项目十二 遥控灯

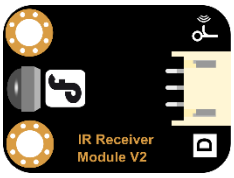
我们知道家里的那些遥控器,不管是电视还是空调都是通过红外来控制的.我们这次也通过红外来做个遥控灯。本章中，设定遥控器的“红色电源键”来控制 LED 的开关，当然看完这一节后，你也可以用其他的按钮来代替。

在开始遥控灯之前，我们先来个预热实验，通过串口来了解下如何使用红外接收管和遥控器。

预热实验:

所需材料

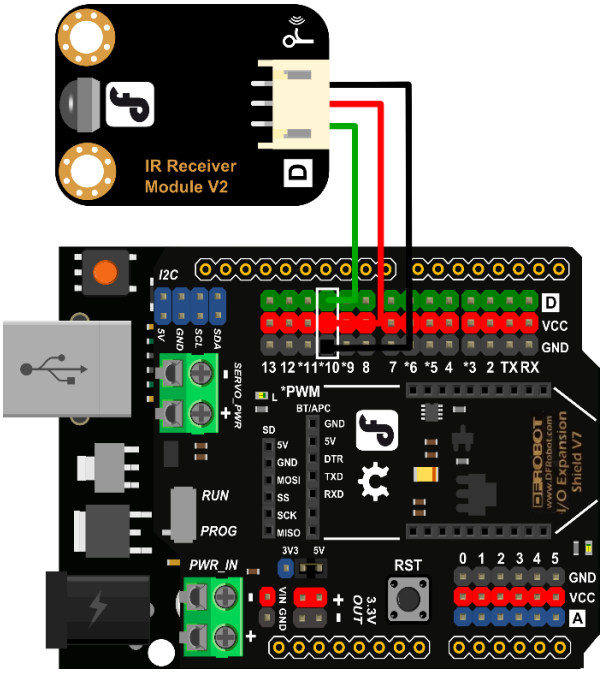
- 1× 数字红外接收模块



