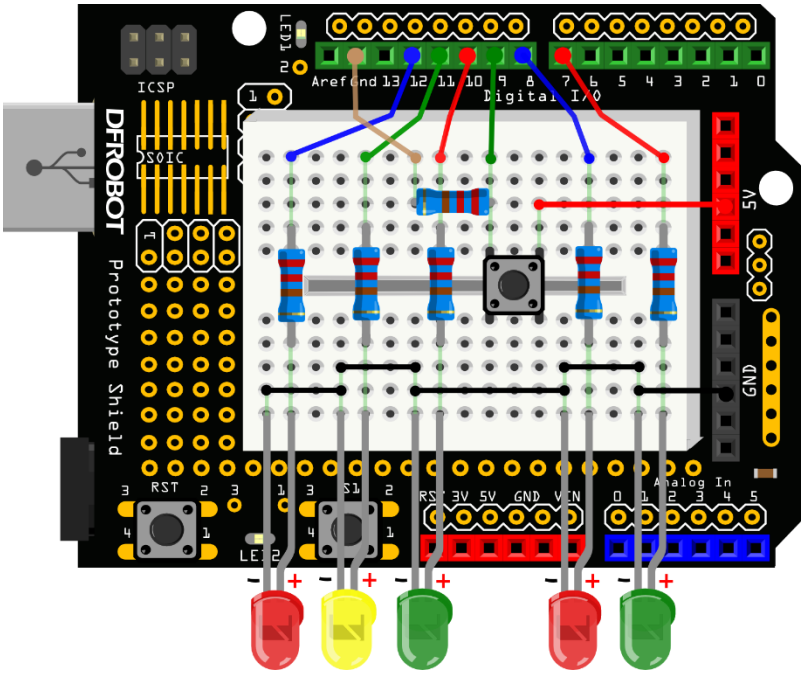


图 3-1 互动交通信号灯连线图

输入代码

输入下面的样例代码 3-1，这段代码引自《beginning-arduino》一书。

样例代码 3-1：

```
//项目三 -- 互动交通信号灯
int carRed = 12; //设置汽车灯
int carYellow = 11;
int carGreen = 10;
int button = 9; //按钮引脚
int pedRed = 8; //设置行人灯
int pedGreen = 7;
int crossTime = 5000;//允许行人通过的时间
unsigned long changeTime;//按钮按下后的时间

void setup() {
        //所有 LED 设置为输出模式
        pinMode(carRed, OUTPUT);
        pinMode(carYellow, OUTPUT);
        pinMode(carGreen, OUTPUT);
```

```
        pinMode(pedRed, OUTPUT);
        pinMode(pedGreen, OUTPUT);
        pinMode(button, INPUT); //按钮设置为输入模式
        digitalWrite(carGreen, HIGH); //开始时，汽车灯绿灯
        digitalWrite(pedRed, LOW);  //行人灯为红灯
}

void loop() {
        int state = digitalRead(button);
        //检测按钮是否被按下，并且是否距上次按下后有 5 秒的等待时间
        if(state == HIGH && (millis() - changeTime)> 5000){
                //调用变灯函数
                changeLights();
        }
}

void changeLights() {
        digitalWrite(carGreen, LOW); //汽车绿灯灭
        digitalWrite(carYellow, HIGH); //汽车黄灯亮
        delay(2000); //等待 2 秒

        digitalWrite(carYellow, LOW); //汽车黄灯灭
        digitalWrite(carRed, HIGH); //汽车红灯亮
        delay(1000); //为安全考虑等待 1 秒

        digitalWrite(pedRed, LOW); //行人红灯灭
        digitalWrite(pedGreen, HIGH); //行人绿灯亮

        delay(crossTime); //等待一个通过时间

        //闪烁行人灯绿灯，提示可过马路时间快到
        for (int x=0; x<10; x++) {
                digitalWrite(pedGreen, HIGH);
                delay(250);
                digitalWrite(pedGreen, LOW);
                delay(250);
        }
```

```
        digitalWrite(pedRed, HIGH);//行人红灯亮
        delay(500);

        digitalWrite(carRed, LOW); //汽车红灯灭
        digitalWrite(carYellow, HIGH); //汽车黄灯亮
        delay(1000);
        digitalWrite(carYellow, LOW); //汽车黄灯灭
        digitalWrite(carGreen, HIGH); //汽车绿灯亮

        changeTime = millis(); //记录自上一次灯变化的时间
        //返回到主函数循环中
}
```

