BAIJIA DAGU
百佳大谷

创客模块套件

使用说明书

中山市百佳大谷电子科技有限公司

目 录

震动模块

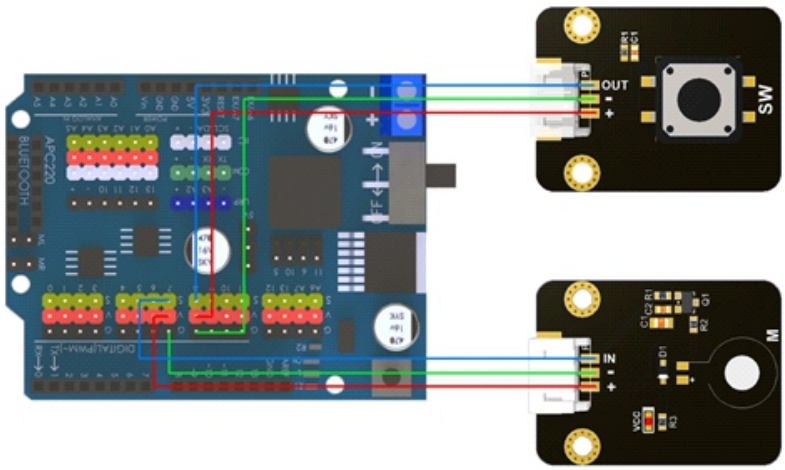
模块介绍：可以直接通过数字口进行控制，可通过PWM控制马达的震动强度。通过此模块可以方便的完成电信号到机械震动感的转换，适用于震动感互动产品的制作，可穿戴智能设备震动提醒等。

输入类型：数字输入，PWM输入。
接口模式：pH2.0-3P

引脚定义：

IN	数字输入、PWM 输入
-	GND
+	VCC

Arduino uno
样例一
连接图

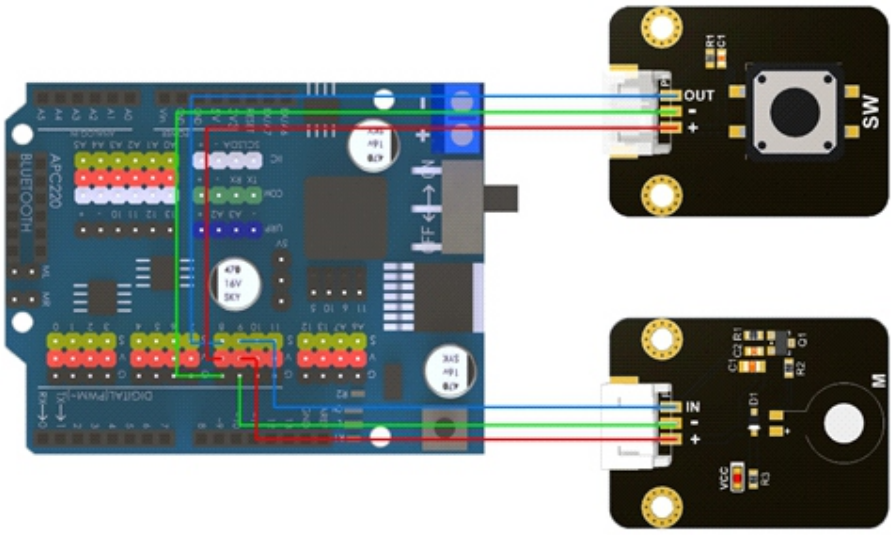


代码

```
int btnpin=8;
int movepin = 7;
void setup() {
   // put your setup code here, to run once:
   pinMode(movepin,OUTPUT);
   pinMode(btnpin,INPUT);
}

void loop() {
   // put your main code here, to run repeatedly:
   if(digitalRead(btnpin)){
      digitalWrite(movepin,HIGH);
   }
   else{
      digitalWrite(movepin,LOW);
   }
}
```

结果：按键接入引脚8，震动模块接入引脚7。当按键按下时，模块震动，反之，模块不震动。

样例二
接线图



代码

```
int movePin = 9;
int buttonpin = 8;
byte value= 0;
int oldstate =0;
int state = 0;
int increase =15;
void setup() {
  pinMode(movePin,OUTPUT);
  pinMode(buttonpin,INPUT);
  Serial.begin(9600);
}
void loop(){
  state = digitalRead(buttonpin);
  if(value == 255){
       increase =-15;
    }
  if(value ==0){
       increase =15;
    }
  if(state == 1 && oldstate == 0){
       value += increase;
  }
  oldstate = state;
  analogWrite(movePin,value);
  Serial.println(value);
}
```

结果：按键接入引脚8，震动模块接入引脚9。每当按键按下一次时，模块震动强度以15的幅度变化。当强度到达最大值时，再按下按键时，模块震动强度减弱15；当强度到达最小值时，再按下按键时，模块震动强度增强15。

雨滴传感器模块

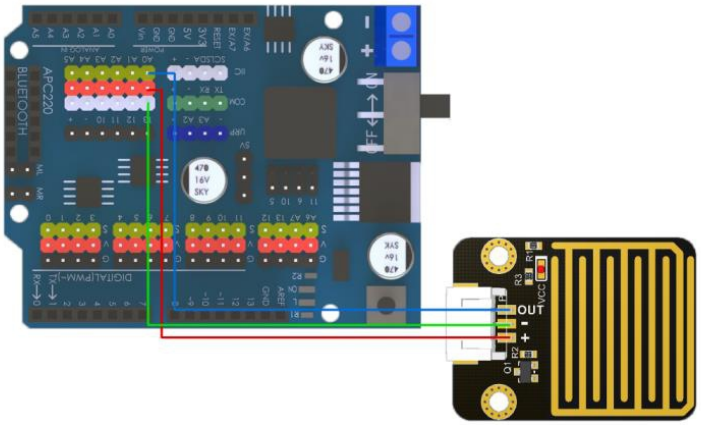
模块介绍：雨滴传感器是一个模拟输入模块，也叫雨水、雨量传感器。可用于各种天气状况的监测，检测是否下雨及雨量的大小，转成模拟信号输出。也可用于汽车自动刮水系统、智能灯光系统和智能天窗系统等。

输入类型：模拟输入·。
接口模式：pH2.0-3P

引脚定义：

Out	模拟输入
-	GND
+	VCC

Arduino
例子一
接线图

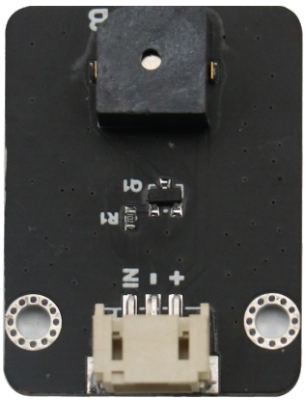


代码

```
void setup(){
   pinMode(A0,INPUT);
Serial.begin(9600);
}

void loop(){
Serial.println(analogRead(A0));
delay(100);
}
```

结果：模块接入引脚A0，打开串口，向渡镍处理表面滴水，会发现串口上的数值变大，越多水滴，数值越大。

有源蜂鸣器模块



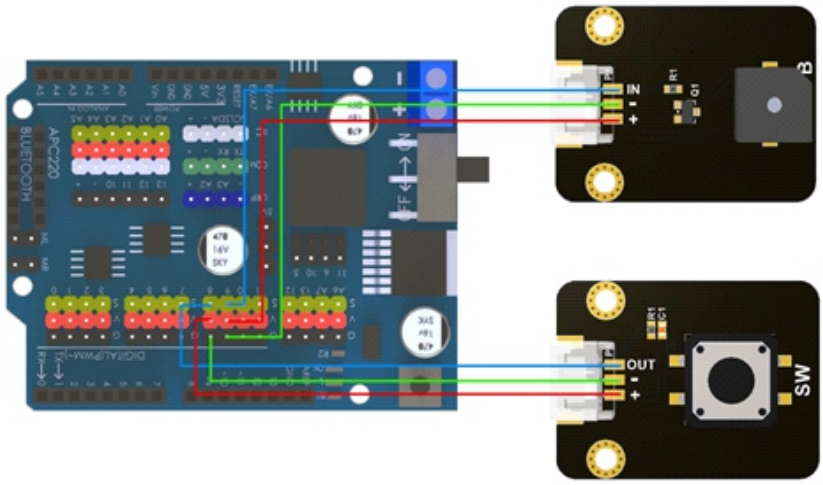
模块介绍：蜂鸣器是一种音频信号装置。作为一种集成结构的电子蜂鸣器，采用直流电压供电的，广泛应用于计算机、打印机、复印机、警报器、电子玩具、汽车电子设备、电话、计时器和其他电子产品语音设备。蜂鸣器按其驱动方式可分为有源蜂鸣器和无源蜂鸣器。
有源蜂鸣器和无源蜂鸣器的区别：这里的源不是指电源，而是指震荡源，也就是，有源蜂鸣器内部带有震荡源，所以只要通电就会叫，而无源蜂鸣器内部不带震荡源，所以用直流信号是无法令其鸣叫的。
此处讲解的蜂鸣器是有源蜂鸣器。只要有电源。我们输出高低电平交替，使蜂鸣器鸣响。

工作电压：3.3V-5V
输入类型：数字信号。
接口模式：pH2.0-3P
模块尺寸：35*26.3mm
模块重量：47g

引脚定义：

IN	数字输入
+	GND
-	VCC

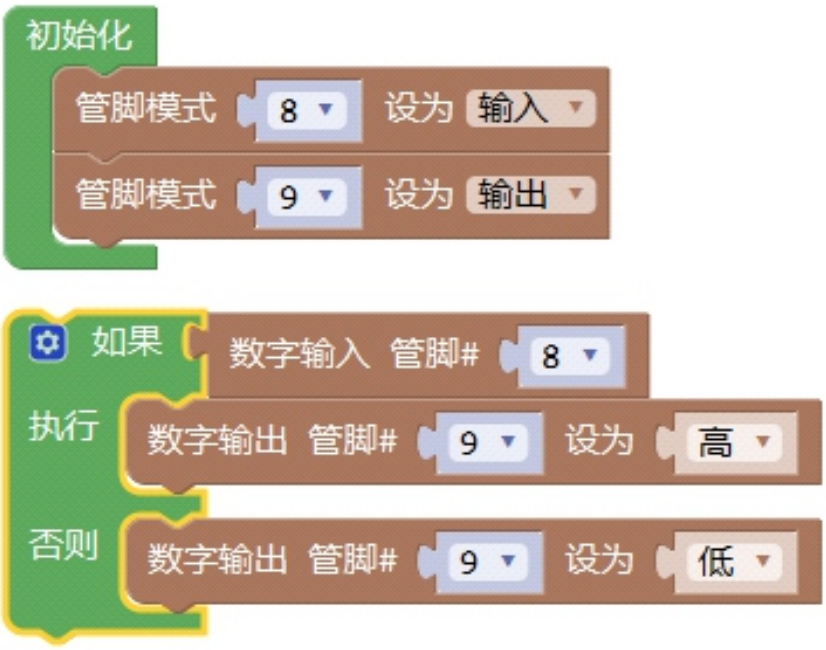
Arduino uno
样例一
接线图



代码

```
void setup() {
   pinMode(8,INPUT);   //定义按键引脚为 8，并为输入引脚
   pinMode(9,OUTPUT); //定义有源蜂鸣器引脚为 9，并为输出引脚
}
void loop() {
   int state = digitalRead(8);//读取按键输入值
   if(state){  //  检查输入是否为高，这里高为按下
      digitalWrite(9,HIGH);//蜂鸣器响起状态
   }
   else{
      digitalWrite(9,LOW);//蜂鸣器关闭状态
   }
}
```

Mixly图形化编程



结果：按键接入引脚8，有源蜂鸣器接入引脚9，当按下按键时，蜂鸣器发出响声，释放按键时，蜂鸣器不发出响声。

循迹模块

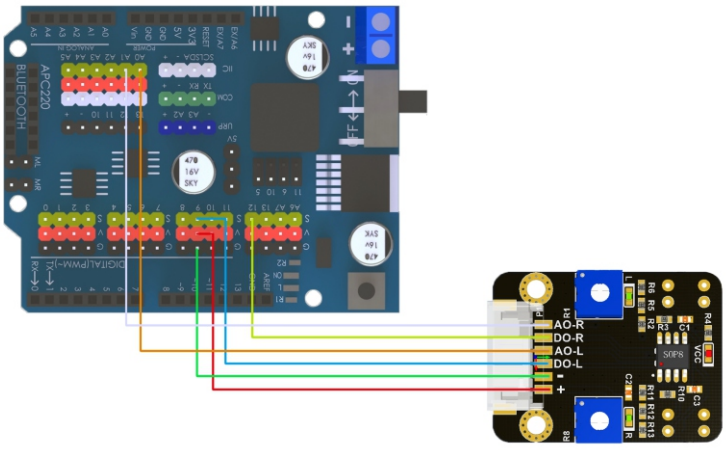
模块介绍：循迹模块是红外发射管发射光线到路面，红外光遇到黑色时则被吸收，接收管没有接收到反射光，输出高电平。当红外光遇到其他颜色时，接收管接收到反射光，输出低电平。该模块在使用模拟输出的时候，可以当作灰度传感器，灰度传感器是模拟传感器，可以感知地面或桌面不同的颜色而产生相应的信号。

工作电压：3.3V到5V
输入类型：数字信号·、模拟信号
接口模式：pH2.0-6P
模块尺寸：35*26.3mm
模块重量：51g

引脚定义：

AO-R	左边模拟输入
DO-R	右边数字输入
AO-L	左边模拟输入
DO-L	左边数字输入
-	GND
+	VCC

Arduin uno
例子一
接线图



代码

```
int data[2];
void setup() {
    pinMode(9,INPUT);//LD
    pinMode(12,INPUT);//RD
    pinMode(A0,INPUT);//LA
    pinMode(A1,INPUT);//RA
    Serial.begin(9600);
}

void loop() {
    distanceonD();
}

void distanceonD(){
    data[0] = digitalRead(9);
    data[1] = digitalRead(12);
    if(data[0] && data[1]){
        Serial.println("stop");
    }
    if(!data[0] && !data[1]){
        Serial.println("go");
    }
    if(!data[0] && data[1]){
        Serial.println("right");
    }
    if(data[0] && !data[1]){
        Serial.println("left");
    }
}
```

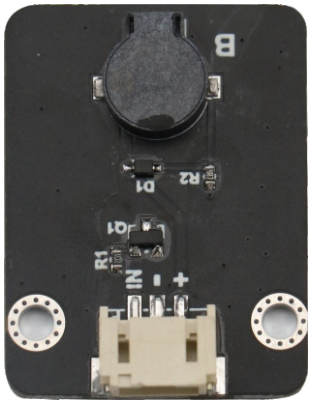
结果：当两边红外被阻挡时，接收不到信号，输出高电平，提示“stop”；当两边红外都没被阻挡时，接收到信号，提示”go”；当左边红外被阻挡时，左边接收不到信号，提示“left”；当右边红外被阻挡时，右边接收不到信号，提示“right”。
提示：尝试用螺丝刀扭一下模块的L和R上的十字口，在测试上例，增大减小检测物和被检测物之间的距离，观察看看串口输出的内容和小车运动有什么变化。

例子二
代码

```
int data[2];
void setup() {
   pinMode(A0,INPUT);//LA
   pinMode(A1,INPUT);//RA

   Serial.begin(9600);
}

void loop() {
   distanceonA();
}

void distanceonA(){
   data[0] = analogRead(9);
   data[1] = analogRead(12);

   if(data[0]>500 && data[1]>500){
      Serial.println("stop");
   }
   if(data[0]<500 && data[1]<500 ){
      Serial.println("go");
   }
   if(data[0]<500   && data[1]>500 ){
      Serial.println("right");
   }
   if(data[0]>500   && data[1]<500 ){
      Serial.println("left");
   }
}
```

结果：当红外循迹模块利用模拟信号传入信号时，此时可当该模块看作为灰度传感器。当两边灰度值大于500时，提示“stop”；当两边灰度值小于500时，提示“go”；当左边灰度值大于500，右边灰度值小于500时，提示“left”；当左边灰度值小于500，右边灰度值大于500时，提示“right”；（代码中的500可根据实际情况做成调整）

无源蜂鸣器模块



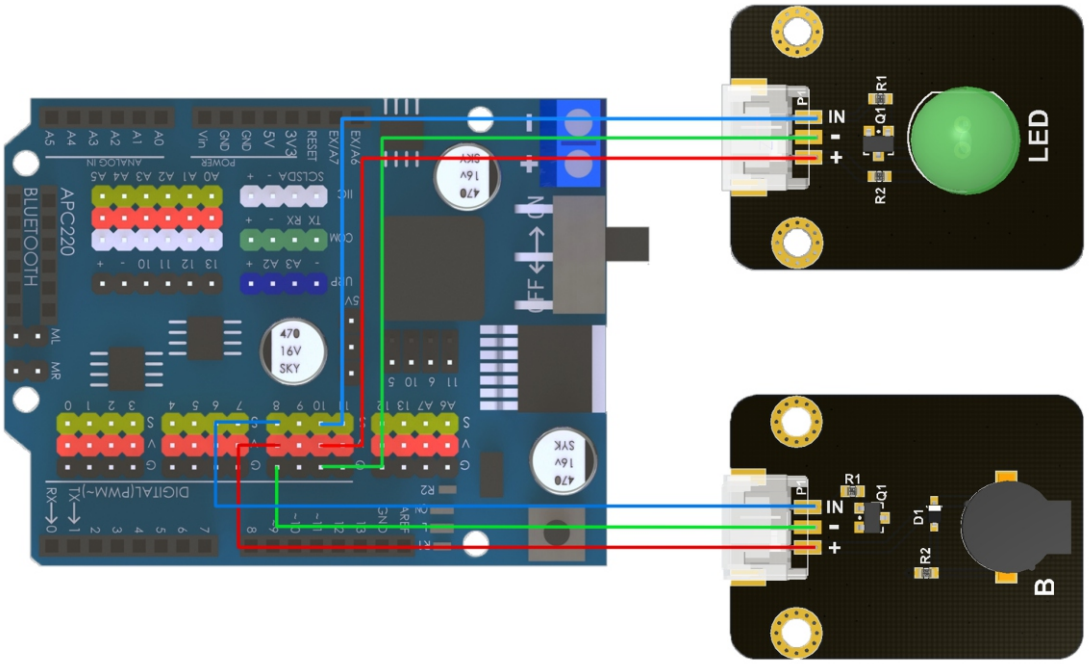
模块介绍：蜂鸣器是一种音频信号装置。作为一种集成结构的电子蜂鸣器，采用直流电压供电的，广泛应用于计算机、打印机、复印机、警报器、电子玩具、汽车电子设备、电话、计时器和其他电子产品语音设备。蜂鸣器按其驱动方式可分为有源蜂鸣器和无源蜂鸣器。
有源蜂鸣器和无源蜂鸣器的区别：这里的源不是指电源，而是指震荡源，也就是，有源蜂鸣器内部带有震荡源，所以只要通电就会叫，而无源蜂鸣器内部不带震荡源，所以用直流信号是无法令其鸣叫的。
此处讲解的蜂鸣器是有源蜂鸣器。只要有电源。我们输出高低电平交替，使蜂鸣器鸣响。

工作电压：3.3V-5V
输入类型：数字信号。
接口模式：pH2.0-3P
模块尺寸：35*26.3mm
模块重量：47g

引脚定义：

IN	数字输入
+	GND
-	VCC

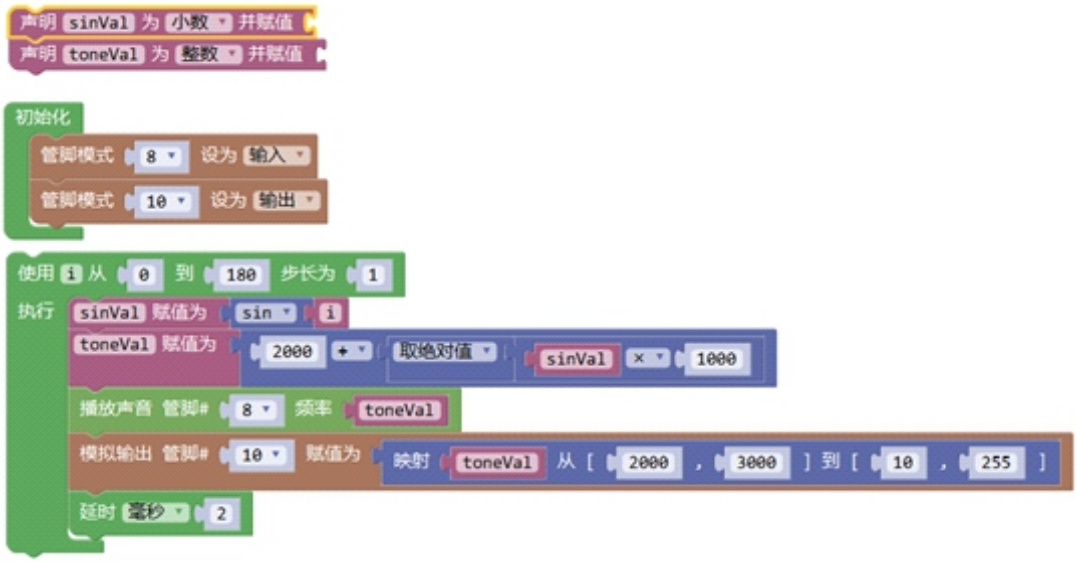
Arduino uno
样例一
接线图



代码

```
float sinVal;
int toneVal;
void setup() {
   // put your setup code here, to run once:
   pinMode(8,OUTPUT);
   pinMode(10,OUTPUT);
}

void loop() {
   // put your main code here, to run repeatedly:
   for(int x =0;x<180;x++){
      sinVal = (sin(x*(3.1412/180)));
      toneVal = 2000+(int(sinVal*1000));
      tone(8,toneVal);
      analogWrite(10,map(toneVal,2000,3000,10,255));
      delay(2);
   }
}
```