- 1× Mini 遥控器

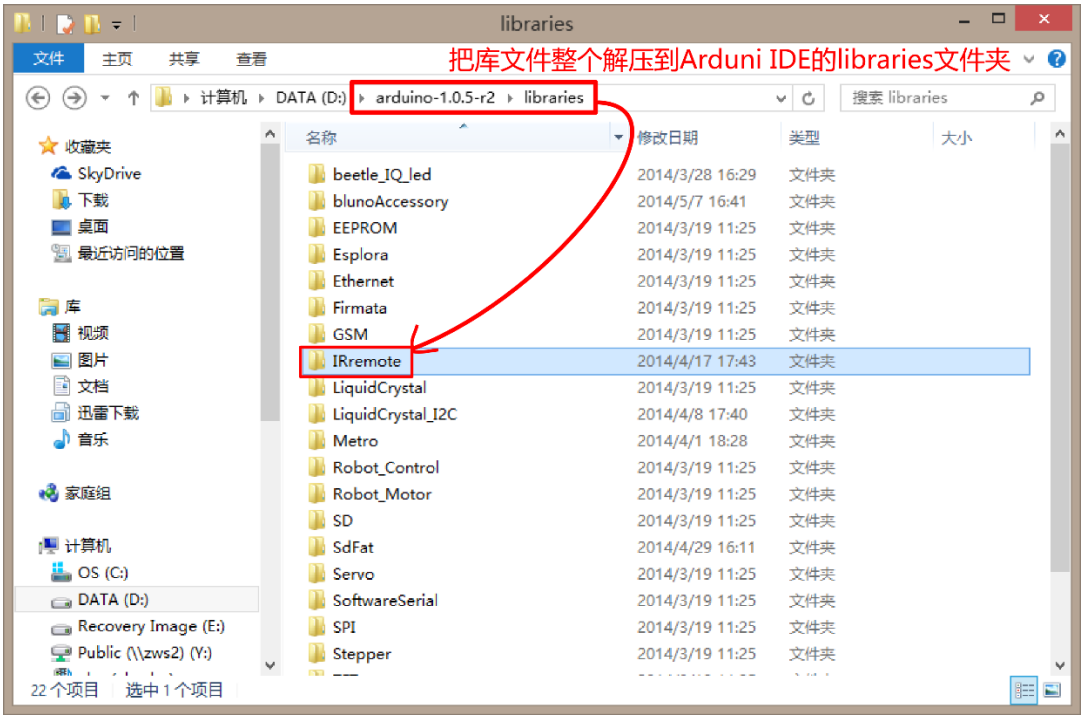
硬件连接

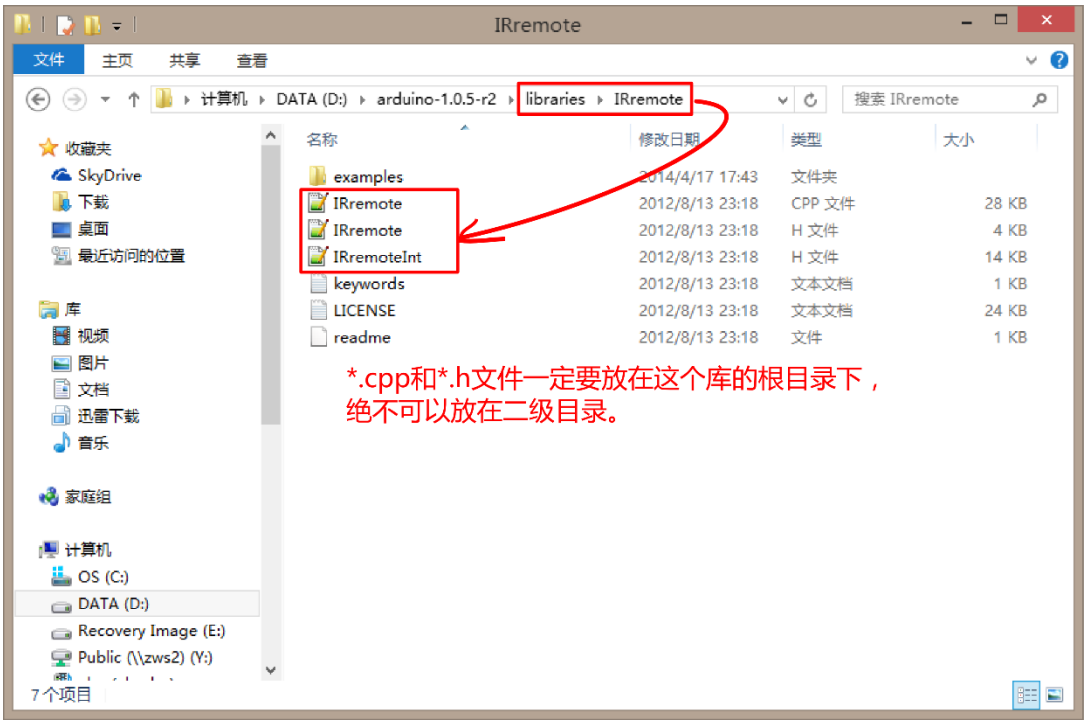
数字红外接收模块 → 数字口 10



输入代码

这段代码，你可以不用自己手动输入，我们提供现成的 IRremote 库，在我们的*教程代码*文件夹中的 *Lesson12_1* 中，把整个库的压缩包解压到 Arduino IDE 安装位置 Arduino 1.0.5/ libraries 文件夹中，。如下图所示。