下载完成后，可以尝试按下按钮。看看是个什么的效果？我们可以看到整个变化过程是这样的——开始时，汽车灯为绿灯，行人灯为红灯，代表车行人停。一旦行人，也就是你，按下按钮，请求过马路，那么行人灯就开始由红变绿，汽车灯由绿变黄，变红。在行人通行的过程中，设置了一个过马路的时间 crossTime，一旦到点，行人绿灯开始闪烁，提醒行人快速过马路。闪烁完毕，最终，又回到了开始的状态，汽车灯为绿灯，行人灯为红灯。

整段代码看起来很复杂，其实理清一下思路并不难。如果你还是没有办法理不清里面变化关系的话，可以试着画一个示意图，像项目 2 的课后作业那样，这样一来可能会方便你理解程序。

代码回顾

通过前面两个项目，你应该能够理解这个代码的大部分内容。代码开始是一串的变量的声明，在声明中，出现了一个新名词。这里就解释一下这个新名词：

```
unsigned long changeTime;
```

这是一个新的变量类型。我们之前，只创建过 int 整型变量，它可以存放一个-32768 到 32767 之间的整数。这次要创建的是一个 long 的变量类型，它可以存放一个-2147483648 到 2147483647 之间的整数。而 unsigned long 数据类型，则不存储负数，所以存储的范围就从 0 到 4294967295.

如果我们使用一个 int 型的话，信号灯状态变化的时间，它只能存储最大 32 秒（32768 毫秒约为 32 秒），一旦出现变量溢出就会造成程序运行出现错误，所以，为了避免这样的

情况，要选用能存储更大数的一个变量，并且不为负，我们就可以考虑使用 unsigned long 型。你可以用笔算下，这个变量最大能存储的数，时间可达 49 天。

变量这个盒子无限大吗？

那么，有人会问为什么有些变量类型可以存储很大的数，而有些变量类型不行呢？这是由变量类型所占的存储空间决定的。就拿我们前面讲变量的时候举过得例子，变量好像好比用来放东西的盒子，那不同类型的变量想象成不同大小的盒子，int 的盒子比 unsigned long 的盒子小，所以放的东西当然少啦！这样解释是不是比较容易理解这个概念呢？

那又有人问，设置不同大小的盒子干嘛呢？一样大不就行啦，都设置的大一点。理论上没有什么不可以的，可是我们不能忽略一个问题，那就是微控制器的内部存储容量是有限定的。电脑有内存，我们的微控制器同样有内存。像 Arduino UNO 板上的用的主芯片 Atmega328 最大内存是 32K。所以，我们要尽量的少用存储空间，能不用则不用。

表 3-1 列出了程序中可能用到的变量数据类型

数据类型	RAM	范围
boolean（布尔型）	1 byte	0 ~ 1（True 或 False）
char（字符型）	1 byte	-128 ~ 127
unsigned char（无符号字符型）	1 byte	0~255
int（整型）	2 byte	-32768 ~ 32768
unsigned int（整型）	2 byte	0 ~ 65535
long（长整型）	4 byte	-2147483648 ~ 2147483647
unsigned long（无符号长整型）	4 byte	0 ~ 4294967295
float（单精度浮点型）	4 byte	-3.4028235E38 ~ 3.4028235E38
double（单精度浮点型）	4 byte	-3.4028235E38 ~ 3.4028235E38

从上面表格可以看到，变量的类型有很多，不同的数对应不同的变量，int 和 long 是针对整数变量，char 是针对字符型变量，而 float，double 是针对含有小数点的变量。

随即进入 setup()函数，对 LED 和按钮进行一些设置，在设置时，需要注意到的是：

```
pinMode(button, INPUT);
```

pinMode()函数我们已经很熟悉了，在项目一的时候就介绍过，只是和 LED 有所不同的是，按钮要设置为 INPUT。

在 setup()函数中，先给定行人灯和汽车灯的一个初始状态：

```
digitalWrite(carGreen, HIGH); //开始时，汽车灯绿灯
digitalWrite(pedRed, LOW);    //行人灯为红灯
```

进入到的主程序中的第一句，就是来检测 button（引脚 9）的状态的：

```
int state = digitalRead(button);
```

此时，一个新函数出现了——digitalRead()!