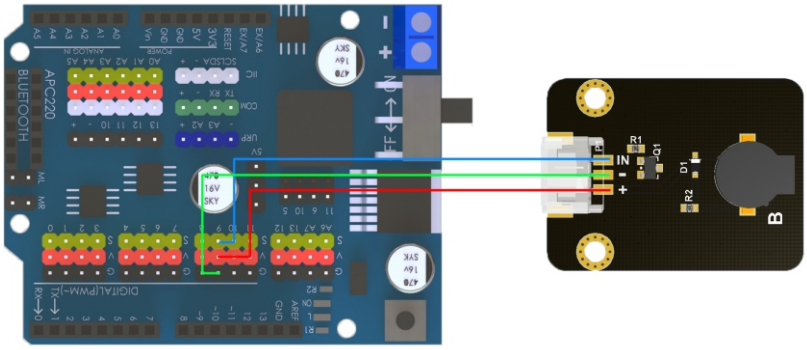
Mixly图形化编程



结果：引脚10接入LED灯，引脚8接入蜂鸣器。蜂鸣器会发出类似报警似的声响，而灯的亮度也随声响的起伏而变化。

样例二

接线图

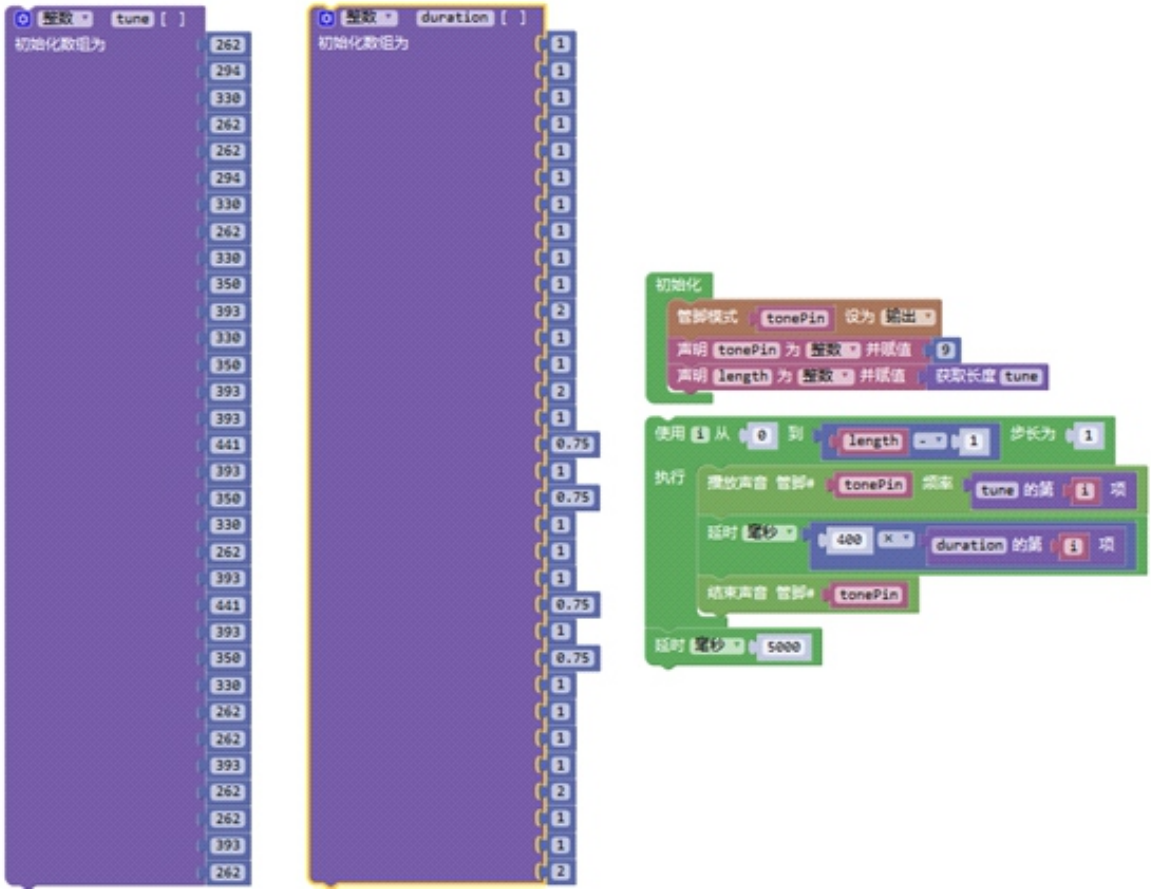


代码

```
#define NOTE_D0 -1
#define NOTE_D1 262
#define NOTE_D2 294
#define NOTE_D3 330
#define NOTE_D4 350
#define NOTE_D5 393
#define NOTE_D6 441
#define NOTE_D7 495
#define NOTE_DL1 131
#define NOTE_DL2 147
#define NOTE_DL3 165
#define NOTE_DL4 175
#define NOTE_DL5 196
#define NOTE_DL6 221
#define NOTE_DL7 248

#define NOTE_DH1 525
#define NOTE_DH2 589
#define NOTE_DH3 661
#define NOTE_DH4 700
#define NOTE_DH5 786
#define NOTE_DH6 882
#define NOTE_DH7 990
//以上部分是定义是把每个音符和频率值对应起来，其实不用打这么多，但是都打上了，后面可以随意编写

int tune[] =
{
  NOTE_D1,NOTE_D2,NOTE_D3,NOTE_D1,
  NOTE_D1,NOTE_D2,NOTE_D3,NOTE_D1,
  NOTE_D3,NOTE_D4,NOTE_D5,
  NOTE_D3,NOTE_D4,NOTE_D5,
  NOTE_D5,NOTE_D6,NOTE_D5,NOTE_D4,NOTE_D3,NOTE_D1,
  NOTE_D5,NOTE_D6,NOTE_D5,NOTE_D4,NOTE_D3,NOTE_D1,
```

```
 NOTE_D1,NOTE_D5,NOTE_D1,
 NOTE_D1,NOTE_D5,NOTE_D1};//这部分就是整首曲子的音符部分，用了一个序列定义为tune， 整数

float duration[]=

{
  1,1,1,1,
  1,1,1,1,
  1,1,1+1,
  1,1,1+1,
  0.5+0.5,0.25+0.5,0.5+0.5,0.25+0.5,1,1,
  0.5+0.5,0.25+0.5,0.5+0.5,0.25+0.5,1,1,
  1,1,1+1,
  1,1,1+1};
 //这部分是整首曲子的节拍部分，也定义个序列 duration，浮点（数组的个数和前面音符的个数是一样的，一一对应么）
int length;//这里定义一个变量，后面用来表示共有多少个音符

int tonePin=9;//蜂鸣器的 pin

void setup()
{
  pinMode(tonePin,OUTPUT);//设置蜂鸣器的 pin 为输出模式
  length = sizeof(tune)/sizeof(tune[0]);//这里用了一个 sizeof 函数， 可以查出 tone 序列里有多少个音符
}

void loop()
{
  for(int x=0;x<length;x++)//循环音符的次数
  {
    tone(tonePin,tune[x]);//此函数依次播放 tune 序列里的数组，即每个 音符
    delay(400*duration[x]);//每个音符持续的时间，即节拍 duration，是调整时间的越大，曲子速度越慢，越小曲子速度越快，自己掌握吧
    noTone(tonePin);//停止当前音符，进入下一音符
  }
  delay(5000);//等待 5 秒后，循环重新开始
}
```

Mixly图形化编程



结果：引脚9接入蜂鸣器，蜂鸣器响出两只两老虎的儿歌。

扩展：利用蜂鸣器播一首自己喜欢的歌吧!
步骤一：获取歌曲的简谱和蜂鸣器发出相对应的声音列表（音符频率表）。这里以两只老虎为例。

两 只 老 虎

1=C $\frac{4}{4}$

1	2	3	1	1	2	3	1	3	4	5	–	3	4	5	–
两	只	老	虎，	两	只	老	虎，	跑	得	快，		跑	得	快，	
5· 6	5· 4	3	1	5· 6	5· 4	3	1	1	5	1	–	1	5	1	–
一	只	没	有	眼睛，	一	只	没	有	耳朵，	真	奇	怪，		真	奇	怪。

音调 音符	1 .	2 .	3 .	4	5 .	6 .	7 .
A	221	248	278	294	330	371	416
B	248	278	294	330	371	416	467
C	131	147	165	175	196	221	248
D	147	165	175	196	221	248	278
E	165	175	196	221	248	278	312
F	175	196	221	234	262	294	330
G	196	221	234	262	294	330	371

音调 音符	1	2	3	4	5	6	7
A	441	495	556	589	661	742	833
B	495	556	624	661	742	833	935
C	262	294	330	350	393	441	495
D	294	330	350	393	441	495	556
E	330	350	393	441	495	556	624
F	350	393	441	495	556	624	661
G	393	441	495	556	624	661	742

音调 音符	. 1	 2	. 3	 4	 5	. 6	. 7
A	882	990	1112	1178	1322	1484	1665
B	990	1112	1178	1322	1484	1665	1869
C	525	589	661	700	786	882	990
D	589	661	700	786	882	990	1112
E	661	700	786	882	990	1112	1248
F	700	786	882	935	1049	1178	1322
G	786	882	990	1049	1178	1322	1484

步骤二：0基础也可以会看谱

简谱中蓝色框表曲子是用C调，所以我们只需要看音符频率表中的C行（用红色字体标注）。由于该简谱没有高低音调，所以这首曲子只用到了音符频率表中的中音的C调部分（用黄色底标注）。因此得到两只老虎的音符序列。
只有音符是不足够的，我们还需要给音符加上节奏，节奏可以分为一拍、半拍、1/4拍、1/8拍、两拍……
简谱中红色框中5后面有个横线，表示拍子+1。简谱中黄色框中的5右边有个小点，表示拍子+0.5；5下面有一条下划线，表示拍子是0.5；6下面有两条下划线，表示拍子是0.25。规律就是时间上单个音符没有下划线，就是一拍，有下划线是半拍（0.5），两个下划线是四分之一拍（0.25），音符后面有“—”等于前面音符的拍子+1，音符后面有点等于前面音符的拍子+0.5。因此得到两只老虎的节拍序列。

土壤湿度传感器模块

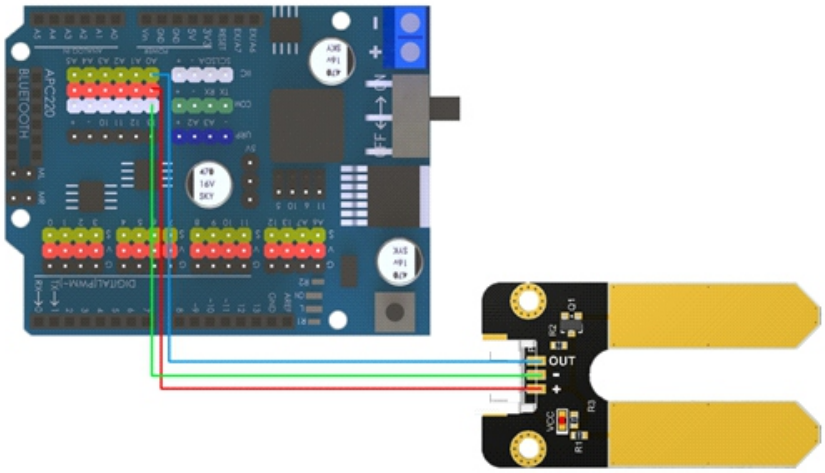
模块介绍：模块介绍：用于检测土壤的水分，当土壤缺水时，传感器输出值将减小，反之将增大。使用这个传感器制作一款自动浇花装置，让您的花园里的植物不用人去管理。传感器表面做了镀金处理，可以延长它的使用寿命。

参考值：0 ~300 ： 干燥土壤300~700 ： 湿润土壤700~950 ： 放到水中
输出类型：模拟输出。
接口模式：pH2.0-3P

引脚定义：

OUT	PIN
-	GND
+	VCC

Arduino uno
例子一
接线图



代码

```
void setup(){
    Serial.begin(9600); //打开波特率为 9600 串口
    pinMode(A0,INPUT);//土壤湿度模块连接引脚 A0，并为输入引脚
}
void loop(){
    Serial.print("Moisture Sensor Value:");//串口输出字符串
    Serial.println(analogRead(A0));        //串口输出读取土壤湿度模块输入模拟值
    delay(100);
}
```

结果：模块接入A0引脚，打开串口，观察到土壤的湿度值。

数字晃动传感器模块

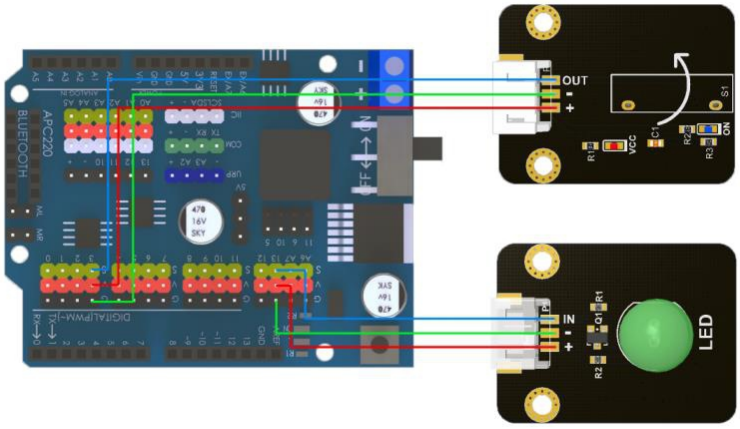
模块介绍：是一款仅对单方向手摇运动敏感的数字传感器。采用弹簧式震动开关，在外力晃动的时候，达到适当的晃动力时导电针将瞬间开启。且只有水平方向晃动才检测到晃动。可以利用digitalRead()来获取模块的高低电平，但由于开启的时间非常的短，建议使用中断来捕捉效果会更加好。该模块平时保

持高电平输出，晃动时输出低电平。
输出类型：数字接口。
接口模式：pH2.0-3P

引脚定义：

Out	PIN
-	GND
+	VCC

Arduino uno
例子一
接线图



代码

```
#define SensorLED        13
#define SensorINPUT      3   //模块连接引脚 3，是中断 1
unsigned char state = 0;
void setup()
{
   pinMode(SensorLED, OUTPUT);
   pinMode(SensorINPUT, INPUT);
   attachInterrupt(1, blink, CHANGE);//当模块电平发生变化时，触发中断
}
void loop()
{
        if(state!=0)
        {
           state = 0;
           digitalWrite(SensorLED,HIGH);
           delay(500);
        }
        else
           digitalWrite(SensorLED,LOW);
}
void blink()//Interrupts function
{
   state++;
}
```

结果：当模块晃动时。控制板上的led亮0.5秒后灭掉，在模块不晃动时，控制板上的led不亮。

数码管理模块

模块介绍：MAX7219是一种集成化的串行输入/输出共阴极显示驱动器,它连接微处理器与8位数字的7段数字LED显示，也可以连接条线图显示器或者64个独立的LED。其上包括一个片上的B型BCD编码器、多路扫描回路，段字驱动器，而且还有一个8*8的静态RAM用来存储每一个数据。只有一个外部寄存器用来设置各个LED的段电流。

输入类型：
接口模式：pH2.0-5P

引脚定义：

CLK	时钟序列输入端
CS	载入数据
DIN	串行数据输入
-	GND
+	VCC

DIN引脚是串行数据输入，数据在CLK的上升沿加载到内部16位位移寄存器中；CS引脚是片选输入，串行数据在CS为低电平时加载到移位寄存器中，最后的16位串行数据被锁存在CS的上升沿；CLK引脚是串行时钟输入，在CLK的上升沿时，最终移位寄存器，在CLK的下降沿时，数据从DOUT输出。

寄存器地址映射

寄存器	地址					HEX码
	D15-D12	D11	D10	D9	D8	
无操作	X	0	0	0	0	0xX0
数字0	X	0	0	0	1	0xX1
数字1	X	0	0	1	0	0xX2
数字2	X	0	0	1	1	0xX3
数字3	X	0	1	0	0	0xX4
数字4	X	0	1	0	1	0xX5
数字5	X	0	1	1	0	0xX6
数字6	X	0	1	1	1	0xX7
数字7	X	1	0	0	0	0xX8
解码模式	X	1	0	0	1	0xX9
强度	X	1	0	1	0	0xXA
扫描限制	X	1	0	1	1	0xXB
关掉	X	1	1	0	0	0xXC
显示测试	X	1	1	1	1	0xXF

初始化max7219

1、解码模式有无解码数字（即D7-D0无解码数字）、只解码D0、对D3-D0解码、解码D7-D0。

表4.解码模式寄存器示例（地址（十六进制）= 0xX9）

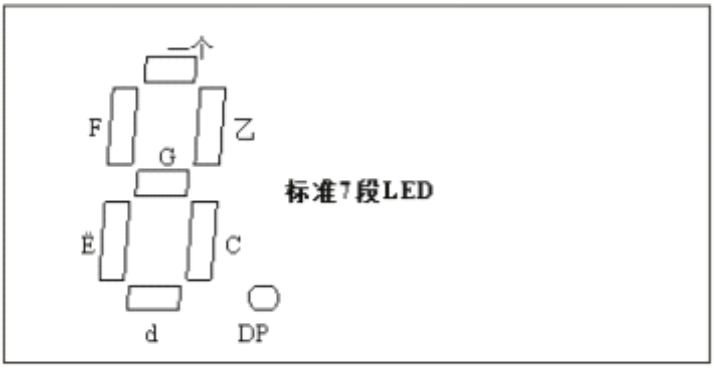
解码模式	注册数据								HEX码
	D7	D6	D5	D4	D3	D2	D1	D0	
没有解码数字7-0	0	0	0	0	0	0	0	0	为0x00
代码B解码数字0 没有解码数字7-1	0	0	0	0	0	0	0	1	0x01
代码B对数字3-0进行解码 数字7-4没有解码	0	0	0	0	1	1	1	1	为0x0F
代码B对数字7-0进行解码	1	1	1	1	1	1	1	1	为0xFF

表5.代码B字体

7段字符	注册数据						ON SEGMENTS = 1							
	D7 *	D6-D4	D3	D2	D1	D0	DP *	一个	乙	C	d	Ë	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
五		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
-		X	1	0	1	0		0	0	0	0	0	0	1
Ë		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
大号		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
空白		X	1	1	1	1		0	0	0	0	0	0	0

*小数点由位D7 = 1设置

表6.无解码模式数据位和对应的细分行



标准7段LED

	注册数据							
	D7	D6	D5	D4	D3	D2	D1	D0
相应段线	DP	一个	乙	C	d	Ë	F	G

2、数码管的亮度

3、扫描限制：数码管显示，从0x00开始，从右往左开始数，一个模块最多到0x07。例如，显示数码管从左边数第四个到最后一个，则扫描限制是0x04。

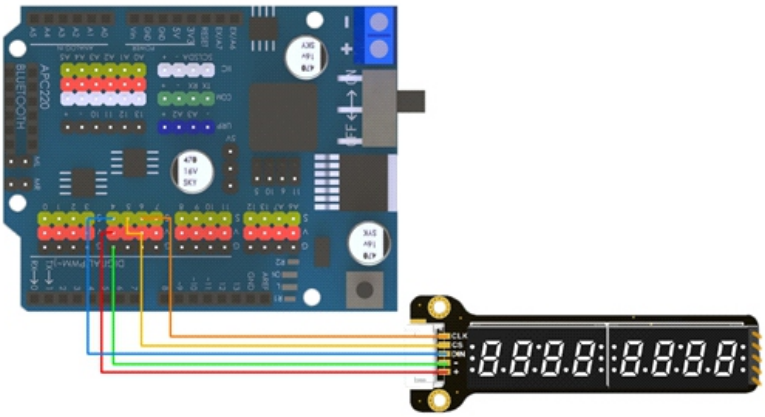
4、掉电模式有关掉模式（0x00）和正常模式（0x01）

表3.关闭寄存器格式（地址（十六进制）= 0xXC）

模式	地址代码（HEX）	注册数据							
		D7	D6	D5	D4	D3	D2	D1	D0
关掉模式	0xXC	X	X	X	X	X	X	X	0
正常手术	0xXC	X	X	X	X	X	X	X	1

5、显示测试

Arduino
样例一
接线图



代码

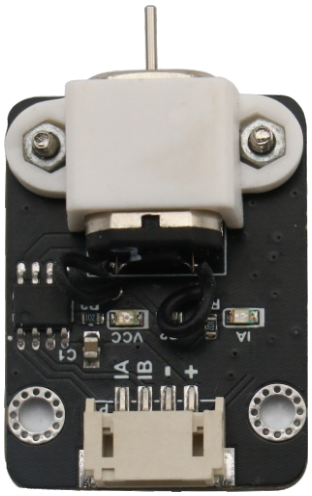
```
int CLK = 6;
int CS = 5;
int DIN = 4; //定义了三个脚

void setup() {
   pinMode(CLK,OUTPUT);
   pinMode(CS,OUTPUT);
   pinMode(DIN,OUTPUT); //三个脚都是输出状态
}

void loop() {
     Delay_xms(50);
     Init_MAX7219();
     Delay_xms(2000);
     Write_Max7219(0x0f, 0x00);          //显示测试：1；测试结束，正常显示：0
     Write_Max7219(1,1);
     Write_Max7219(2,2);
     Write_Max7219(3,3);
     Write_Max7219(4,4);
     Write_Max7219(5,5);
     Write_Max7219(6,6);
```

```
    Write_Max7219(7,7);
    Write_Max7219(8,8);
    while(1);
}
void Delay_xms(unsigned int x)
{
   unsigned int i,j;
   for(i=0;i<x;i++)
   for(j=0;j<112;j++);
}
//-----------------------------------------
//功能：向 MAX7219(U3)写入字节
//入口参数：DATA
//出口参数：无
//说明：
void Write_Max7219_byte(unsigned char DATA)
{
      unsigned char i;
      digitalWrite(CS,LOW);
      for(i=8;i>=1;i--)
      {
         digitalWrite(CLK,LOW);
         if(DATA&0X80)
                digitalWrite(DIN,HIGH);
         else
                digitalWrite(DIN,LOW);
         DATA<<=1;
         digitalWrite(CLK,HIGH);
       }
}
//-----------------------------------------
//功能：向 MAX7219 写入数据
//入口参数：address   dat
//出口参数：无
//说明：
void Write_Max7219(unsigned char address,unsigned char dat)
{
    digitalWrite(CS,LOW);
    Write_Max7219_byte(address);              //写入地址，即数码管编号
    Write_Max7219_byte(dat);                  //写入数据，即数码管显示数字

    digitalWrite(CS,HIGH);
}
```

```
void Init_MAX7219(void)
{
    Write_Max7219(0x09, 0xff);          //译码方式：BCD 码//1001
    Write_Max7219(0x0a, 0x02);           //亮度//1010
    Write_Max7219(0x0b, 0x07);           //扫描界限；4 个数码管显示//1011
    Write_Max7219(0x0c, 0x01);           //掉电模式：0，普通模式：1//1100
    Write_Max7219(0x0f, 0x01);           //显示测试：1；测试结束，正常显示：0//1111
}
```