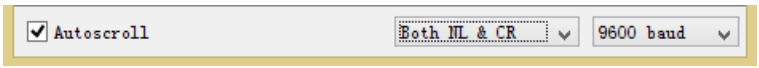


直接运行 Example 中的 IRrecvDemo 代码即可。

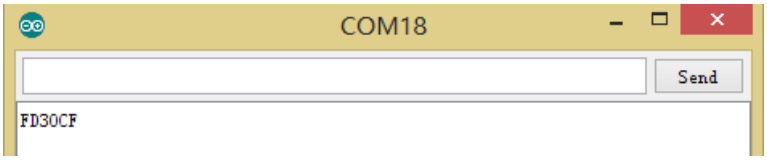
样例代码 12-1:

```
//这段代码来自 IRremote 库中 examples 中的 IRrecvDemo
//项目十二 - 红外接收管
#include <IRremote.h>        //调用 IRremote.h 库
int RECV_PIN = 10;           //定义 RECV_PIN 变量为 10
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
void setup(){
  Serial.begin(9600);          //串口波特率设为 9600
  irrecv.enableIRIn();         //启动红外解码
}
void loop() {
    //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)) {
       //接收到的数据以 16 进制的方式在串口输出
       Serial.println(results.value, HEX);
       irrecv.resume();  // 继续等待接收下一组信号
  }
}
```

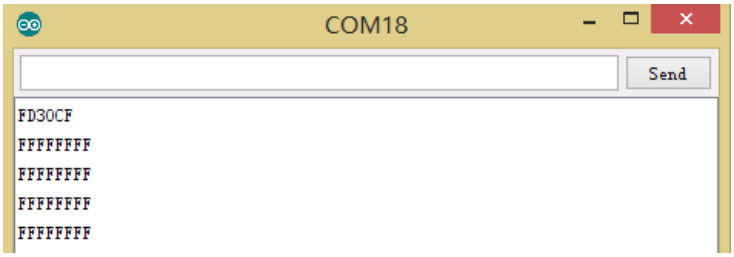
下载完成后，打开 Arduino IDE 的串口监视器（Serial Monitor），设置波特率 baud 为 9600，与代码中 Serial.begin(9600)相匹配。



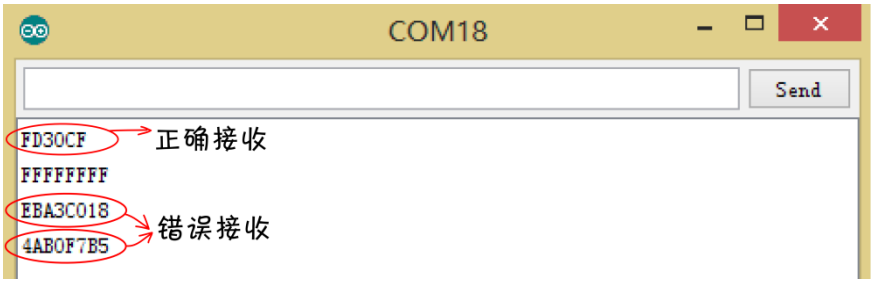
设置完后，用 Mini 遥控器的按钮对着红外接收管的方向，任意按个按钮，我们都能在串口监视器上看到相对应的代码。如下图所示，按数字“0”，接收到对应 16 进制的代码是 FD30CF。每个按钮都有一个特定的 16 进制的代码。



如果按住常按一个键不放就是出现“FFFFFFFF”。



在串口中，正确接收的话，应该收到以 FD-开头的六位数。如果遥控器没有对准红外接收管的话，可能会接收到错误的代码。如我们下图所示：

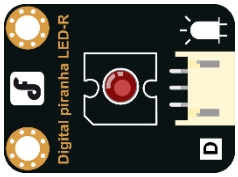


上面这段代码我们没有像以前一样一步一步做详细说明，原因就是由于红外解码较为复杂，所幸的是，高手把这些难的工作已经做好了，提供给我们这个 IRremote 库，我们只需要会用就可以了，先不需要弄明白函数内部如何工作的。要用的时候，把代码原样搬过来就好了。先用起来再说~