函数格式如下：

digitalRead(pin)

这个函数是用来读取数字引脚状态，HIGH 还是 LOW（其实 HIGH 还有一种表达就是“1”，LOW 是“0”，只是 HIGH/LOW 更直观）。函数需要一个传递参数--pin，这里需要读取是按键信号，按键所在引脚是数字引脚 9，由于前面做了声明，所以这里用 button。

并且把读到的信号传递给变量 state，用于后面进行判断。state 为 HIGH 或者说为 1 时，说明按键被按下了。state 为 LOW 或者 0，表明按键没被按下。

所以，可以直接检查 state 的值来判断按钮是否被按下：

```
if(state == HIGH && (millis() - changeTime)> 5000) {
            //调用变灯函数
            changeLights();
}
```

这里涉及新的语句-- if 语句。if 语句是一种条件判断的语句，判断是否满足括号内的条件，如满足则执行花括号内的语句，如不满足则跳出 if 语句。

if 语句格式如下：

```
if(表达式){
    语句;
}
```

表达式是指我们的判断条件，通常为一些关系式或逻辑式，也可是直接表示某一数值。如果 if 表达式条件为真，则执行 if 中的语句。表达式条件为假，则跳出 if 语句。

我们代码中，第一个条件是 state 变量为 HIGH。如果按键被按下，state 就会变为 HIGH。第二个条件是 millis()的值减 changeTime 的值大于 5000。这两个条件之间有个“&&”符号。这是一种逻辑运算符，表示的含义是两者同时满足。

```
(millis() - changeTime)> 5000)
```

millis()是一个函数，该函数是 Arduino 语言自有的函数，它返回值是一个时间，Arduino 开始运行到执行到当前的时间，也称之为机器时间，就像一个隐形时钟，从控制器开始运行的那一刻起开始计时，以毫秒为单位。变量 changeTime 初始化时，不存储任何数值，只有在 Arduino 运行之后，将 millis()值赋给它，它才开始有数值，并且随着 millis()值变化而变化。通过 millis()函数不断记录时间，判断两次按键之间的时间是不是大于 5 秒，如果在 5 秒之内不予反应。这样做的目的是，防止重复按键而导致的运行错误。

逻辑运算符

前面说到的&&是一个逻辑运算符，常用的逻辑运算符有：

- && —— 逻辑与 （两者同时满足）
- || —— 逻辑或 （两者其中一个满足）
- ！ —— 逻辑非 （取反，相反的情况）

if 语句内只有一个函数：

```
changeLights();
```

这是一个函数调用的例子。该函数单独写在了 loop()函数之外。我们需要使用的时，直接写出函数名就可以实现调用了。该函数是 void 型，所以是无返回值、无传递参数的函数。当函数被调用时，程序也就自动跳到它的函数中运行。运行完之后，再跳回主函数。需要特别注意的：函数调用时，函数名后面的括号不能省，要和所写的函数保持一致。changeLights() 函数内部就不做说明了。