结果：数码管从右往左显示1到8。

风扇模块



模块介绍：直流电动机是把直流电能转换成机械能的一类电机。最常见的类型依赖于磁场产生的力。几乎所有类型的直流电动机都有一些内部机制，无论是机电的还是电子的，以周期性地改变部分电动机的电流流向。大多数类型产生旋转运动；直线电机直接产生力和直线运动

工作电压： 5V。
输入类型：数字信号、PWM
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：

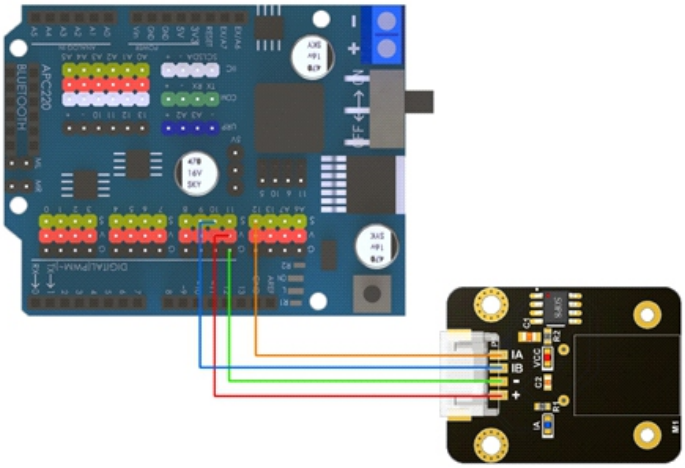
引脚定义：

IA	正转数字输入
IB	反转数字输入
-	GND
+	VCC

控制方式

IA	IB	功能
0	0	停下
0	1	反向转
1	0	正向转

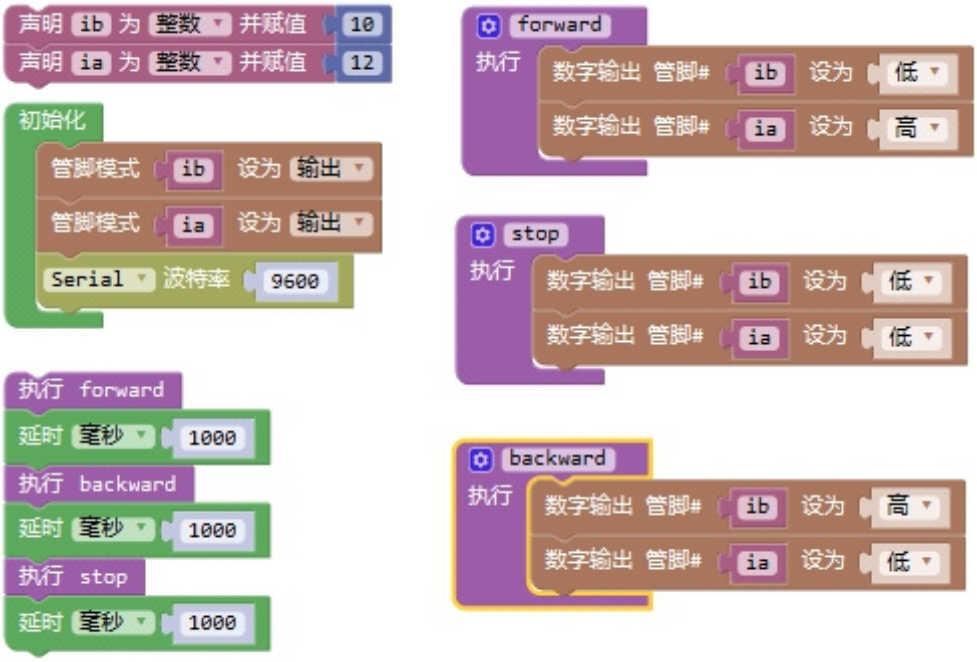
Arduino uno
样例一
接线图



代码

```
const int IB = 6;
const int IA = 5;
int buttonpin = A1;
boolean state = false;

void setup() {

   Serial.begin(9600);
   pinMode(IA, OUTPUT);
   pinMode(IB, OUTPUT);
   digitalWrite(IA, LOW);
   digitalWrite(IB, LOW);
   pinMode(buttonpin, INPUT);
   pinMode(4, INPUT);
}

void loop()
{

    analogWrite(IA, 0);
      analogWrite(IB, 0);
      delay(1000);

    analogWrite(IA, 0);
      analogWrite(IB, 255);
      delay(1000);
}
```

Mixly图形化编程



结果：IB接口接入引脚6，IA接口接入引脚5。电机旋转1秒后，停止旋转1秒为周期循环运动。

麦克风模块

（声音传感器模块）

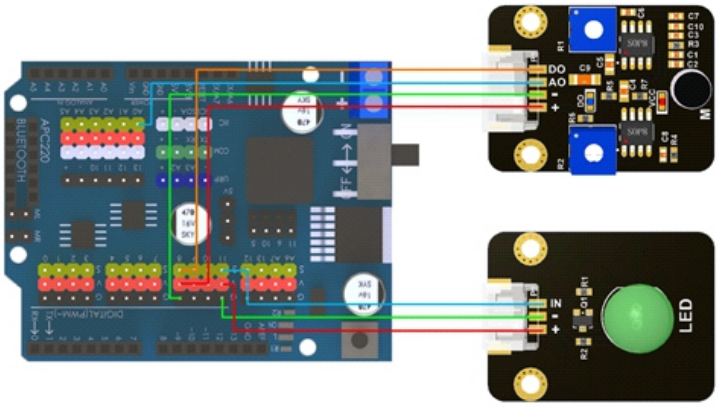
模块介绍：输出模块，数字模拟接口。其作用相当于话筒。用来接收声波，显示振动图像，但不能对噪声的强度进行测量。声波使话筒内的驻极体薄膜振动，导致电容的变化，从而产生与之对应变化的微小电压，经过数字/模拟转换被数据采集器接收，并传给控制板。
AO模拟量实时输出麦克风的接收到声量信号
DO当声音强度到达某个阈值，则输出低电平信号（可通过模块上的R2来调节），由于输出低电平的时间非常的短，为了捕捉到有用的信息使用信息过滤或中断来呈现，此处用信息过滤。

工作电压：3.3V到5V
输出类型：数字输出、模拟输出。
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：50g

引脚定义：

DO	数字输出
AO	模拟输出
-	GND
+	VCC

Arduino
例子一
接线图



代码

```
void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600);
   pinMode(8,INPUT);
   pinMode(A0,INPUT);
   pinMode(11, OUTPUT);
}

void loop() {
  int astate = analogRead(A0);
   int dstate = digitalRead(8);
   if(dstate==0){
      Serial.print("数字接口：");
      Serial.println(dstate);
      Serial.print("模拟接口：");
      Serial.println(astate);
      analogWrite(11,(map(astate, 1, 1023, 1, 255)));
   }
 else {
      digitalWrite(11,LOW);
   }
}
```

结果：模块的DO接入引脚8，对着话筒拍手，模块上的DO处的led灯亮；反之，led灯灭。旋转R2上的十字口，会发现逆时针旋转，模块对声音的灵敏度会越高；加上一个有源蜂鸣器，旋转R1上的十字口，会发现模块对接收到声音信号的放大不同的倍数后显示出来，接收的声音越大，放大的效果更明显。

继电器模块

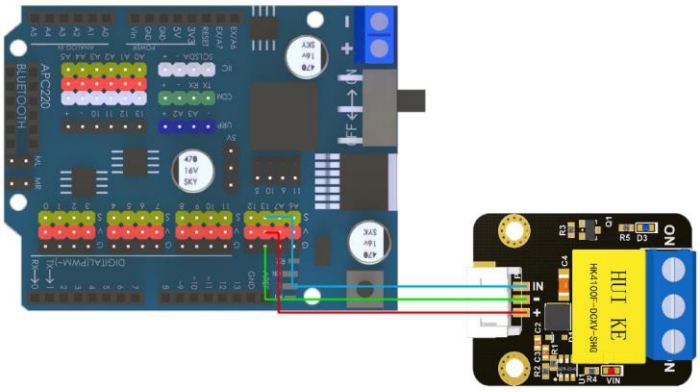
模块介绍：继电器（英文名称：relay）是一种电控制器件，是当输入量的变化达到规定要求时，在电气输出电路中使被控量发生预定的阶跃变化的一种电器。它具有控制系统（又称输入回路）和被控制系统（又称输出回路）之间的互动关系。通常应用于自动化的控制电路中，它实际上是用小电流去控制大电流运作的一种“自动开关”。故在电路中起着自动调节、安全保护、转换电路等作用。
继电器旁有三个接口，NO是nomal open，常开触点，继电器线圈未通电时断开，NC是nomal close，常闭触点，继电器线圈未通电时导通。线圈得电时，常开接点会闭合，而常闭接点会断开。对于 电器上配置的与主开关联动的辅助开关：当主开关合上时，辅助开关也合上。

输入类型：数字输入
接口模式：pH2.0-3P

引脚定义：

In	数字输入
-	GND
+	VCC

Arduino
例子一
接线图

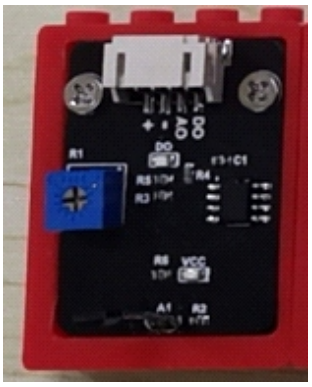


代码

```
int incomedata =0;
int relayPin =13;
void setup() {
   pinMode(relayPin,OUTPUT);
   Serial.begin(9600);
}

void loop() {
   if(Serial.available()>0){
      incomedata = Serial.read();
      if(incomedata == 'H'){
         digitalWrite(relayPin,HIGH);
         Serial.println("open");
      }
      else if(incomedata == 'L'){
         digitalWrite(relayPin,LOW);
         Serial.println("close");
      }
   }
}
```

结果：继电器接入引脚13，打开串口，当向串口输入“H”时，连接在继电器上的灯亮；当向串口输入”L”时，连接在继电器上的灯灭。

霍尔模块



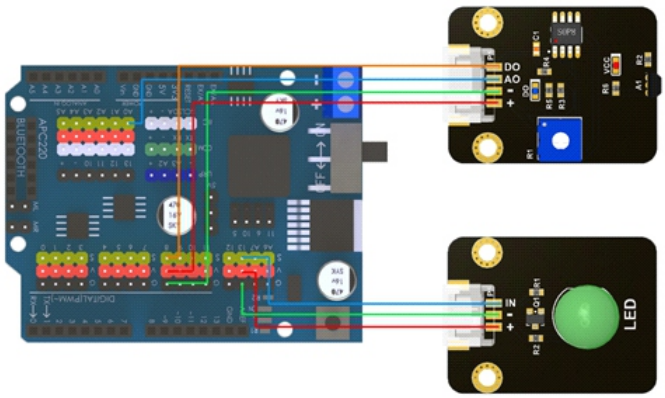
模块介绍：根据霍尔效应制作的一种磁场传感器。霍尔效应是磁电效应的一种，这一现象是霍尔（A.H.Hall，1855—1938）于1879年在研究金属的导电机构时发现的。后来发现半导体、导电流体等也有这种效应，而半导体的霍尔效应比金属强得多，利用这现象制成的各种霍尔元件，广泛地应用于工业自动化技术、检测技术及信息处理等方面。霍尔元件是一种基于霍尔效应的磁传感器。用它们可以检测磁场及其变化，可在各种与磁场有关的场合中使用。霍尔器件具有许多优点，结构上比较牢固，体积小巧，重量轻，安装起来很方便，霍尔元件的功耗普遍很小，所以它的使用寿命很长。除此之外霍尔元件还有功耗小，频率高（可达1MHZ），耐震动，不怕灰尘、油污、水汽及盐雾等的污染或腐蚀的优良特点，被广泛应用于工业、汽车以及消费电子产品中。该模块使用的霍尔元件是49E（线性霍尔），输出模拟电压信号，随着磁场的极性和大小变化其输出电压相应变化。常用于角度控制、调速应用等。

输出类型：数字输出、模拟输出。
接口模式：pH2.0-4P

引脚定义：

DO	数字输出
AO	模拟输出
-	GND
+	VCC

Arduino
例子一
接线图

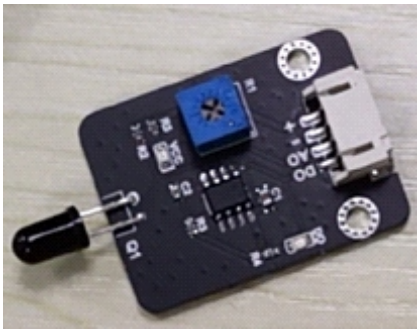


代码

```
int ledpin =13;
int hallpin =8;
int hallApin =A0;
void setup() {
   pinMode(hallpin,INPUT);
   pinMode(hallApin,INPUT);
   pinMode(ledpin,OUTPUT);
   Serial.begin(9600);
}

void loop() {
   Serial.println(analogRead(huoerApin));
   if(digitalRead(huoerpin)){
      digitalWrite(ledpin,LOW);
   }
   else{
      digitalWrite(ledpin,HIGH);
      delay(500);
   }
}
```

结果：数字接口接入引脚8，模拟接口接入引脚A0，当摩尔模块感测到有磁场，便触发信号，模块指示块和板子上的led亮，并延时0.5秒。打开串口观察到模块从模拟接口发出的信息值变化。
提示：尝试用螺丝刀扭一下模块的R1上的十字口，再测试上例，增大减小摩尔模块和磁铁之间的距离，观察看看led的在什么位置会亮。

火焰传感器模块



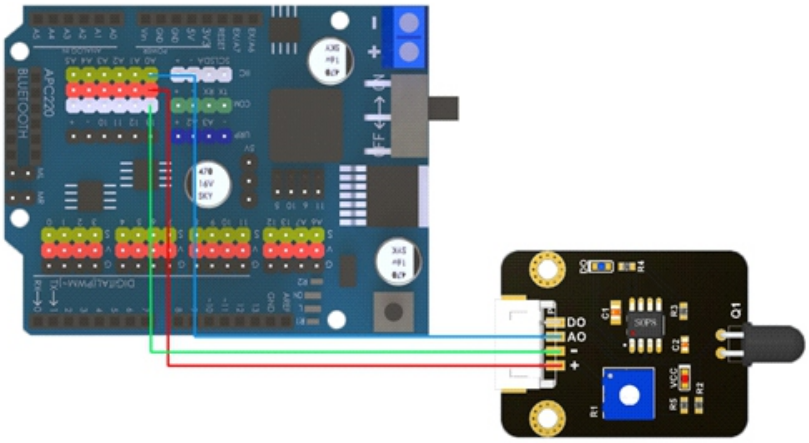
模块介绍：模块介绍：火焰传感器可以用来探测火源或其它波长在760纳米～1100纳米范围内的光源。在灭火机器人比赛中，火焰探头起着非常重要的作用，它可以用作机器人的眼睛来寻找火源或足球。利用它可以制作灭火机器人、足球机器人等。火焰传感器的探测角度达60度，对火焰光谱特别灵敏，2个M3安装孔，可以稳定模块不会旋转。 这款火焰传感器能在-25到85摄氏度下工作，性能稳定可靠。尽管这款传感器是用来感知火焰，但是它并不防火。因此使用时请与火焰保持距离，以免烧坏传感器。

输出类型：模拟输出、数字输出。
接口模式：pH2.0-4P

引脚定义：

DO	数字输出
AO	模拟输出
-	GND
+	VCC

Arduino
例子一
接线图

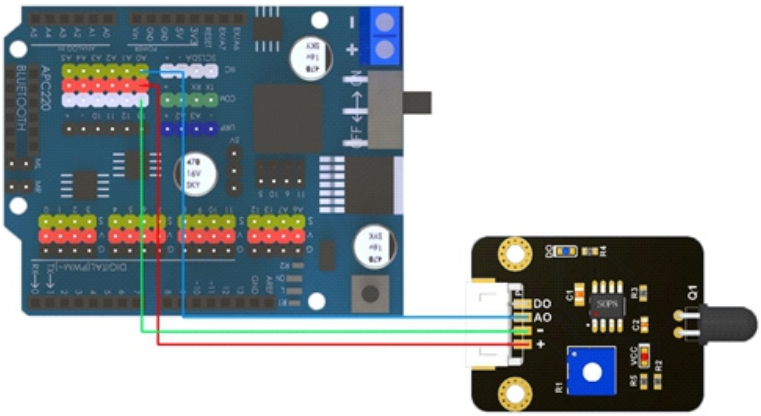


代码

```
void setup() {
  Serial.begin(9600);
  pinMode(A0,INPUT);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

结果：当火焰传感器检测得到火源之火，在串口中可观察点数值的变化。

例子二
接线图

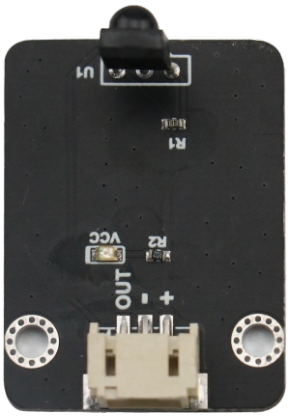


代码

```
void setup() {
  Serial.begin(9600);
  pinMode(4,INPUT);
}

void loop() {
  int sensorValue = digitalRead(4);
  Serial.println(sensorValue);
}
```

结果：当火焰传感器检测得到火源之火，在串口中可观察电平的变化。旋转数码盘，可发现模块的灵敏度发生变化。

红外接收模块



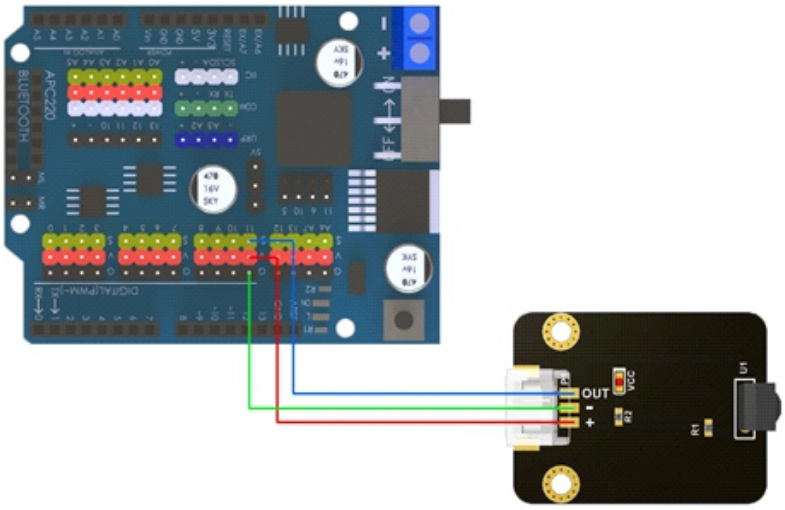
模块介绍：外接收管就是接收红外光的电子器件。比如我们电视机，空调这些家电，其实它们都需要用到红外接收管。我们都知道遥控器发射出来的都是红外光，电视机上势必要有红外接收管，才能接收到遥控器发过来的红外信号。

工作电压：3.3V到5V
输出类型：数字信号。
接口模式：pH2.0-3P
模块尺寸：35*26.3mm
模块重量：40g

引脚定义：

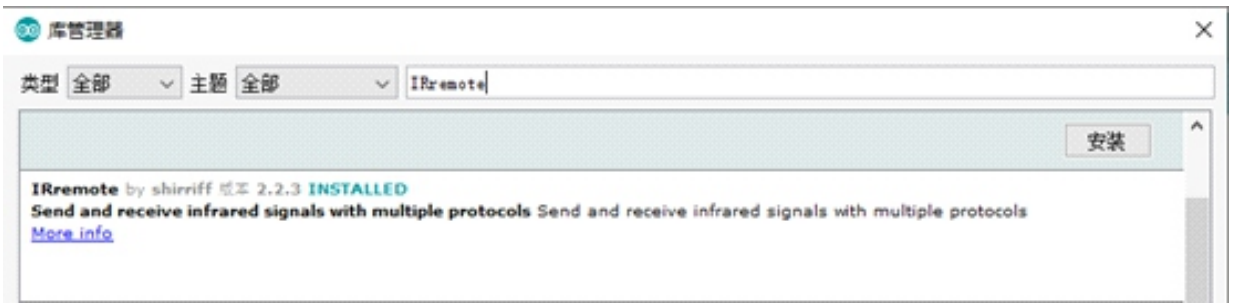
OUT	输出
-	GND
+	VCC

Arduino uno
样例一
接线图



代码

在开始编程之前，我们需要添加红外接收的库“IRremote.h”。点击“项目”->“加载库”->”管理库”->搜索“IRremote”，安装下图的IRremote库。



安装库后可以编程了。我们在“文件”->“示例”->在“第三方库示例”下找到IRremote，该库提供了该库使用红外接收的一些示例。

代码