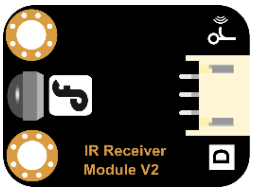
预热完之后，我们言归正传，开始制作遥控灯。

所需材料

- 1×　数字食人鱼红色 LED 发光模块



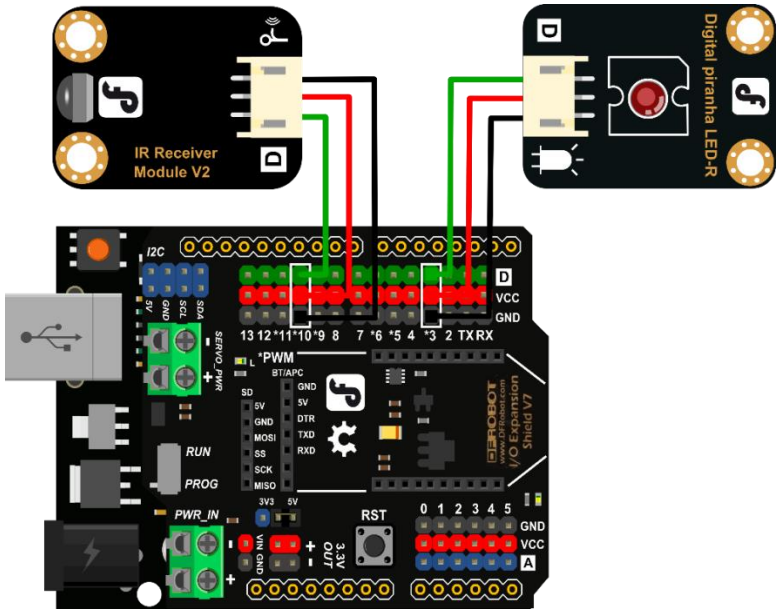
- 1×　数字红外接收模块



- 1×　Mini 遥控器

硬件连接

其实就是在原有的基础上，加了个 LED，LED 使用的是数字引脚 10。红外接收管仍然接的是数字引脚 3。



输入代码

这里不建议一步一步输入代码，可以在原有的代码上进行修改，观察下相对前一段代码增加了哪些内容。

样例代码 12-2:

```
#include <IRremote.h>
int RECV_PIN = 10;
int ledPin = 3;                // LED - digital 3
boolean ledState = LOW;         // ledstate 用来存储 LED 的状态
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(ledPin,OUTPUT);       // 设置 LED 为输出状态
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);

    //一旦接收到电源键的代码，LED 翻转状态，HIGH 变 LOW，或者 LOW 变 HIGH
    if(results.value == 0xFD00FF){
        ledState = !ledState;            //取反
        digitalWrite(ledPin,ledState);    //改变 LED 相应状态
     }
    irrecv.resume();
  }
}
```

代码回顾

程序一开始还是对红外接收管的一些常规定义，按原样搬过来就可以了。

```
#include <IRremote.h>     //调用 IRremote.h 库
int RECV_PIN = 10;        //定义 RECV_PIN 变量为 10
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 ledPin = 3;             // LED - digital 3
boolean ledState = LOW;      // ledstate 用来存储 LED 的状态
```

在这里，我们多定义了一个变量 ledState，通过名字应该就可以看出来含义了，用来存储 LED 的状态的，由于 LED 状态就两种（1 或者 0）,所以我们使用 boolean 变量类型。

setup()函数中，对使用串口，启动红外解码，数字引脚模式进行设置。

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，判断是否接收到了电源键的红外码。

```
if(results.value == 0xFD00FF)
```

第二件事，就是让 LED 改变状态。

```
ledState = !ledState;            //取反
digitalWrite(ledPin,ledState);    //改变 LED 相应状态
```

这里可能对“!”比较陌生，“!”是一个逻辑非的符号，“取反”的意思。我们知道“!=”代表的是不等于的意思，也就是相反。这里可以类推为，!ledState 是 ledState 相反的一个状态。“!”只能用于只有两种状态的变量中，也就是 boolean 型变量。

最后，继续等待下一组信号。

```
irrecv.resume();
```