硬件回顾

按键开关

按键一共有 4 个引脚，图 3-2 分别显示了正面与背面。而图 3-3 则说明了按键的工作原理。一旦按下后，左右两侧就被导通了，而上下两端始终导通。

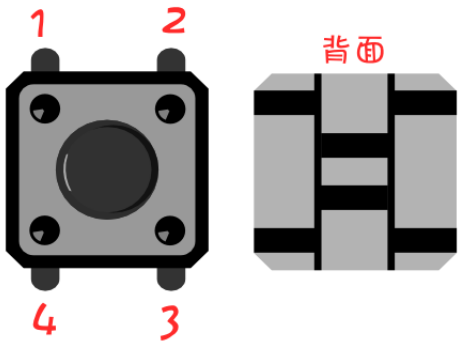


图 3-2 按钮结构图

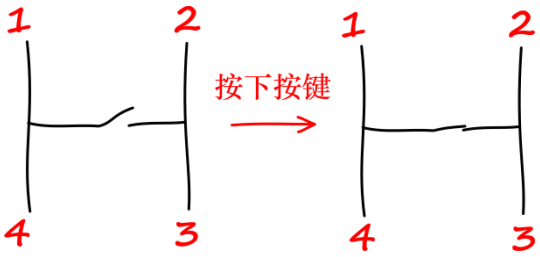


图 3-3 按钮原理图

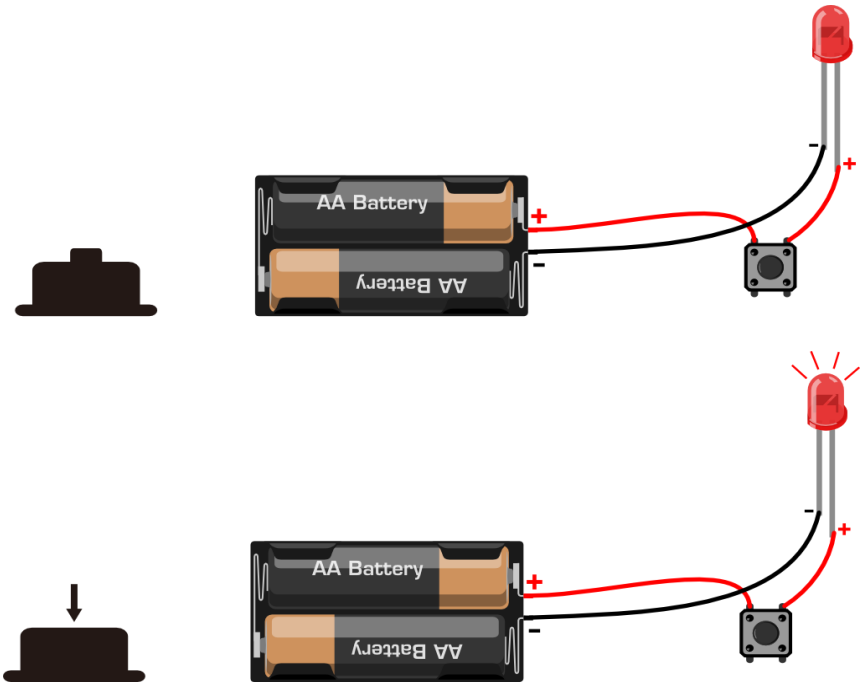


图 3-4 按钮示意图

图 3-4 传达的意思是，按钮就是起到一个通断的作用。在我们这个项目中，按钮控制数字引脚是否接高（接 5V）。按下的话，数字引脚 9 就能检测到为高电平。否则就是保持一个低电平的状态（接 GND）。