```
#include <IRremote.h>
int RecvPin =11;//红外接收模块连接引脚 11
IRrecv irrecv(RecvPin);//初始化 IRrecv 实例
decode_results results;//接收到的数据 results
void setup() {
  Serial.begin(9600);//打开波特率为 9600 的串口
  irrecv.enableIRIn();//实例允许接收数据
}

void loop() {
  if(irrecv.decode(&results)){//对 results 解码，并判断是否接收到数据
    Serial.println(results.value,HEX);//串口输出接收数据值，并以 16 进制输出
    irrecv.resume();//实例重新接收数据
  }
}
```

Mixly图形化编程

结果：引脚11接入红外接收模块，打开串口窗口。当红外遥控向红外接收器发送信息，红外接收模块接收到信息，显示在串口窗口上。

后续可利用接收到的16进制代码，控制灯、舵机、电机等等。

红绿灯模块

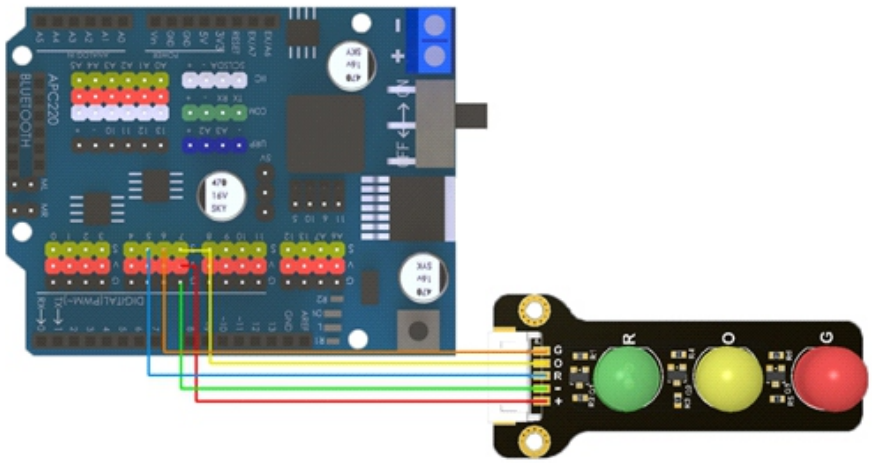
模块介绍：是由红绿黄三个颜色不同的led组成，每个led单独控制，通过高低电平实现led的亮灭状态。

输入类型：数字输入
接口模式：pH2.0-5P

引脚定义：

G	绿色 led 输入
O	黄色 led 输入
R	红色 led 输入
-	gnd
+	Vcc

Arduino
样例一
接线图

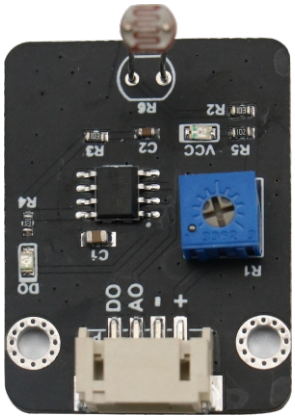


代码

```
void setup() {
  // put your setup code here, to run once:
  pinMode(5,OUTPUT);//r
  pinMode(6,OUTPUT);//g
  pinMode(7,OUTPUT);//o
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(5,HIGH);
  delay(5000);
  digitalWrite(5,LOW);
  digitalWrite(7,HIGH);
  delay(5000);
  digitalWrite(7,LOW);
  delay(1000);
  digitalWrite(7,HIGH);
  delay(1000);
  digitalWrite(7,LOW);
  delay(1000);
  digitalWrite(7,HIGH);
  delay(1000);
  digitalWrite(7,LOW);
  delay(1000);
  digitalWrite(7,HIGH);
  delay(1000);
  digitalWrite(7,LOW);
  digitalWrite(6,HIGH);
  delay(2000);
}
```

结果：模块模拟红绿灯闪烁，红灯亮5秒，转黄灯闪烁4次后，转绿灯亮5秒。

光敏模块



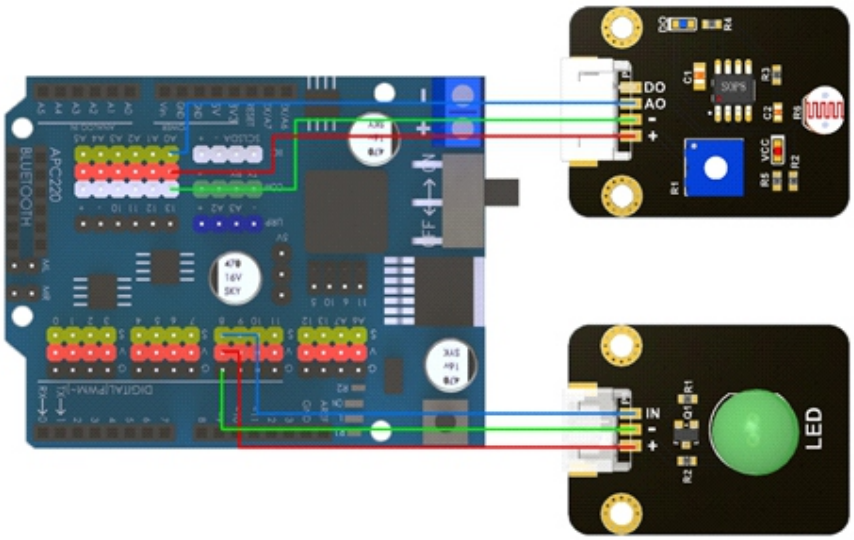
模块介绍：光敏电阻是一种光控可变电阻器。光敏电阻具有光电导性，可用于光敏探测器电路。光敏电阻是由高电阻半导体制成的。在黑暗中光敏电阻的电阻可以高达几兆欧（MΩ），而在充足光线下，光敏电阻的电阻可以低至几百欧姆。如果光敏电阻上的入射光超过一定的频率，光子被半导体吸收给束缚电子足够的能量跳跃进入传导带。由此产生的自由电子导电，从而降低电阻。

工作电压：3.3V到5V
输出类型：数字、模拟接口。此模块，入射光线越强，输出模拟值越低。
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：42g

引脚定义：

DO	数字输出
AO	模拟输出
-	GND
+	VCC

Arduino uno
样例一
接线图



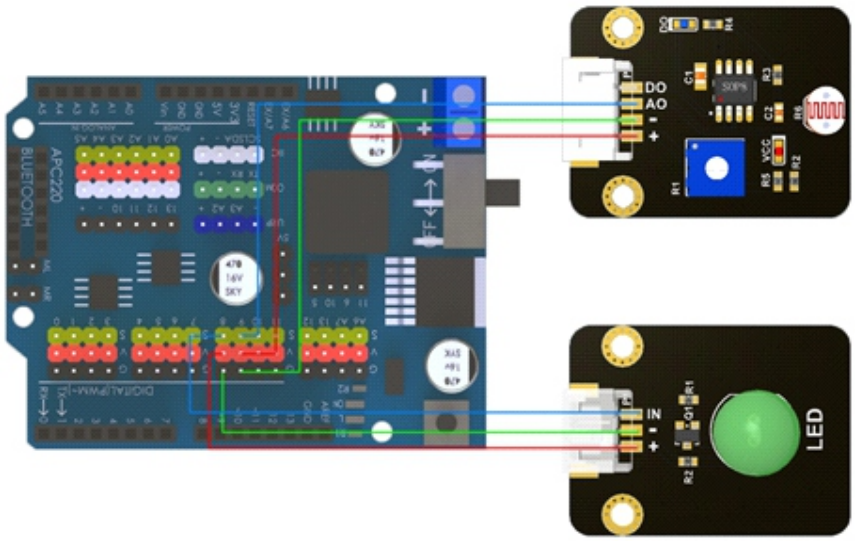
代码

```
void setup() {
  pinMode(8, OUTPUT);//LED 的引脚 8，并为输出引脚
  pinMode(A0,INPUT);//光敏电阻的引脚 A0，并为输入引脚
  Serial.begin(9600);//打开波特率为 9600 串口
}
void loop(){
  int state = analogRead(A0);//读取光敏电阻模拟输入

    Serial.println(state);//串口输出光敏电阻输入模拟值
    if(state<800){//判断光敏电阻输入模拟值是否小于 800
        digitalWrite(8,LOW);//灯亮起状态
    }
    else{
        digitalWrite(8,HIGH);//灯关闭状态
    }
}
```

Mixly图形化编程

结果：引脚A0接入光敏电阻的模拟接口，引脚8接入led灯，用手电筒照射光敏电阻模块，在模拟输出值高于800时，led亮；反之，led灯不亮。

样例二
接线图



代码

```
void setup() {
  pinMode(8, OUTPUT); //LED 的引脚 8，并为输出引脚
  pinMode(9,INPUT); //光敏电阻的引脚 9，并为输入引脚
  Serial.begin(9600); //打开波特率为 9600 串口
}
void loop(){
  int state = digitalRead(9); //读取光敏电阻数字输入

    Serial.println(state); //串口输出光敏电阻输入数字值
    if(state){ //判断光敏电阻输入数字值是否是否为高
       digitalWrite(8, HIGH);  //  灯亮起状态
    }
    else{
       digitalWrite(8, LOW);  //  灯关闭状态
    }
}
```

Mixly图形化编程

结果：引脚9接入光敏电阻的数字接口，引脚8接入Led灯，当引脚9输出的数字信号为0时，led灯亮；反之，led灯不亮。
提示：尝试用螺丝刀扭一下模块的R1上的十字口，在测试下上例，观察看看串口输出的内容有什么变化。

干簧管模块

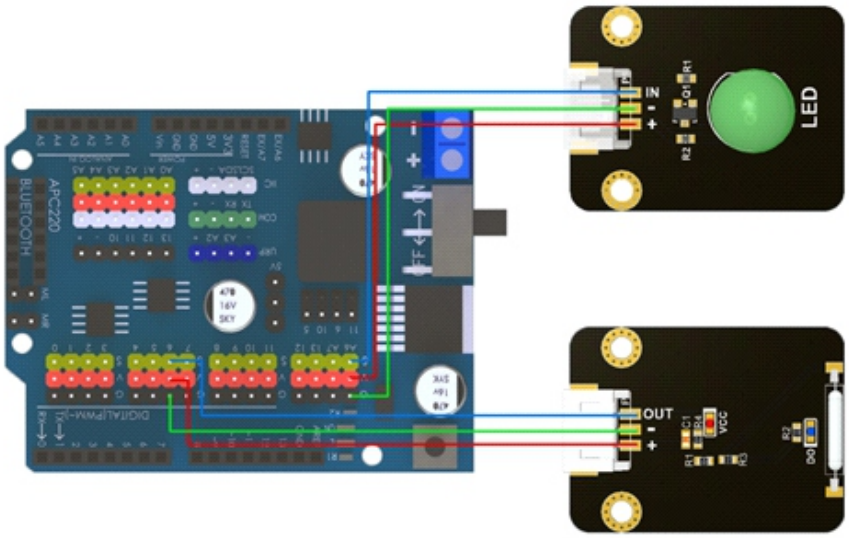
模块介绍：是当磁铁靠近干簧管时，干簧管里的两个簧片被磁化而接通，所以干簧管边起到的开关的作用。干簧管在程控交换机、复印机、洗衣机、电冰箱、照相机、消毒碗柜、门磁、窗磁、电磁继电器、电子衡器、液位计、煤气表、水表中等等方面都得到了很好的应用。

输出类型：数字输出。
接口模式：pH2.0-3P

引脚定义：

Out	输出
+	电源
-	地

Arduino
样例一
接线图

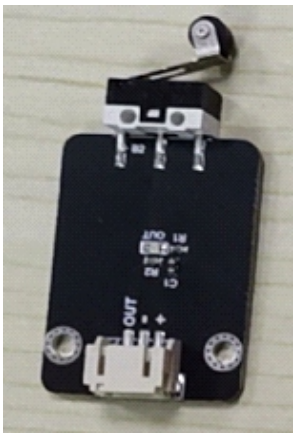


代码

```
void setup() {
   pinMode(6,INPUT);
   pinMode(13,OUTPUT);
}

void loop() {
   if(digitalRead(6)){
      digitalWrite(13,LOW);
   }
   else{
      digitalWrite(13,HIGH);
   }
}
```

结果：当模块靠近磁铁的时候，输出低电平，接到13引脚的led亮起；远离磁铁的时候，输出高电平，led秒掉。

碰撞模块



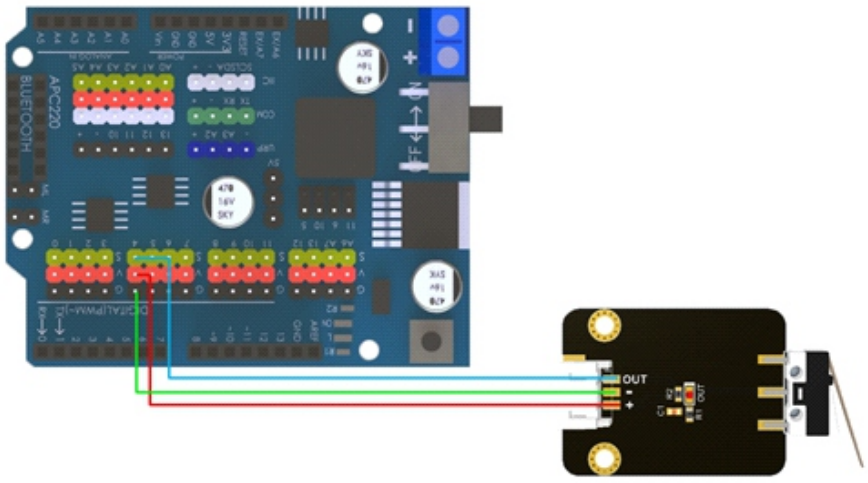
模块介绍：碰撞模块也称为电子开关。当碰撞模块被检测到碰撞，会发出低电平信号给控制板，反之发出高电平信号。可以实现发光灯控制，发声器控制，LCD显示按键选择功能等；也可以将传感器扩展板安装到机器人上，实现碰撞检测功能，使用方便简单。

输出类型：数字输出
接口模式：pH2.0-3P

引脚定义：

Out	PIN
-	GND
+	VCC

Arduino
例子一
接线图

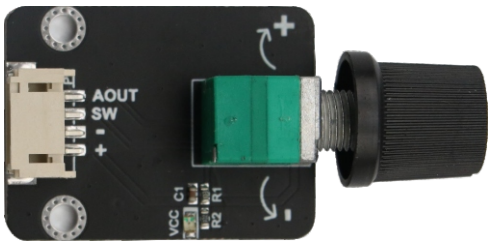


代码

```
void setup() {
   pinMode(4,INPUT);//模块连接 4 引脚，并为输入
   Serial.begin(9600);//打开波特率为 9600 串口
}

void loop() {
   Serial.println(digitalRead(4));//串口输出读取模块数字值
}
```

结果：碰撞模块，观察串口输出内容。

电位器模块



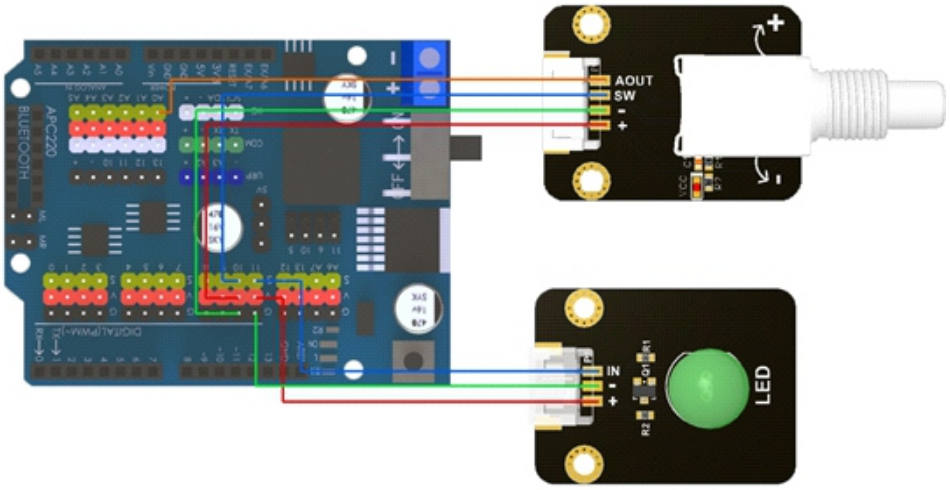
模块介绍：电位器是一种三端电阻元件，其电阻可根据一定的规则变化。它通常由一个电阻元件和一个可移动电刷组成。当电刷沿着元件移动时，在端子处会产生相对于其移动距离的电阻或电压。

工作电压：3.3V到5V
输出类型：获得电位器开关的数字输出和获取电位器值（0-1023）的模拟输出。
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：93g

引脚定义：

AUTO	模拟输出
SW	数字输出
-	GND
+	VCC

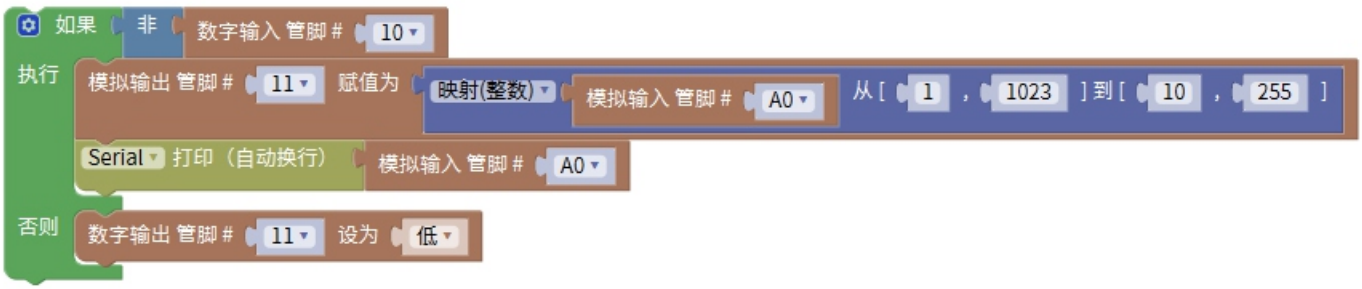
Arduino uno
样例一
接线图



代码

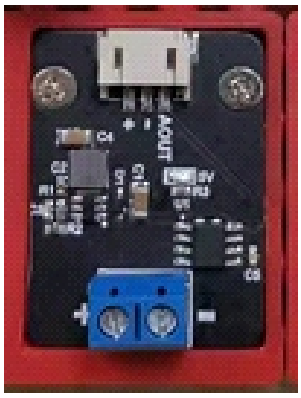
```
void setup() {
  pinMode(10,INPUT);//电位器 sw 连接 10 引脚，并为输入
  pinMode(A0,INPUT);//电位器 AOUT 连接 A0 引脚，并为输入
  pinMode(11,OUTPUT);//LED 连接 11 引脚，并为输出
  Serial.begin(9600);//打开波特率为 9600 串口
}

void loop() {
  int state = digitalRead(10);//读取电位器 sw 电平，高电平为按键未按下，低电平为按键按下
  int value = analogRead(A0);//读取电位器 AOUT 的模拟值，0-1023
  if(state==0){//判断是否未按下按键
    analogWrite(11,map(value,0,1023,10,255));//LED 根据电位器输入的模拟值，变化亮度
  }
  else{
    digitalWrite(11,LOW);//LED 关闭
  }
  Serial.println(value);//串口输出电位器模拟值
}
```

Mixly图形化编程



结果：引脚10接入电位器的数字接口，引脚A0接入电位器的模拟接口，引脚11接入led灯。Led灯的亮度随电位器的扭动而变化。

电流传感器模块



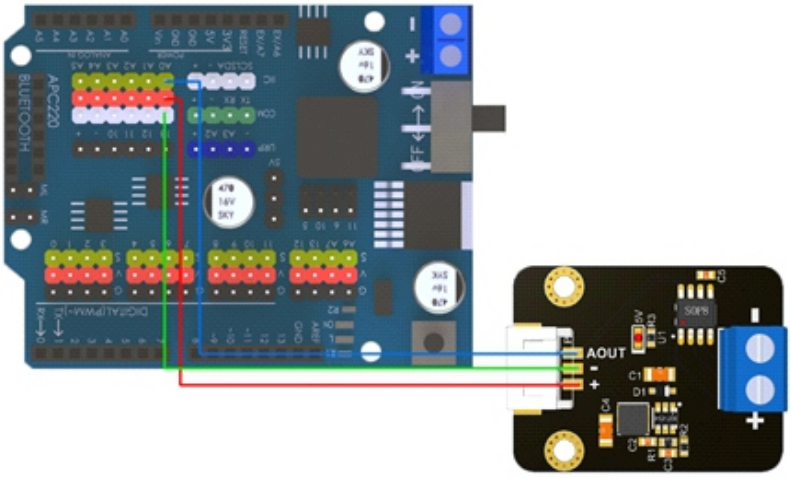
模块介绍：采用ACS712霍尔电流感应芯片，具有量程大、简单易用、体积小巧、精度较高、无需焊接等特点，可用于直流电流和交流电流的测量。在设计上做了高压隔离，确保使用的安全性。模块输出的电压线性对应测量电流。该模块不建议在20A及以上的大电流情况下长时间工作。在测量高压电时，请注意安全。

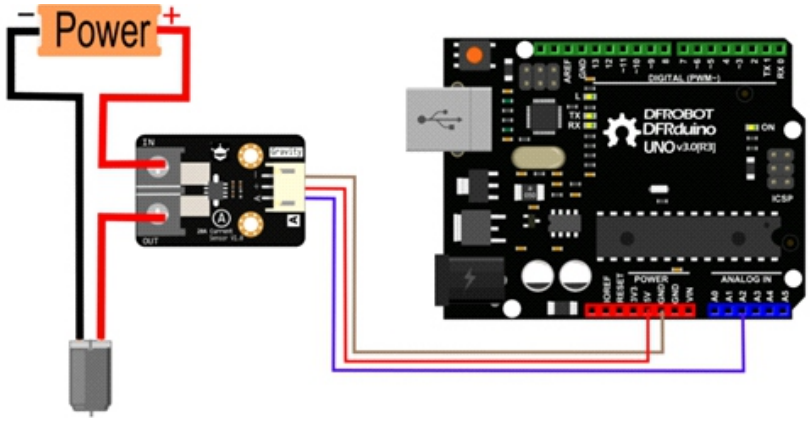
输出类型：模拟信号。
接口模式：pH2.0-3P

引脚定义：

AOUT	模拟输出
+	电源
-	地
测量电流输入+	测量电流输入端
测量电流输出-	测量电流输出端

Arduino
样例一
接线图





代码

```
const int currentSensorPin = A0; //define sensor pin
const int mVperAmp = 100; // use 185 for 5A Module, and 66 for 30A Module
float Vref    = 0; //read your Vcc voltage,typical voltage should be 5000mV(5.0V)

void setup()
{
      Serial.begin(115200);
      Vref = readVref(); //read the reference votage(default:VCC)
}

void loop()
{
      /*If you are reading DC current, use this function to read DC current. Then uncomment
the AC function.*/
      float CurrentValue =    readDCCurrent(currentSensorPin);
      /*If you are reading AC current, use this function to read AC current,it returns the RMS.
Then uncomment the DC function.*/
      //float CurrentValue =    readACCurrent(currentSensorPin);
      Serial.println(CurrentValue);
      delay(500);
}

/*read DC Current Value*/
float readDCCurrent(int Pin)
{
      int analogValueArray[31];
      for(int index=0;index<31;index++)
      {
         analogValueArray[index]=analogRead(Pin);
      }
      int i,j,tempValue;
```

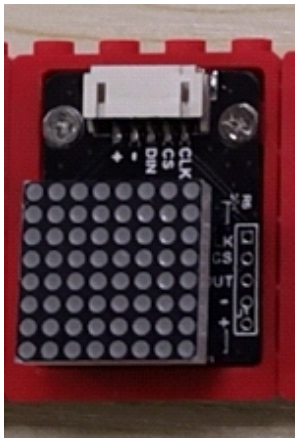
```
    for (j = 0; j < 31 - 1; j ++)
    {
        for (i = 0; i < 31 - 1 - j; i ++)
        {
            if (analogValueArray[i] > analogValueArray[i + 1])
            {
                tempValue = analogValueArray[i];
                analogValueArray[i] = analogValueArray[i + 1];
                analogValueArray[i + 1] = tempValue;
            }
        }
    }
    float medianValue = analogValueArray[(31 - 1) / 2];
    float DCCurrentValue = (medianValue / 1024.0 * Vref - Vref / 2.0) / mVperAmp;
//Sensitivity:100mV/A, 0A @ Vcc/2
    return DCCurrentValue;
}

/*read AC Current Value and return the RMS*/
float readACCurrent(int Pin)
{
    int analogValue;                //analog value read from the sensor output pin
    int maxValue = 0;               // store max value
    int minValue = 1024;            // store min value
    unsigned long start_time = millis();
    while((millis()-start_time) < 200) //sample for 0.2s
    {
        analogValue = analogRead(Pin);
        if (analogValue > maxValue)
        {
            maxValue = analogValue;
        }
        if (analogValue < minValue)
        {
            minValue = analogValue;
        }
    }
    float Vpp = (maxValue - minValue) * Vref / 1024.0;
    float Vrms = Vpp / 2.0 * 0.707 / mVperAmp; //Vpp -> Vrms
    return Vrms;
}

/*read reference voltage*/
long readVref()
```

```
{
    long result;
#if defined(__AVR_ATmega168__) || defined(__AVR_ATmega328__) || defined
(__AVR_ATmega328P__)
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#elif defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) ||
defined(__AVR_ATmega2560__) || defined(__AVR_AT90USB1286__)
    ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    ADCSRB &= ~_BV(MUX5);   // Without this the function always returns -1 on the
ATmega2560 http://openenergymonitor.org/emon/node/2253#comment-11432
#elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) ||
defined(__AVR_ATtiny84__)
    ADMUX = _BV(MUX5) | _BV(MUX0);
#elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) ||
defined(__AVR_ATtiny85__)
    ADMUX = _BV(MUX3) | _BV(MUX2);
#endif
#if defined(__AVR__)
    delay(2);                                       // Wait for Vref to settle
    ADCSRA |= _BV(ADSC);                            // Convert
    while (bit_is_set(ADCSRA, ADSC));
    result = ADCL;
    result |= ADCH << 8;
    result = 1126400L / result;          //1100mV*1024 ADC steps
http://openenergymonitor.org/emon/node/1186
    return result;
#elif defined(__arm__)
    return (3300);                                  //Arduino Due
#else
    return (3300);                                  //Guess that other un-
supported architectures will be running a 3.3V!
#endif
}
```

函数功能说明： float readDCCurrent(int Pin),该函数用来测量直流电流。float readACCurrent(int Pin),该函数用来测量交流电流，测得的是交流电流的有效值。 根据被测电流，选择相应的函数调用即可，不能两个函数同时调用。long readVref()，该函数用来读取参考电压。

结果：串口上显示当前电机的直流电压值。

点阵模块



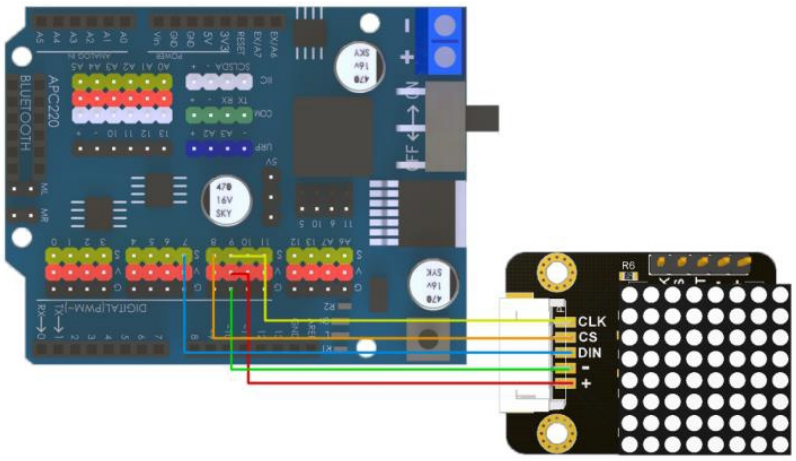
模块介绍：MAX7219是一种集成化的串行输入/输出共阴极显示驱动器，它连接微处理器与8位数字的7段数字LED显示，也可以连接条线图显示器或者64个独立的LED。

输入类型：数字输入
接口模式：pH2.0-5P

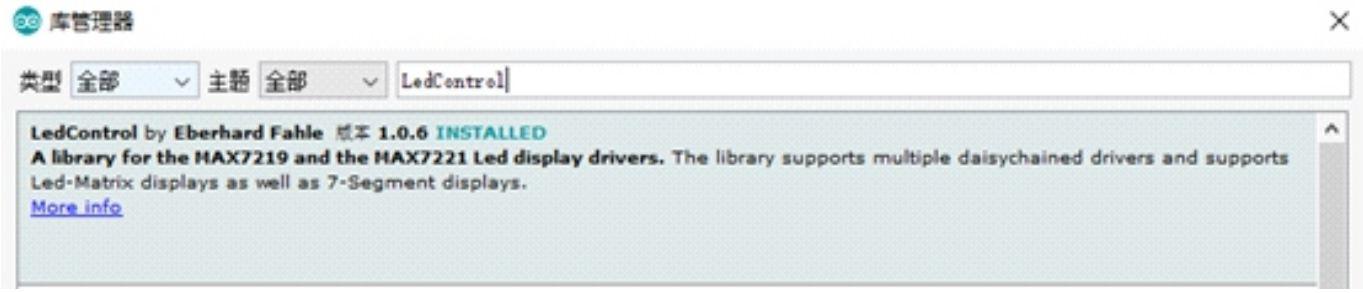
引脚定义：

CLK	时钟序列输入
CS	载入数据
DIN	行数据输入
-	GND
+	VCC
DOUT	串行数据输出

Arduino
例子一
接线图



安装库：IDE ->项目->加载库->管理库->搜索“LedControl”->安装



LedControl库的常用函数讲解：

LedControl（int dinpin, int clkpin, int cspin, int numDevices）

用途：初始化设备，定义一个对象，设置DIN、CLK、CS的I/O口及连接设备数量，连接数量只能设置1-8，超过8个设备需另外定义一个对象及使用另外的IO口。

Void shutdown(int addr, bool status) 用途：设置（节电）模式。第一个参数表需设置的设备号，如第一个设备为0，第二个设备为1等等。第二个参数如为true则开启节电模式，为false则关闭。

Void setIntensity(int addr, int intensity) 用途：设置亮度。第一个参数是设备号，第二个参数是亮度值，亮度值在0-15。

Void clearDisplay(int addr) 用途：清屏。参数是设备号。

Void setRow(int addr, int row, byte value) 用途：设置一行8个led的开关状态。第一个参数是设备号，第二个参数是行号（0-7），第三个参数是led亮灭数据，1为亮，0为灭，可只输入一个或者多个数字（小于等于8个），即采用二进制来表示，例如表示一行的亮灭是“亮亮灭灭亮亮亮灭”，则二进制表示是“B11001110”。LED按顺序亮灭，为输入部分默认为灭，还可以输入一个十六进制的数值来表示了led的亮灭。利用下表，8个led的亮灭可用对照获得led灯的亮灭十六进制数，例如：表示一行的亮灭是“亮亮灭灭亮亮亮灭”，则二进制表示是“0xCE”。为了方便编辑，可以采用数组形式存储8*8led亮灭数据。

二进制和十六进制对照表

十六进制	0	1	2	3	4	5	6	7
二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	8	9	A	B	C	D	E	F
二进制	1000	1001	1010	1011	1100	1101	1110	1111

```
#include <LedControl.h>

int DIN = 7;
int CS = 8;
int CLK = 9;


LedControl lc=LedControl(DIN,CLK,CS,4);

void setup(){
  lc.shutdown(0,false);          //启动时，MAX7219 处于省电模式
  lc.setIntensity(0,8);         //将亮度设置为最大值
  lc.clearDisplay(0);            //清除显示
}

void loop(){
      byte smile[8]=     {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C};//笑脸
      byte neutral[8]= {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C};//标准脸

      printByte(test);//显示 8
      delay(1000);//延时 1 秒
      printByte(neutral);//显示标准脸
      delay(1000);
}

//点阵显示函数
void printByte(byte character [])
{
   int i = 0;
   for(i=0;i<8;i++)
   {
      lc.setRow(0,i,character[i]);
   }
}
```

结果：8*8点阵led上循环出现笑脸和标准脸。
扩展：多屏并排显示时，第一个模块的输入端接控制板，输出端接第二个模块的输入端，第二模块的输出端接第三个模块的输入端，以此类推。但注意，一个ledControl对象最多可以是连接8个设备模块。

触摸模块

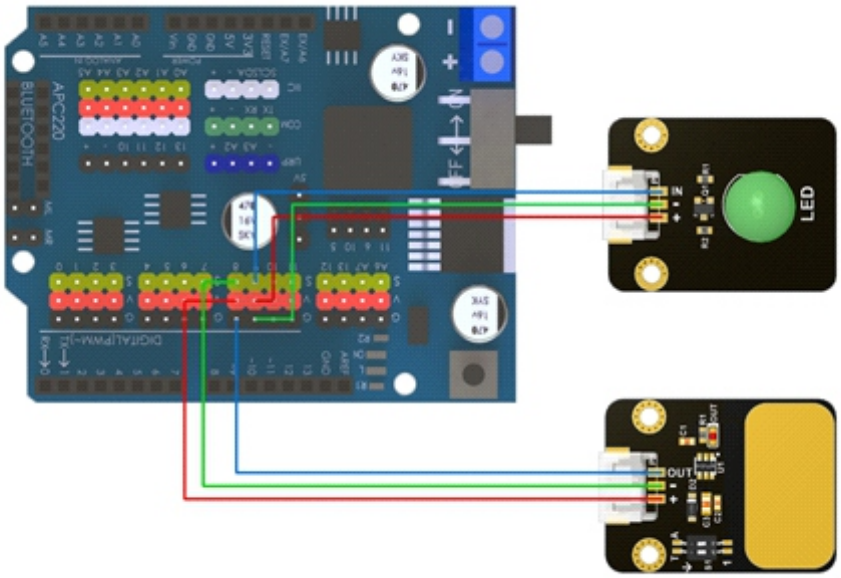
模块介绍：触摸模块是一个基于触摸检测的电容式点动型触摸开关模块。常态下，模块输出低电平；当用手指触摸相应位置时，模块会输出高电平。可以将模块安装在非金属材料如塑料、玻璃的表面，另外将薄薄的纸片（非金属）覆盖在模块的表面，只要触摸的位置正确，即可做成隐藏在墙壁、桌面等地方的按键。该模块可以让你免除常规按压型按键的烦恼。

工作电压：3.3V到5V
输出类型：数字信号
工作模式：4种
接口模式：pH2.0-3P

引脚定义：

OUT	输出
-	GND
+	VCC

Arduino uno
例子一
接线图



代码

```
void setup() {
   pinMode(8,INPUT);//触摸模块连接 8 引脚，并为输入模块
   pinMode(9,OUTPUT);//LED 连接 9 引脚，并为输出模块
}

void loop() {
   int state = digitalRead(8);//读取触摸模块电平
   if(state){//判断触摸模块电平高低
      digitalWrite(9,HIGH);//LED 亮
   }
   else{
      digitalWrite(9,LOW);//LED 灭
   }
}
```

Mixly图形化编程

结果：引脚8接入触摸模块，引脚9接入led灯。每当触摸触摸模块的时候，led灯会亮，直到不再触摸触摸模块时，led灯才会熄灭。

超声波模块

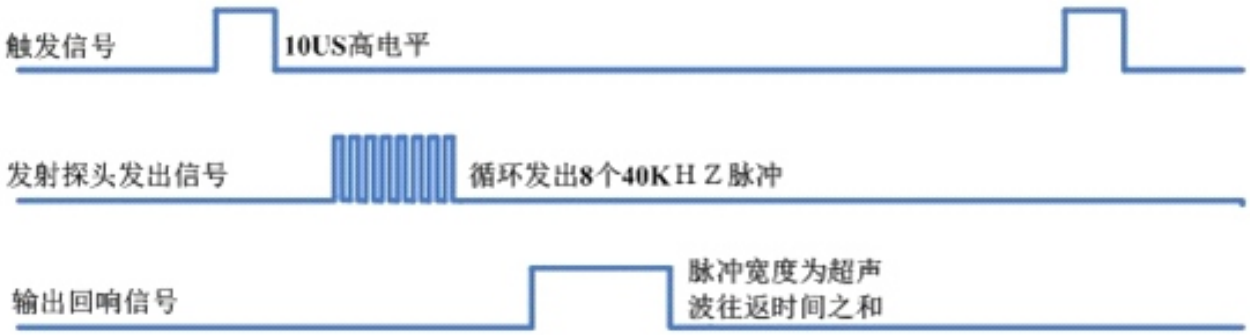
模块介绍：用来测量距离，通过发送和接收超声波，测量声音从物体上反弹并返回传感器所需的时间，利用时间差和声音传播速度，计算模块到前方障碍物的距离。

工作电压：3.3V或5V
最小测量距离测量：2cm
最大测量距离测量：350cm
类型：Echo是输入数字信号，Trig是输出数字信号。
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：86g

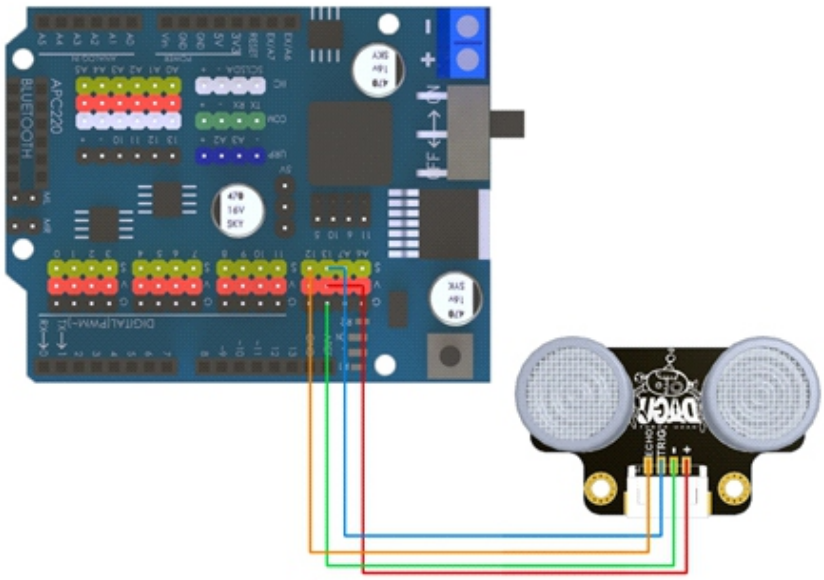
引脚定义：

Echo	接收输入
Trig	发送输出
-	GND
+	VCC

工作原理：向Trig管脚输入一个10uS以上的高电平，可触发模块测距。当测距结束时，Echo管脚会输出一个高电平，电平宽度为超声波往返时间之和。



Arduino uno
例子一
接线图



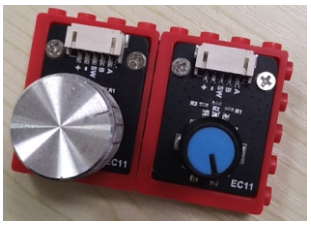
代码

```
void setup() {
   pinMode(13,OUTPUT);//发送连接 13 引脚，并为输出
   pinMode(12,INPUT);//接收连接 12 引脚，并为输入
   Serial.begin(9600);//打开波特率为 9600 串口
}

void loop() {
   digitalWrite(13,LOW);//触发超声波测距
   delayMicroseconds(2);
   digitalWrite(13,HIGH);
   delayMicroseconds(10);
   digitalWrite(13,LOW);
   int distance = pulseIn(12,HIGH)/58;//计算距离
   Serial.println(distance);//串口输出距离
   delay(200);//延时 200 毫秒
}
```

Mixly图形化编程

结果：引脚13接入Trig，引脚12接入Echo。打开串口，移动超声波测距模块前的障碍物，输出到串口的数据发生变化。
解释：pulseIn()返回的时间，是超声波发送到接收的时间，单位是微妙，而声速是340m/s，因为距离 = 速度*时间，所以距离（cm）= 340 *100/1000000 * pulseIn()/2，即距离（cm）= pulseIn()/58。

编码器模块



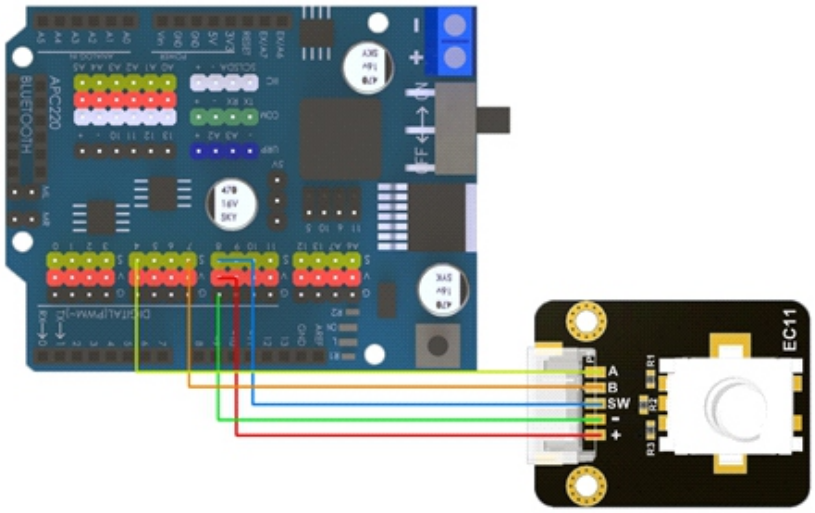
模块介绍：旋转编码器是用来测量转速并配合PWM技术可以实现快速调速的装置，光电式旋转编码器通过光电转换，可将输出轴的角位移、角速度等机械量转换成相应的电脉冲以数字量输出。分为单路输出和双路输出两种。单路输出是指旋转编码器的输出是一组脉冲，而双路输出的旋转编码器输出两组A/B相位差90度的脉冲，通过这两组脉冲不仅可以测量转速，还可以判断旋转的方向。该模块是双路输出。编码器输出信号中A、B两相的通道信号序列相位差90度，当主轴以顺时针方向旋转时，A通道位于B通道之前；当主轴以逆时针方向旋转时，A通道位于B通道之后。从而判断主轴是正转还是反转。

输出类型：PWM输出、数字输出。
接口模式：pH2.0-5P

引脚定义：

A（A 相）	PWM 输出
B（B 相）	PWM 输出
SW(开关)	数字输出
-	GND
+	VCC

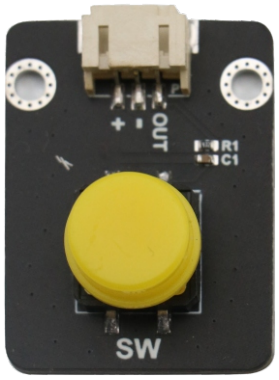
Arduino
例子一
接线图