什么是下拉电阻？

下拉电阻这个名词可能比较抽象，就从字的含义着手，“下拉”我们就理解为把电压往下拉，降低电压。

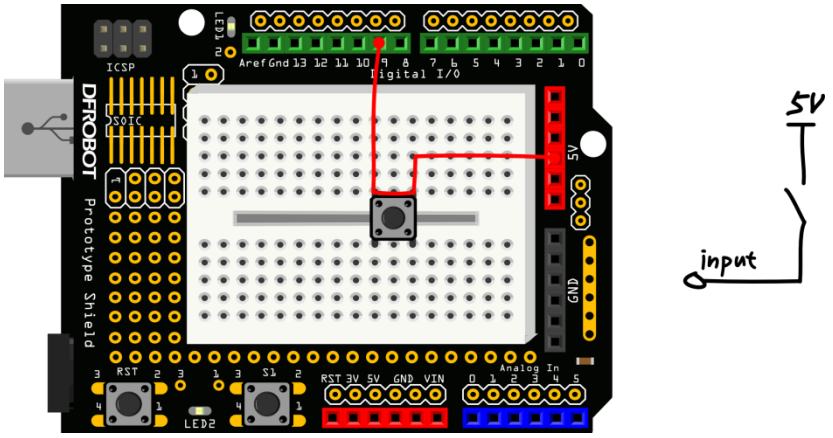


图 3-5 未接下拉电阻

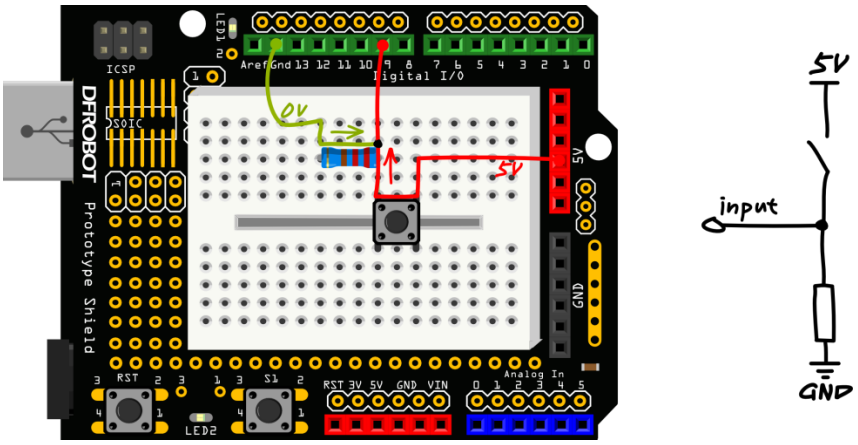


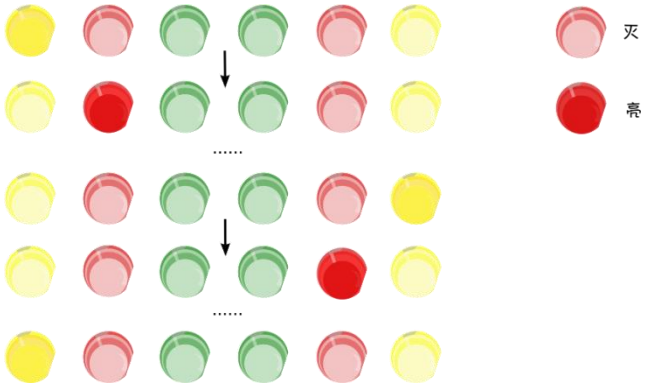
图 3-5 有下拉电阻

按键作为开关。当输入电路状态为 HIGH 的时候，电压要尽可能接近 5V。输入电路状态为 LOW 的时候，电压要尽可能接近 0V。如果不能确保状态接近所需电压，这部分电路就会产生电压浮动。所以，我们在按钮那里接了一个电阻来确保一定达到 LOW，这个电阻就是所谓下拉电阻。

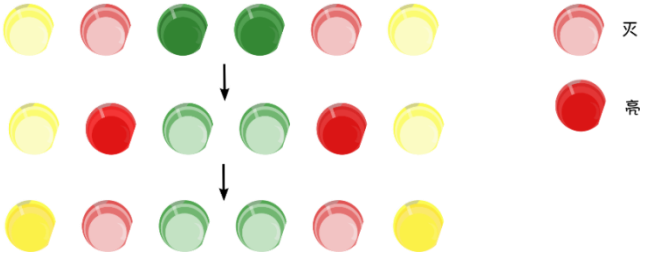
可以从上面两张图看到，第一张是未接下拉电阻的电路，按键没被按下时， input 引脚就处于一个悬空状态。空气会使该引脚电压产生浮动，不能确保是 0V。然而第二张是接了下拉电阻的电路，当没被按下时，输入引脚通过电阻接地，确保为 0V，不会产生电压浮动现象。

课后作业

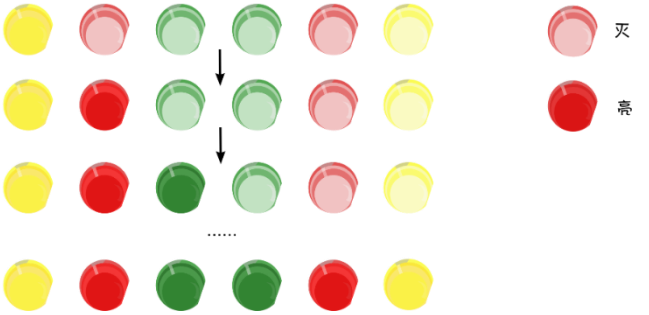
（1） 选择任意颜色 LED 6 个，做一个流水灯的效果，6 盏灯从左至右依次点亮，然后再从右至左依次熄灭。



（2） 如果上面那个你已经完成了的话，可以尝试一下，先从中间的灯开始亮起，依次向两边扩开。下图是个变换过程的示意图。



（3）再比如，从左至右，依次亮起 1 个，2 个，3 个……



（4）再结合按钮，用按键开关和 LED 互动。（提供供参考教程）

① 用一个按键，按一下控制灯亮，再按一下控制灯灭。

http://www.dfrobot.com.cn/community/forum.php?mod=viewthread&tid=1395&extra=page%3D2

②又或者用两个按键，一个控制灯亮，另一个控制灯灭。

http://learn.adafruit.com/adafruit-arduino-lesson-6-digital-inputs?view=all

玩儿法有很多，就全靠你的想象了!