```
int encoderPinA = 4;
int encoderPinB = 7;
int swpin =8;
boolean encoderALast = LOW;
int count=0;
boolean oldstate;
boolean state;
void setup()
{
   pinMode(encoderPinA, INPUT);
   pinMode(encoderPinB, INPUT);
   pinMode(swpin,INPUT);
   digitalWrite(swpin,HIGH);
   oldstate =digitalRead(swpin);
   Serial.begin (9600);
}

void loop()
{
   state =digitalRead(swpin);
   if(oldstate!=state && state==LOW){
      count=0;
      Serial.println(count);
   }
   oldstate =state;
   boolean encoderA = digitalRead(encoderPinA);
   if ((encoderALast == HIGH) && (encoderA == LOW))
   {
      if (digitalRead(encoderPinB) == LOW)
      {
         count--;
      }
      else
      {
         count++;
      }
      Serial.println(count);
   }
   encoderALast = encoderA;
}
```

结果：模块A接口接入4引脚，模块B接口接入引脚7，模块SW接口接入引脚8。当编码器顺时针旋转时，增加计数；当编码器逆时针旋转时，减小计数；当按下编码器，清零重新计数。

按键模块



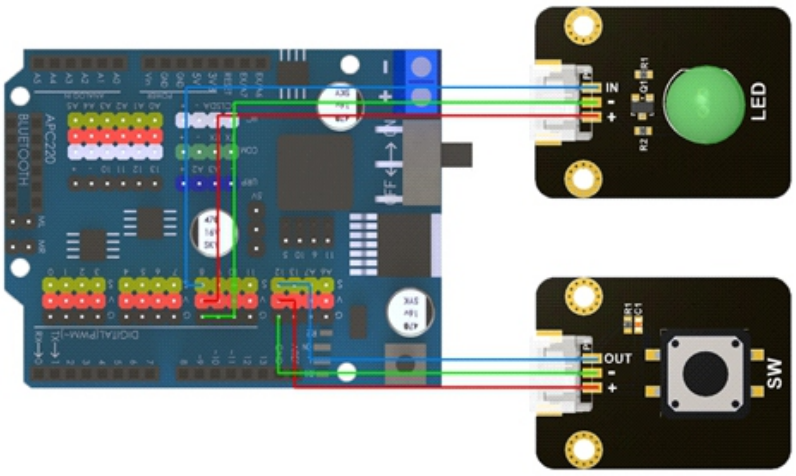
模块介绍：按钮是用于控制电子设备的常见组件。他们是通常用作连接或断开电路的开关。在正常情况下，按钮的两个触点处于断开状态，只有按下按钮时它们才闭合。

工作电压：3.3V到5V
输出类型：数字信号。按键按下，高电平；按键弹起，低电平。
接口模式：pH2.0-3P
模块尺寸：35*26.3mm
模块重量：49g

引脚定义：

OUT	输出
+	VCC
-	GND

Arduino uno
样例一
接线图



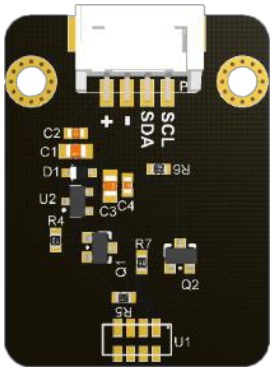
代码

```
int buttonpin = 8;                  // 按键连接引脚 8
int LEDpin = 12;              // LED 的引脚 12

void setup() {
   pinMode(LEDpin, OUTPUT);                  // 定义 LED 的引脚为输出引脚
   pinMode(buttonpin,INPUT);                        // 定义按键引脚为输入引脚
}

void loop() {
   int state = digitalRead(buttonpin);                 //读取输出值
   if(state == HIGH ){  // 检查输入是否为高，这里高为按下
      digitalWrite(LEDpin, HIGH);     // 灯亮起状态
   }
   else{
      digitalWrite(LEDpin, LOW);                  // 灯关闭状态
   }
}
```

Mixly图形化编程

结果：当你按下按键时，板子上的13号引脚的LED将会被点亮，松开后，灯熄灭。

环境光模块



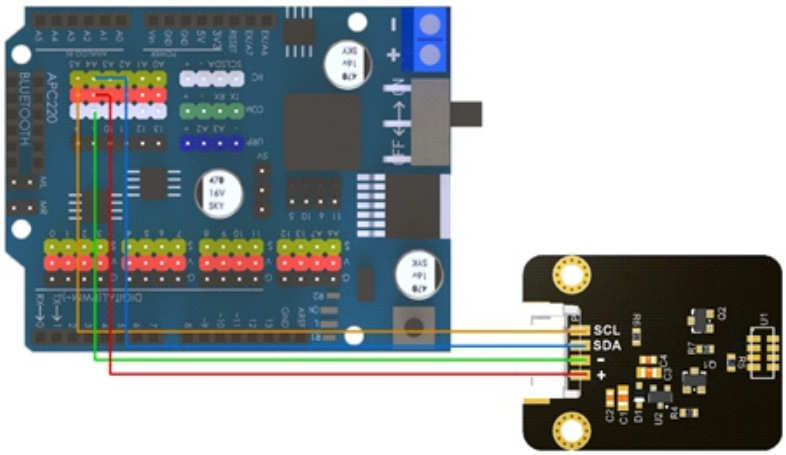
模块介绍：测量不同环境下光的强度，在串口监视器中以数值显示，单位是lx。它的测量精度高，量程较广，探测范围从0lx到120klx，分辨率0.0036lx/ct。实际使用中可测得大于120klx的环境光，但是精度不高。

输出类型：数字量输出
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND

Arduino
接线图



代码

```
#include <Wire.h>
#include "DFRobot_VEML7700.h"

DFRobot_VEML7700 als;

void setup()
{
    Serial.begin(9600);
    als.begin();
}

void loop()
{
    float lux;

    als.getALSLux(lux);
    Serial.print("Lux:");
    Serial.print(lux);
    Serial.println(" lx");

    delay(200);
}
```

结果：再不同环境光强度下，串口窗口输出对应的光强度。

U型测速模块

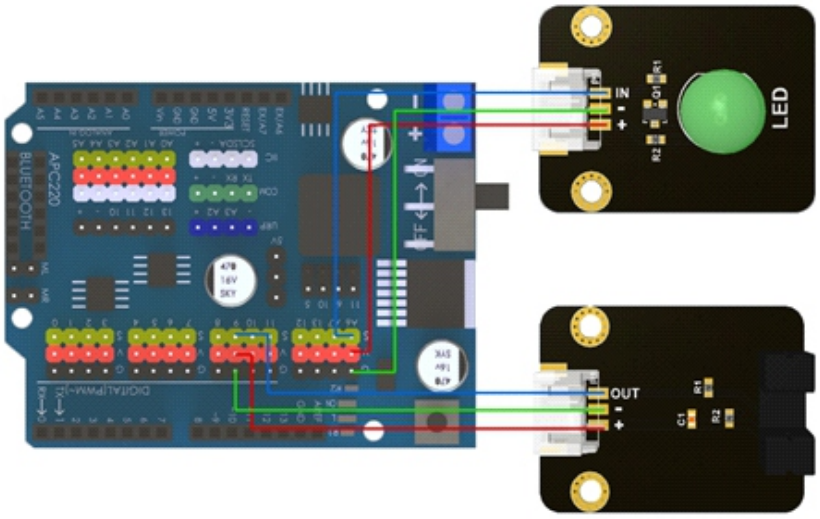
模块介绍：U型槽的两边分别存在一个光电传感器和一个光敏晶体管。接收管的导通状态决定了模块是输出高电平还是低电平。然后通过一定时间内电平发生改变的次数来推断电机的转速。这个模块被广泛用于电机转速检测、脉冲计数、位置限位等。

输出类型：数字输出。
接口模式：pH2.0-3P

引脚定义：

Out	数字输出
-	GND
+	VCC

Arduino
例子一
接线图



代码

```
void setup() {

pinMode(9,INPUT);
pinMode(13,OUTPUT);
Serial.begin(9600);

}

void loop() {
   // put your main code here, to run repeatedly:
if(digitalRead(9))
digitalWrite(13,HIGH);
else
digitalWrite(13,LOW);
}
```

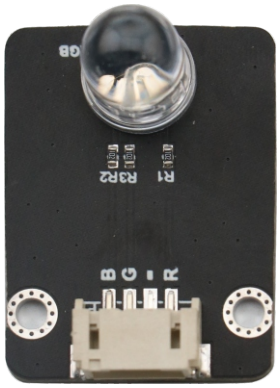
结果：模块接入引脚9，当有阻碍物阻挡在U型之间时，板子上的led秒掉，直到阻碍物移开U型之间，LED灯再次亮起。

例子二

```
float val = 0;
float time;
float Speed;
void setup() {
attachInterrupt(0,count,CHANGE);
//pinMode(2,INPUT);
pinMode(5,OUTPUT);
pinMode(10,OUTPUT);
pinMode(13,OUTPUT);
Serial.begin(9600);

}

void loop() {
  digitalWrite(10, HIGH);
  analogWrite(5, 50);
  time =millis();
  Speed = (val/40)/(time/60000);
  Serial.println(Speed);
  if(digitalRead(2))
    digitalWrite(13,HIGH);
  else
    digitalWrite(13,LOW);
}

void count(){
   val+=1;
}
```

结果：模块接入引脚2，使用中断只能接引脚2、3，20孔的码盘和马达安装在一起，打开串口，显示的数据便是马达的速度，单位是转/分钟

RGB 模块



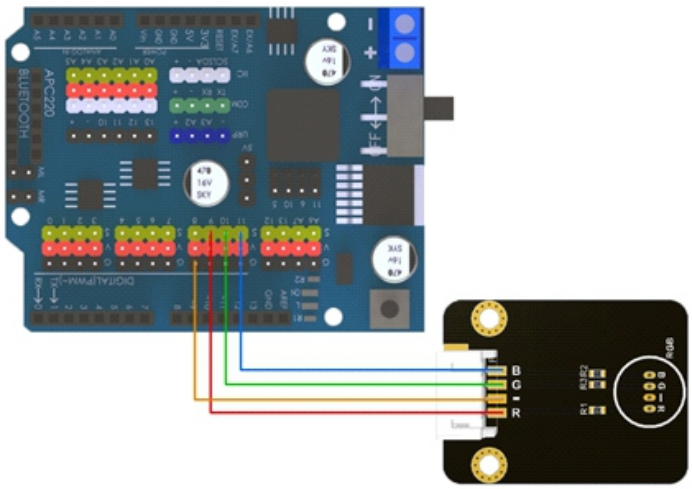
模块介绍：RGB 模块由三个LED组成。每个LED实际上有一个红光、一个绿光和一个蓝光。这三种颜色的LED能够产生任何颜色。RGB LED带有红色、绿色和蓝色发射器，通常使用三条线与一根公共导线（阳极或阴极）连接，该模块是共阴极LED。

工作电压：.3V到5V
输入类型：PWM，值（0-255）作为数字值分别输入RGB三个接口。通过PWM来控制RGB发光二极管的颜色。
接口模式：pH2.0-4P
模块尺寸：35*26.3mm
模块重量：49g

引脚定义：

B	蓝色输入
G	绿色输入
-	GND
R	红色输入

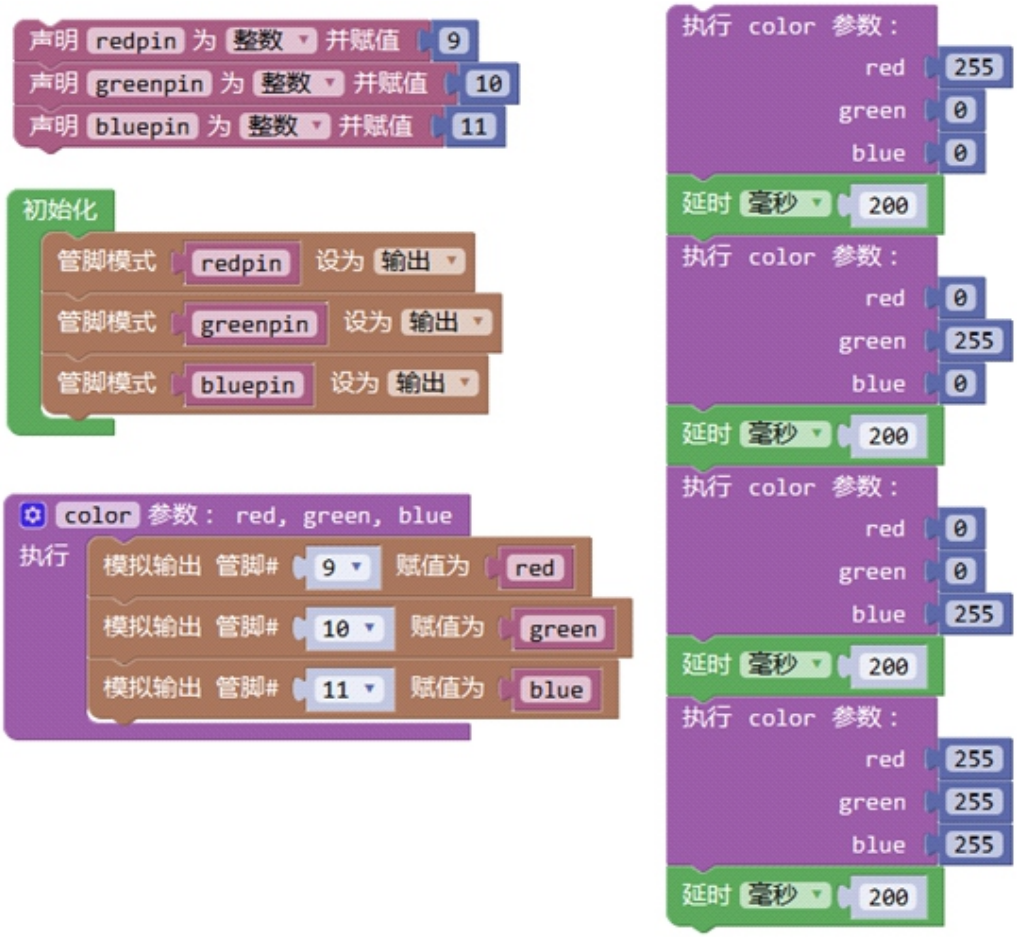
Arduino uno
样例一
连接图



代码

```
int redpin = 9;          //RGB LED 红灯的引脚 9
int greenpin = 10;       //RGB LED 绿灯的引脚 10
int bluepin = 11;        //RGB LED 蓝灯的引脚 11
void setup() {
   pinMode(redpin,OUTPUT); // 定义 LED 的三种颜色引脚为输出引脚
   pinMode(greenpin,OUTPUT);
   pinMode(bluepin,OUTPUT);
}

void loop() {
   color(255,0,0);
   delay(200);
   color(0,255,0);
   delay(200);
   color(0,0,255);
   delay(200);
   color(255,255,255);
   delay(200);
}

//定义函数，分别定义写入输出值
void color(int red,int green ,int blue){
   analogWrite(redpin,red);
   analogWrite(greenpin,green);
   analogWrite(bluepin,blue);
}
```

Mixly图形化编程



结果：模块的R接入引脚9，G接入引脚10，B接入引脚11的RGB led灯以红绿蓝白为周期循环闪烁。

PS摇杆模块

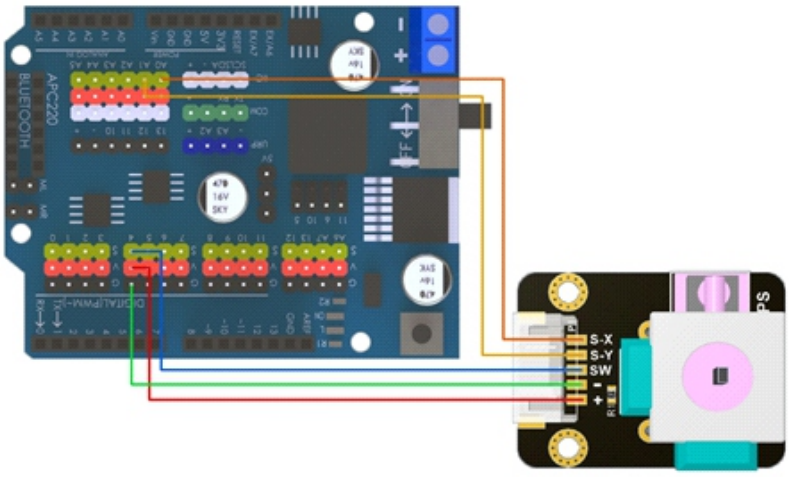
模块介绍：由两个滑动变阻器和一个按键组成，当拨动摇杆时，滑动变阻器的阻值就发生变化，对应的X/Y电压值也随之变化，而用力按下摇杆就会触发按键按下，对应的SW信号变为低电平。通常是将两个规格相同的电位器装在同摇杆电位器结构转轴上，调节转轴时，两个电位器的滑动触点异步转动。这个结构被称为同轴异步双联动的电位器。

输出类型：数字信号、模拟信号。
接口模式：pH2.0-5P

引脚定义：

S-X	模拟输入
S-Y	模拟输入
SW	数字输入
+	GND
-	VCC

Arduino
样例一
接线图



代码

```
int x,y,sw;
void setup() {
   pinMode(A0,INPUT);
   pinMode(A1,INPUT);
   pinMode(4,INPUT);
   Serial.begin(9600);
}

void loop() {
   x = analogRead(A0);
   y = analogRead(A1);
   sw = digitalRead(4);
   if(sw){
      digitalWrite(13,HIGH);
   }
   Serial.print("x:");
   Serial.println(x);
   Serial.print("y:");
   Serial.print(y);
   Serial.println();
}
```

结果：串口上不断地显示当前摇杆的X和Y的值，当向下按下摇杆时，板子上的led灯亮；反之led灯不亮。

LED模块

（红黄绿蓝白）

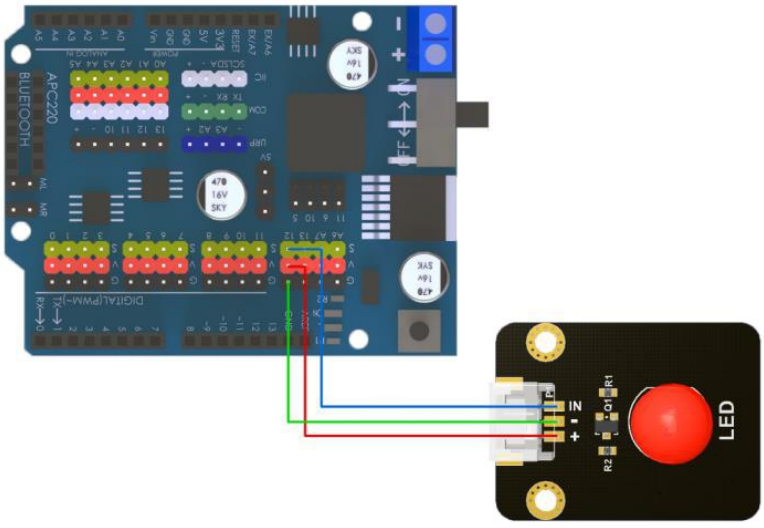
模块介绍：LED是发光二极管的缩写。它通常是由砷化镓，磷化镓半导体材料。LED有两个电极，一个正极和一个负极，当正向电流通过时，它就会发光。它可以是红、蓝、绿或黄光，光的颜色取决于它所用的材料。

工作电压：3.3V到5V
输入类型：数字信号，对于0V到5V的模拟电压，值（0-255）也允许作为数字值输入。
接口模式：pH2.0-3P

引脚定义：

IN	输入
+	VCC
-	GND

Arduino uno
样例一
连接图



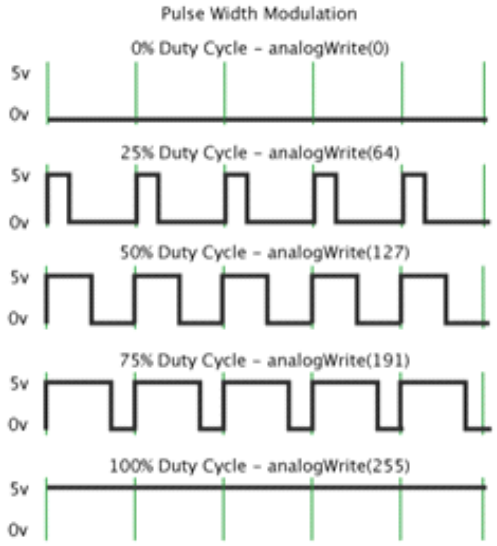
代码

```
int LEDpin = 12;      // LED 的引脚 12

void setup() {
   pinMode(LEDpin, OUTPUT);  // 定义 LED 的引脚为输出引脚
}

void loop() {
   digitalWrite(LEDpin, HIGH);    //写入输出值为高
   delay(1000);
   digitalWrite(LEDpin, LOW);    //写入输出值为低
   delay(1000);
}
```

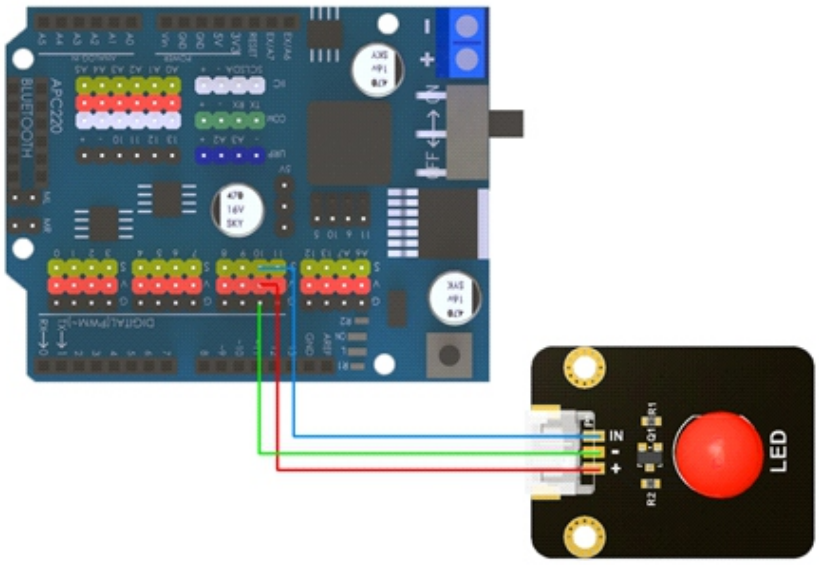
Mixly图形化编程

结果：引脚12插上LED灯后，LED灯会循环的亮1秒灭1秒。

脉冲宽度调制（Pulse Width Modulation，或称为PWM）是一种获得模拟结果的技术数字手段。数字控制用来产生一个方波，一个信号交换介于开和关之间。这种开关模式可以模拟电压之间的电压打开（5伏）和关闭（0伏）。通过改变信号的时间部分，持续时间与信号停止时间的对比。电压打开的持续时间称为脉冲宽度。要获得不同的模拟值，可以更改脉冲宽度。例如LED，其结果就像信号是介于0和5v之间的稳定电压，来控制LED的亮度。
在下图中，绿线表示一个固定的时间段。这个持续时间或周期是脉宽调制频率的倒数。换句话说，与Arduino的脉冲宽度调制频率约为500赫兹，绿线测量值为每次20毫秒。调用analog Write(pin,value),其中value取值在0-255，这样analog Write(pin,255)请求100%占空比（始终打开），并且例如，analog Write（pin,127）是50%的占空比（一半时间）。



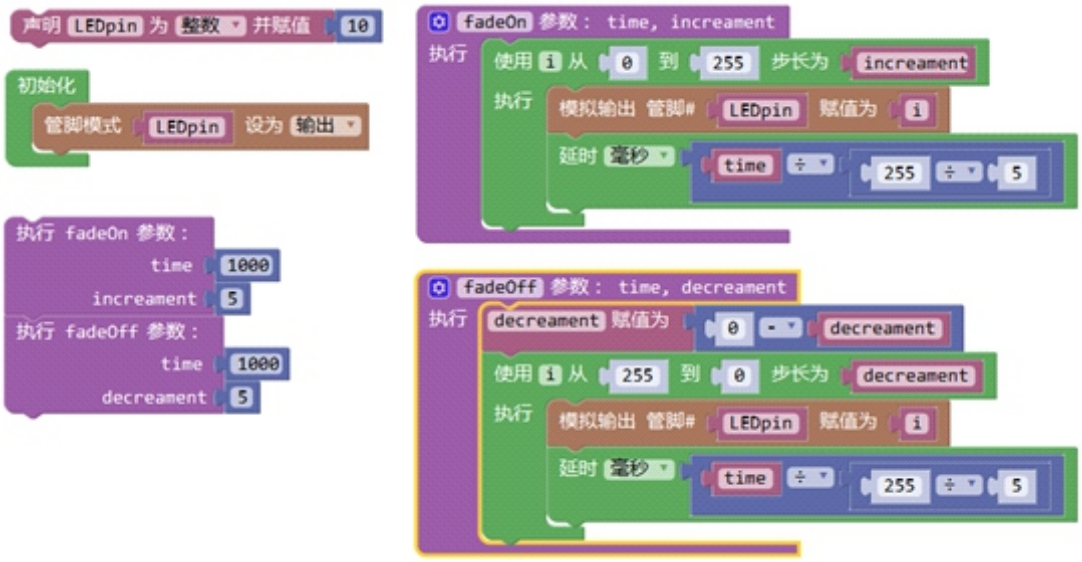
样例二
连接图



代码

```
int ledPin = 10;      // LED 的引脚 10

void setup() {
  pinMode(ledPin,OUTPUT);  // 定义 LED 的引脚为输出引脚
}
void loop(){
fadeOn(1000,5);
fadeOff(1000,5);
}

//定义函数，LED 模拟输出值按自定义 increament 的逐增
void fadeOn(unsigned int time,int increament){
  for (byte value = 0 ; value < 255; value+=increament){
  analogWrite(ledPin, value);
  delay(time/(255/5));
  }
}

//定义函数，LED 模拟输出值按自定义 decreament 的逐减
void fadeOff(unsigned int time,int decreament){
  for (byte value = 255; value >0; value  =decreament){
  analogWrite(ledPin, value);
  delay(time/(255/5));
  }
}
```

Mixly图形化编程



结果：引脚10上的led灯有个逐渐由亮到灭的一个缓慢过程，而不是直接的亮灭。

6812RGBW模块

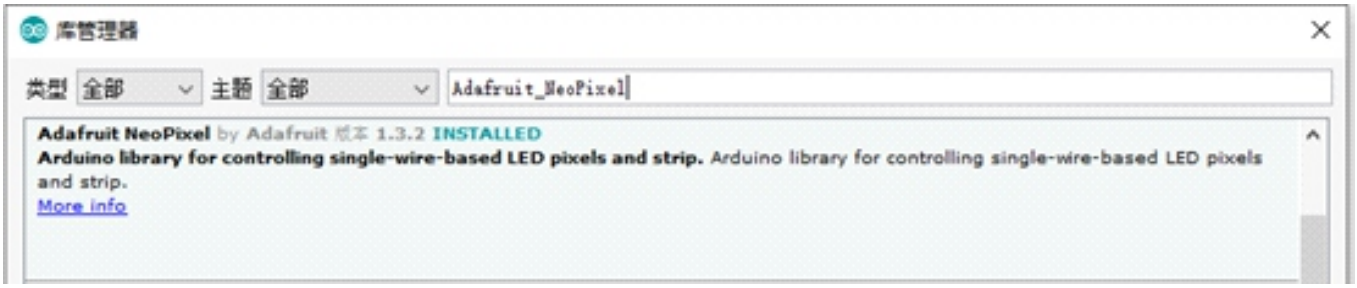
模块介绍：SK6812RGBW是一个集控制电路与发光电路于一体的智能外控LED光源。其外型与一个5050LED灯珠相同，每个元件即为一个像素点。主要适用于Led全彩灯串、Led全彩模组、Led幻彩软硬灯带、Led护栏管、Led外观/场景照明、Led点光源、Led像素屏、Led异形屏、各种电子产品、电器设备跑马灯。

输入类型：任何引脚都可输入·。
接口模式：pH2.0-3P

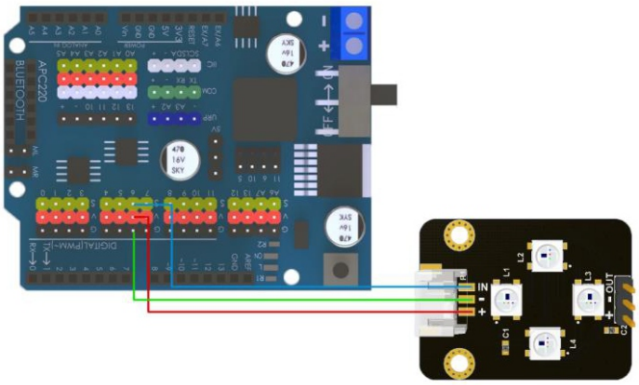
引脚定义：

IN	输入引脚
-	GND
+	VCC

Arduino
安装Adafruit_NeoPixel库



样例一
接线图



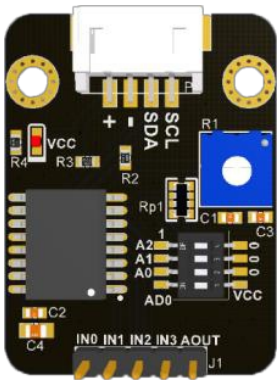
代码

```
#include <Adafruit_NeoPixel.h>

#define PIN 6
#define BRIGHTNESS 4

Adafruit_NeoPixel strip = Adafruit_NeoPixel(BRIGHTNESS, PIN, NEO_GRBW + NEO_KHZ800);

void setup() {
  strip.setBrightness(BRIGHTNESS);
  strip.begin();
  strip.show();
}

void loop() {
  colorWipe(strip.Color(150, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 150, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 150), 50); // Blue
  colorWipe(strip.Color(150, 150, 150), 50); // BlueWite
  rainbowCycle(1);

}

void colorWipe(uint32_t c, uint8_t wait) {
  for (uint16_t i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
```

```
        delay(wait);
    }
}

void rainbow(uint8_t wait) {
    uint16_t i, j;
    for (j = 0; j < 256; j++) {
        for (i = 0; i < strip.numPixels(); i++) {
            strip.setPixelColor(i, Wheel((i + j) & 255 ));
        }
        strip.show();
        delay(wait);
    }
}

void rainbowCycle(uint8_t wait) {
    uint16_t i, j;
    for (j = 0; j < 256 * 5; j++) { // 5 cycles of all colors on wheel
        for (i = 0; i < strip.numPixels(); i++) {
            strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
        }
        strip.show();
        delay(wait);
    }
}

uint32_t Wheel(byte WheelPos) {
    if (WheelPos < 85) {
        return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
    } else if (WheelPos < 170) {
        WheelPos -= 85;
        return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
    } else {
        WheelPos -= 170;
        return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
    }
}
```

结果：模块的四个rgbw是以红、绿、蓝、蓝白、彩虹循环为一个循环周期循环显示。

PCF8591 AD DA转换模块



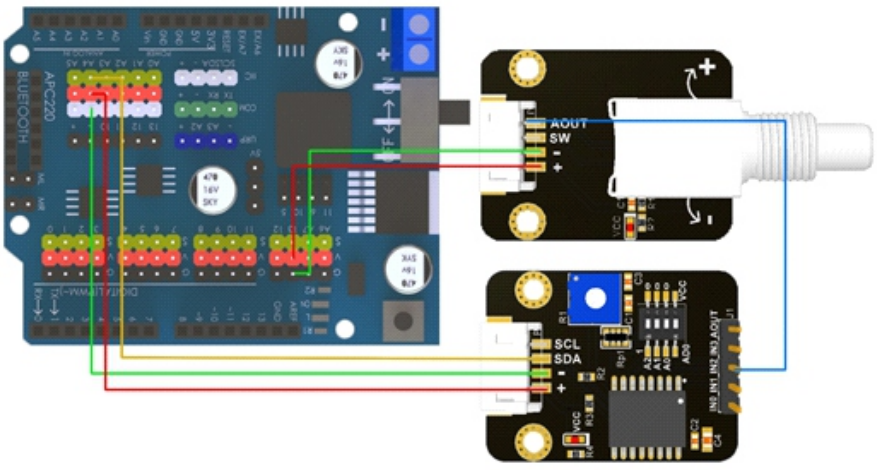
模块介绍：AD转换的功能是把模拟量电压转换为数字量电压。DA转换的功能正好相反，就是讲数字量转换位模拟量。PCF8591是一个单片集成、单独供电、低功耗、8-bit CMOS数据获取器件。PCF8591具有4个模拟输入、1个模拟输出和1个串行I²C总线接口。PCF8591的3个地址引脚A0，A1和A2可用于硬件地址编程，允许在同个I2C总线上接入8个PCF8591器件，而无需额外的硬件。在PCF8591器件上输入输出的地址、控制和数据信号都是通过双线双向I²C总线以串行的方式进行传输。

输出类型：IIC协议
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND

Arduino
连接图



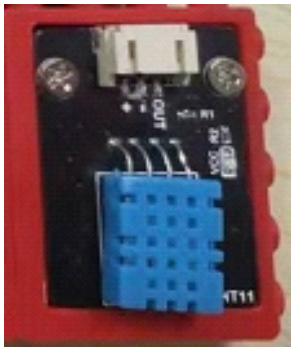
代码

```
#include <Wire.h>
void setup()
{
   Wire.begin();                          // join i2c bus (address optional for master)
   Serial.begin(9600);                  // start serial communication at 9600bps
   float reading = 0,date=0;
}

void loop(){
   Wire.requestFrom(0x4F,8);
   Wire.beginTransmission(0x4F);
   Wire.write(byte(0x02));//0x00==1;0x01==1;0x02==2;0x03==3
   Wire.endTransmission();
   while( Wire.available())
   {
     int reading =Wire.read();
   int date=(reading*5)/256;
   Serial.print(reading);
      Serial.print("       ");
   Serial.print(date);
   Serial.println("v");
   }
}
```

结果：串口窗口显示出读取AIN2引脚上的模拟值。
可以使用库文件，参考库下载地址：

https://github.com/xreef/PCF8591_library/blob/master/PCF8591.cpp

DHT11-温湿度模块



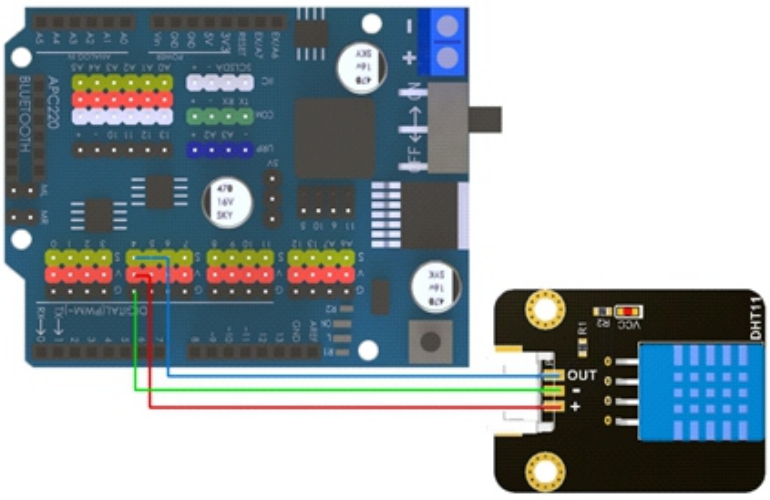
模块介绍：是用于测量温度和湿度相对便宜的传感器。

输出类型：已校准数字信号输出。
接口模式：pH2.0-3P

引脚定义：

Out	串行数据，单总线
-	GND
+	VCC

Arduino
接线图



添加dth11库，打开连接 https://playground.arduino.cc/Main/DHT11Lib/ ，新建dht11.h和dht11.cpp，分别复制的dht11.h和dht11.cpp处内容到对应位置。

样例一

```
#include <dht11.h>                          //引用 dht11 库文件，使得下面可以调用相
关参数
#define DHT11PIN 4                          //定义温湿度针脚号为 4 号引脚
dht11 DHT11;                                //实例化一个对象

void setup() {
  Serial.begin(9600);                       //设置波特率参数
  pinMode(DHT11PIN, OUTPUT);                //定义输出口
}

void loop() {
  int chk = DHT11.read(DHT11PIN);           //将读取到的值赋给 chk

  float tem = DHT11.temperature;            //将温度值赋值给 tem
  float hum = DHT11.humidity;               //将湿度值赋给 hum
  Serial.print("Tempeature:");              //打印出 Tempeature:
  Serial.println(tem);                      //打印温度结果

  Serial.print("Humidity:");                //打印出 Humidity:
  Serial.print(hum);                        //打印出湿度结果
  Serial.print("%");                        //打印出%

  delay(1000);                              //延时一段时间

}
```

结果：串口会每个一秒输出当前温度和湿度。

扩展模块

模块介绍：PCF8574 是CMOS电路，它通过两条双向总线可使大多数MCU实现远程I/O扩展。该器件包含一个8位准双向口（“准”就是“基本上”的意思，也就是“准双向口”不是真正的双向口。所以，有方向判断的才为真正的双向口）和一个总线接口（连接在总线上的设备与总线的连接电路称为总线接口）。pcf8574电流消耗很低，并且输出锁存，具有大电流驱动能力，可直接驱动LED。它还带有一种中断接线可与MCU的中断逻辑相连。通过INT发送中断信号，远端I/O口不必经过总线通信就通知MCU是否有数据从端口输入。这意味着 PCF8574 可以作为一个单被控器。

输出类型：IIC协议。
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND
P0	准双向 I/O 口 0
P1	准双向 I/O 口 1
P2	准双向 I/O 口 2
P3	准双向 I/O 口 3
P4	准双向 I/O 口 4
P5	准双向 I/O 口 5
P6	准双向 I/O 口 6
P7	准双向 I/O 口 7
INT	中断输入（低电平有效）

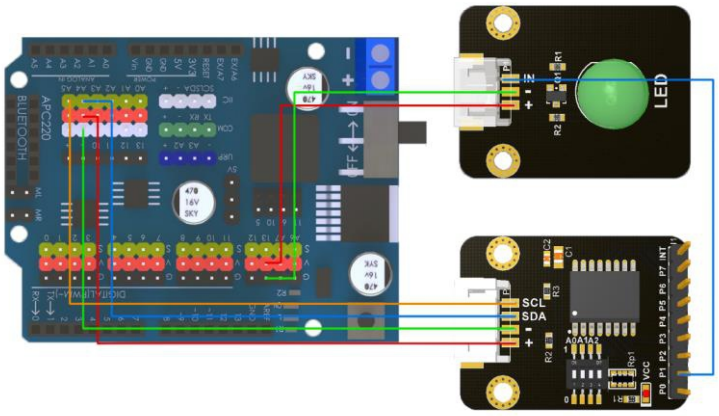
器件地址：
A0、A1、A2全部为高电，设备地址为0x3F
A0、A1、A2全部为低电平，设备地址为0x38
A0为高电平，A1、A2为低电平，设备地址为0x39
（其他情况类似，H为高电，L为低电平）这里的设备地址主要是用于I2C多机通信时区分设备用的。

Arduino
安装“PCF8574.h"，在https://github.com/skywodd/pcf8574_arduino_library下载库文件，解压压缩包，将PCF874文件夹复制到arduino ide安装地址下的librarise中。
拨动码盘，确定设备地址。如果上图看不懂或是不确认，可以连接好模块，上传下面代码，串口便输出设备地址

```
#include <Wire.h>

void setup() {
    Serial.begin (115200); // Leonardo: wait for serial port to connect
    while (!Serial) { }
    Serial.println ();
    Serial.println ("I2C scanner. Scanning ...");
    byte count = 0;
    Wire.begin();
    for (byte i = 8; i < 120; i++) {
        Wire.beginTransmission (i);
        if (Wire.endTransmission () == 0) {
            Serial.print ("Found address: ");
            Serial.print (i, DEC); Serial.print (" (0x");
            Serial.print (i, HEX); Serial.println (")");
            count++; delay (1); // maybe unneeded?
        } // end of good response
    } // end of for loop
    Serial.println ("Done.");
    Serial.print ("Found ");
    Serial.print (count, DEC);
    Serial.println (" device(s).");
} // end of setup

void loop() {}
```

当我们知道设备地址，便可通过IIC与模块通信

样例一
接线图

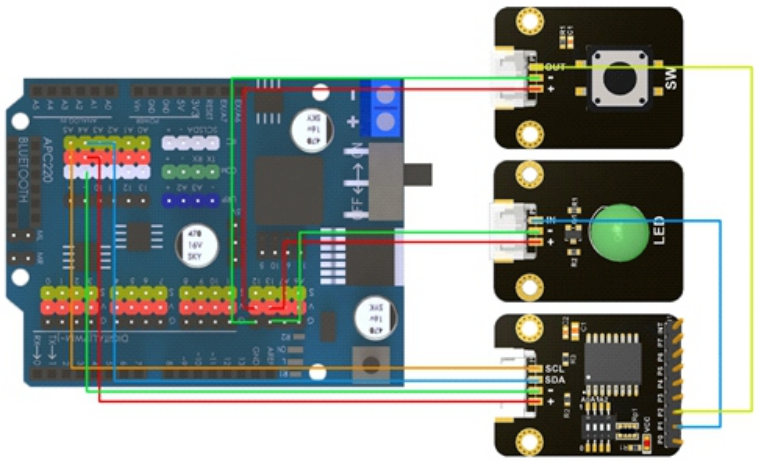


代码

```
#include <Wire.h>
#include "PCF8574.h"

PCF8574 expander;

void setup() {
  // put your setup code here, to run once:
  expander.begin(0x3f);//设备地址
  expander.pinMode(1,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:

  expander.digitalWrite(1,HIGH);
  delay(1000);
  expander.digitalWrite(1,LOW);
  delay(1000);
}
```

结果：连接模块上的led闪一秒，灭一秒，循环闪烁

样例二
接线图

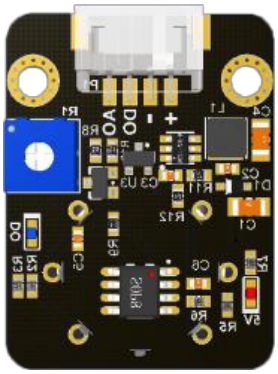


代码

```
#include <Wire.h>
#include "PCF8574.h"

PCF8574 expander;

void setup() {
   // put your setup code here, to run once:
   expander.begin(0x3f);
   expander.pinMode(1,OUTPUT);
   expander.pinMode(2,INPUT);
   expander.digitalWrite(2,HIGH);
   Serial.begin(9600);
}

void loop() {
   // put your main code here, to run repeatedly:
     if(!expander.digitalRead(2))
     {
       expander.digitalWrite(1,HIGH);
     }
     else{
       expander.digitalWrite(1,LOW);
     }
     Serial.println(expander.digitalRead(2));
}
```

结果：当按键按下时，led灯亮起，松开按键，led灯熄灭。

MQ系列模块



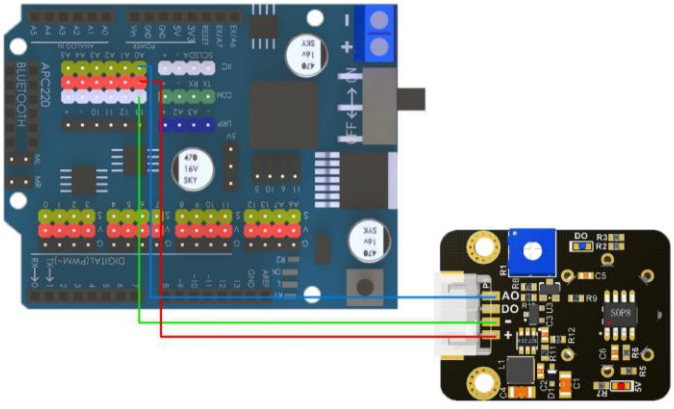
模块介绍：MQ-x系列是各种奇怪气体的探测器，煤气、天然气、打火机泄露可以探测，酒精、甲醛、可吸入悬浮物可以探测。

型号	探测气体	探测范围
MQ-2	烟雾	300-10000ppm
MQ-2B	可燃气体	300-10000ppm
MQ-5	液化气	300-5000ppm
MQ-7	一氧化碳	10-1000ppm
MQ-135	空气质量（二氧化碳、酒精、苯、氮氧化物、氨等气体的浓度）	

注意的是MQ-135可以检测到上面列出的所有气体，但无法区分它们。如果您要专门针对一种气体，则最好选择其他传感器。传感器还需要使用加热器来预热传感器。建议不要将其与较小的电池一起使用，因为它会很快耗尽电池电量。

AO	模拟 pin
DO	数字 pin
-	GND
+	VCC

Arduino
例子一
接线图



代码

```
const int gasSensor =A0;
void setup(){
   Serial.begin(9600);
}
void loop(){
   float voltage;
   voltage = analogRead(gasSensor) /204.6;
   //analogRead(gasSensor) * 0.004882814
   //此公式将 AnalogLead（）的值 0-1023 转换为 0.0 到 5.0 的值,反映所读取的真实电压
   Serial.println(voltage);
   delay(1000);
}
```

结果：模块的AO接口接入引脚A0，打开串口观察到从模块读取到的电压值。R1上的十字口可以调节DO接口检测灵敏度，直接读取DO口上的数字信号。

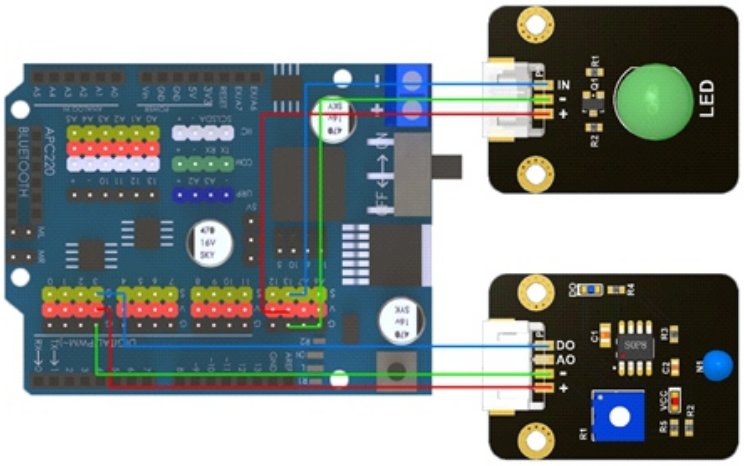
热敏模块

模块介绍：热敏电阻模块对环境温度很敏感，一般用来检测周围环境的温度。热敏电阻器是敏感元件的一类，按照温度系数不同分为正温度系数热敏电阻器（PTC）和负温度系数热敏电阻器（NTC）。热敏电阻器的典型特点是对温度敏感，不同的温度下表现出不同的电阻值。正温度系数热敏电阻器（PTC）在温度越高时电阻值越大，负温度系数热敏电阻器（NTC）在温度越高时电阻值越低，它们同属于半导体器件。该模块是负温度系数热敏电阻器，由于模块根据温度变化产生电阻阻值变化且是非线性变化，所以模块选用上拉电阻。

输出类型：数字、模拟输出。
接口模式：pH2.0-4P

DO	数字输出
AO	模拟输出
-	GND
+	VCC

Arduino
例子一
接线图



代码

```
void setup() {
   pinMode(3,INPUT);
   pinMode(13,OUTPUT);
}

void loop() {
   if (digitalRead(3)) {
      digitalWrite(13, LOW);
   }
   else {
      digitalWrite(13, HIGH);
      delay(2000);
   }
}
```

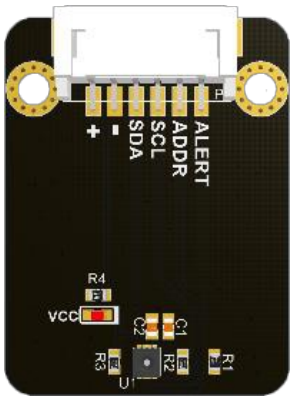
结果：DO输出端接入引脚3检测高低电平，由此来检测环境的温度改变；通过对电位器的调节，可以改变温度检测的阀值（即控制温度值），模块则在相应环境温度调到其led灯亮，DO则输出低电平，低于此设定温度值时，输出高电平，led灯不亮。如需要控制环境温度为具体数值时，建议使用模块的模拟输出更为准确。

例子二
接线图
代码

```
void setup() {
   pinMode(A0,INPUT);
   pinMode(13,OUTPUT);
   Serial.begin(9600);
}

void loop() {
   int value = analogRead(A0);
   Serial.println(value);
   if (value>400) {
      digitalWrite(13, LOW);
   }
   else {
      digitalWrite(13, HIGH);
      delay(2000);
   }
}
```

结果：AO输出端接入引脚A0检测高低电平，由此来检测环境的温度值。

SHT30数字温湿度模块



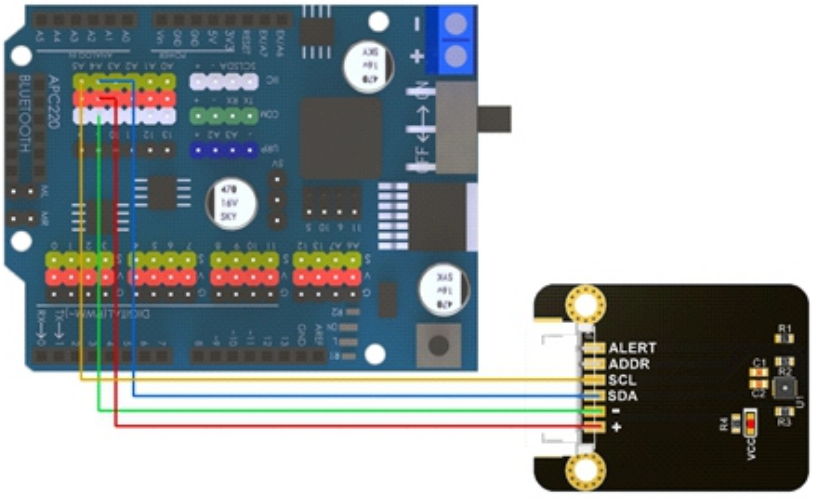
模块介绍：SHT30是湿度和温度传感器。SHT30测量数据经过出厂校正、线性化和温度补偿，具有温湿度报警输出、软硬件复位功能。传感器的湿度测量范围是0-100%RH，温度测量范围是-40-125℃。

输出类型：IIC协议
接口模式：pH2.0-6P

引脚定义：

SCL	时钟线
SDA	双向数据线
ADDR	地址引脚
ALERT	报警引脚
+	VCC
-	GND

Arduino
接线图



代码

```
#include <Wire.h>

// SHT30 I2C address is 0x44
#define Addr 0x44

void setup()
{
  // Initialise I2C communication as MASTER
  Wire.begin();
  // Initialise serial communication, set baud rate = 9600
  Serial.begin(9600);
  delay(300);
}

void loop()
{
  unsigned int data[6];

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Send measurement command
  Wire.write(0x2C);
  Wire.write(0x06);
  // Stop I2C transmission
  Wire.endTransmission();
  delay(500);

  // Request 6 bytes of data
  Wire.requestFrom(Addr, 6);

  // Read 6 bytes of data
  // cTemp msb, cTemp lsb, cTemp crc, humidity msb, humidity lsb, humidity crc
  if (Wire.available() == 6)
  {
    data[0] = Wire.read();
    data[1] = Wire.read();
    data[2] = Wire.read();
    data[3] = Wire.read();
    data[4] = Wire.read();
    data[5] = Wire.read();
  }

  // Convert the data
  float cTemp = ((((data[0] * 256.0) + data[1]) * 175) / 65535.0) - 45;
  float fTemp = (cTemp * 1.8) + 32;
```

```
    float humidity = ((((data[3] * 256.0) + data[4]) * 100) / 65535.0);

    // Output data to serial monitor
    Serial.print("Relative Humidity : ");
    Serial.print(humidity);
    Serial.println(" %RH");
    Serial.print("Temperature in Celsius : ");
    Serial.print(cTemp);
    Serial.println(" C");
    Serial.print("Temperature in Fahrenheit : ");
    Serial.print(fTemp);
    Serial.println(" F");
    delay(500);
}
```

结果：在串口窗口可以查看到当前湿度、华氏温度和摄氏温度

高精度时针模块

模块介绍：高精度的实时时钟，可以保持小时、分钟和秒，以及日、月和年等信息。此外，它还可以自动补偿闰年和少于31天的月份。该模块使用I2C通信协议。

传输类型：I2C通信协议
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
-	GND
+	VCC

Arduino
我们连接模块，我们需要对Arduino开发板进行编程以使用实时时钟。但是，在编写Arduino和I2C模块之间的通信时，代码并不是那么少而且容易。但是DS3231 RTC已经有几个库，可以在网上找到这些库。该模块我们选择使用Henning Karlsen制作的库文件，
可以在http://www.rinkydinkelectronics.com/library.php?id=74下载。

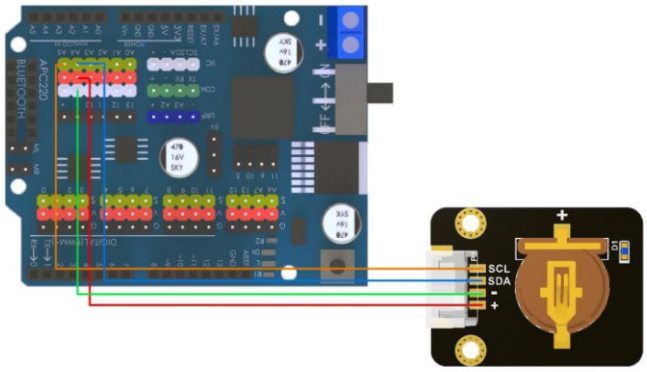
下载:

DS3231.zip
(文件大小为378.70 KiB。已下载163794次)

其中包括一些演示程序，以演示大多数功能。
这是一个多平台库，可与几种不同的开发板类型一起使用。

下载完成后解压压缩文件，并将解压的文件夹复制或剪贴至Arduino IDE安装地址下的libraries。重新打开ide即可使用DS3231库。

接线图



样例一
（利用库的示例，文件->示例->DS3231->Arduino->DS3231_Serial_Easy）

```
// DS3231_Serial_Easy
#include <DS3231.h>
DS3231  rtc(SDA, SCL);

void setup()
{
  Serial.begin(115200);
  rtc.begin();

  // The following lines can be uncommented to set the date and time
//  rtc.setDOW(MONDAY);        // Set Day-of-Week to SUNDAY
//  rtc.setTime(13, 18, 40);     // Set the time to 12:00:00 (24hr format)
//  rtc.setDate(18, 11, 2019);   // Set the date to January 1st, 2014
}

void loop()
{
  // Send Day-of-Week
  Serial.print(rtc.getDOWStr());
  Serial.print(" ");

  // Send date
  Serial.print(rtc.getDateStr());
  Serial.print(" -- ");

  // Send time
  Serial.println(rtc.getTimeStr());
```

```
    delay (1000);
}
```

先初始激活RTC模块的时钟，取消代码中下图代码注释，修改时针的数据，setDOW方法设置星期，srtTime方法设置时间（24小时制），serDate方法设置年月日。并上传代码。

```
//  rtc.setDOW(MONDAY);     // Set Day-of-Week to SUNDAY
//  rtc.setTime(13, 18, 40);     // Set the time to 12:00:00 (24hr format)
//  rtc.setDate(18, 11, 2019);   // Set the date to January 1st, 2014
```

设置好时钟之后重新注释上面3行代码，并重新上传代码。
结果：现在即使我们断开Arduino电源，然后重新连接并再次运行串口监视器，我们也可以注意到时间不会复位。

滚珠水平传感器模块

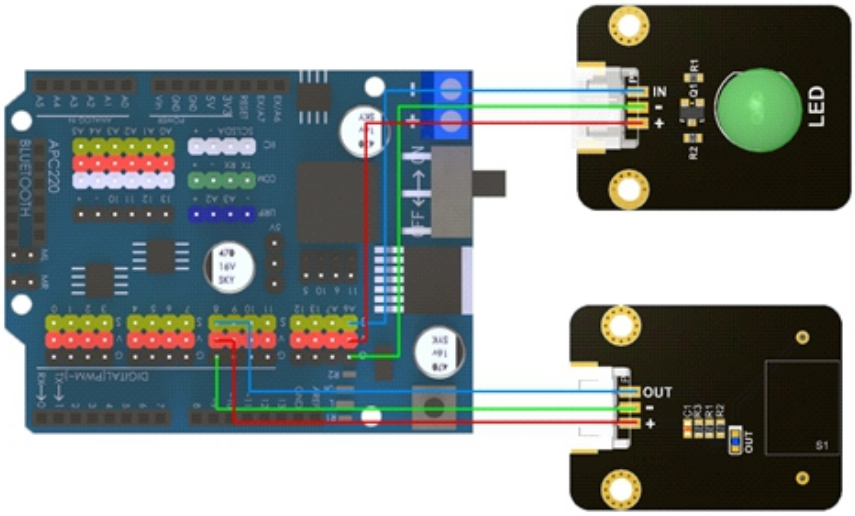
模块介绍：滚珠水平又称倾斜角度振动开关滚珠。开关里面有两个触点，通过珠子（单珠或者双珠）的倾斜15度而触发电路指令，大多应用在拥有智能开关这类产品上，比如烘干机防倒自动断电功能。

输出类型：数字输出。
接口模式：pH2.0-3P

引脚定义：

Out	数字输出
-	GND
+	VCC

Arduino
例子一
接线图

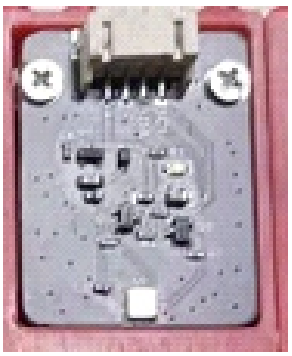


代码

```
void setup()
{
   pinMode(13,OUTPUT);
pinMode(8,INPUT);
Serial.begin(9600);
}
void loop()
{
   int state=digitalRead(8);
   Serial.println(state);
if(state)
      digitalWrite(13,HIGH);//点亮 led 灯
else//否则
      digitalWrite(13,LOW);//熄灭 led 灯
}
```

结果：当开关一端低于水平位置倾斜，开关寻通，点亮led灯。当另一端低于水平位置倾斜，开关停止，熄灭led 灯。

BMP180气压模块



模块介绍：如今在电子市场上，具有GPS功能的导航设备越来越多。借助全球卫星定位系统可以判定设备的经纬平面所处位置，但是无法得知目前的海拔高度。高精度数字传感器BMP180，适合智能的高精度测量和数据采集，输出高精度的压力（或高度）和温度测量数据。可与加速度计相互搭配，在没有GPS讯号的环境下，提供可靠的楼层侦测能力，以实现三维(3D)室内导航。

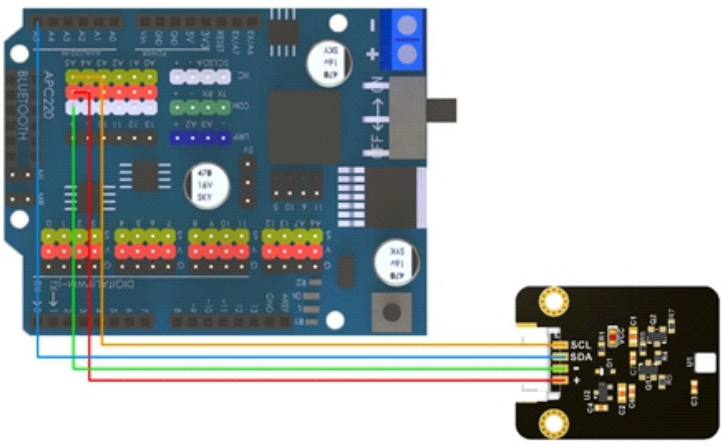
输出类型：i2C总线。
接口模式：pH2.0-4P

引脚定义：

SCL	串行时钟输入
SDA	串行接口数据输入/输出
-	GND
+	VCC

Arduino
添加SFE_BMP180库到Arduino ide
库下载地址：https://github.com/LowPowerLab/SFE_BMP180
打开示例FE_BMP180_example.ino进行学习

样例一
接线图



代码

```
#include <SFE_BMP180.h>

SFE_BMP180 AirPresure;
char presureDelayTime;
double presureP, presureT;

void setup() {
  Serial.begin(9600);
  AirPresure.begin();
}

void loop()
{
  presureDelayTime = AirPresure.startPressure(3);
  if (presureDelayTime != 0)
  {
    delay(presureDelayTime);
    presureDelayTime = AirPresure.getPressure(presureP, presureT);
    if (presureDelayTime != 0)
    {
      //当前气压
      Serial.print("Current Preasure: ");
      Serial.print(presureP);
      Serial.println(" bar");

      //换算成标准大气压
      Serial.print(presureP);
      Serial.print(" bar is");
      Serial.print(presureP / 1000.0);
      Serial.println(" atm");
    }
    else
    {
      Serial.println("ERROR");
    }
  }
  else
  {
    Serial.println("ERROR");
  }
  delay(1000);
}
```

结果：观察串口输出的当前气压和换算成标准大气压值。

水位传感器模块

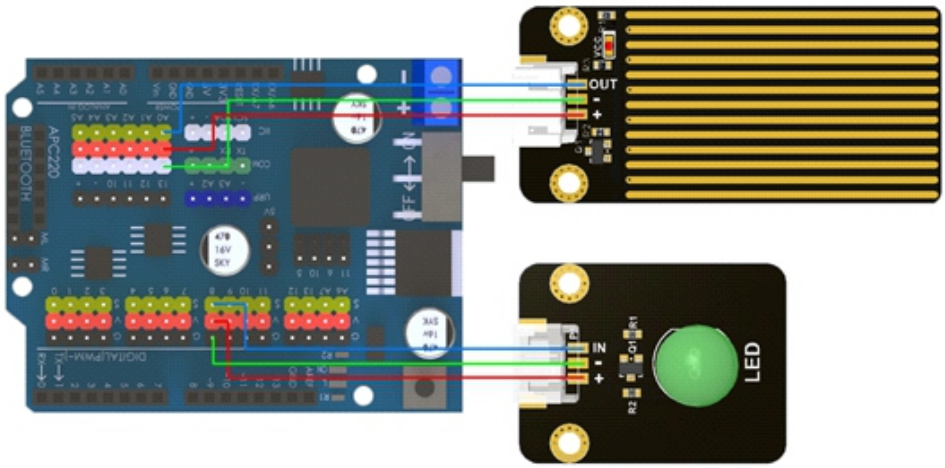
模块介绍：通过具有一系列的暴露的平行导线线迹测量其水滴/水量大小从而判断水位。轻松完成水量到模拟信号的转换，输出的模拟值可以直接被程序中函数所应用，达到水位报警的功效，低功耗，灵敏度是其又一大特点。

输出类型：模拟输出。
接口模式：pH2.0-3P

引脚定义：

Out	PIN
-	GND
+	VCC

Arduino
例子一
接线图

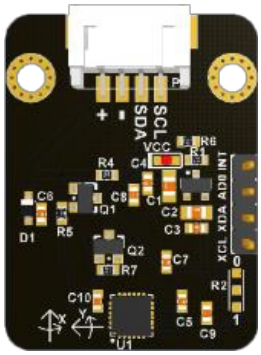


代码

```
int LEDPin=8;
void setup() {
Serial.begin(9600);
pinMode(LEDPin, OUTPUT);
pinMode(A0, INPUT);
}

void loop() {
int value=analogRead(A0);
Serial.println(value);
double data=(value/650)*4;
Serial.print("the depth is:");
Serial.print(data);
Serial.println("cm");
if(value>620)//这个值可自由设置
{
digitalWrite(LEDPin,HIGH);
}
else
{
digitalWrite(LEDPin,LOW);
}
delay(100);
}
```

结果：当水位传感器接收到水位到达一定的深度时，点亮led灯，当水位下降到某个高度一下时，led灯灭掉。

MPU6050陀螺仪模块



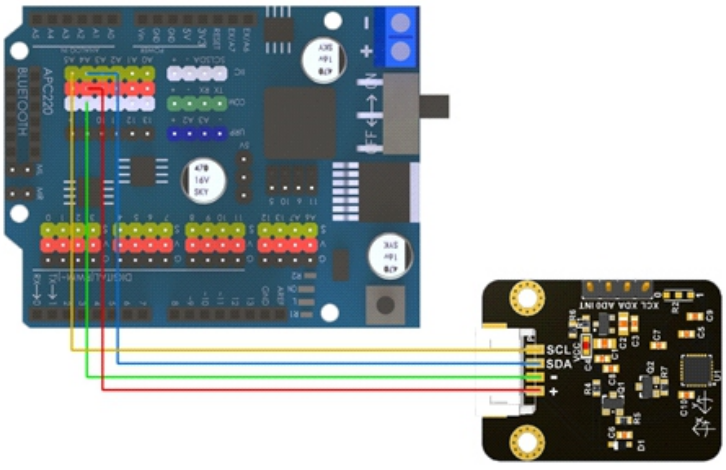
模块介绍：MPU6050是结合3轴陀螺仪，3轴加速度计和数字温度计的系统，他的特殊功能是内置的数字运动处理器（DMP）单元，每个通道都有一个16位模数转换器硬件，同时捕获X、Y、Z通道。对于无人机、机器人、运动传感器的许多应用是非常有用。

输出类型：数字量输出。
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND
XCL	传感器 I2C 时钟线，用于配置和读取外部传感器
XDL	传感器 I2C 数据线，用于配置和读取外部传感器
ADO	I2C 从地址 LSB
INT	用于指示数据准备就绪的中断引脚

Arduino
接线图



安装MPU6050库，库下载地址https://github.com/jarzebski/Arduino-MPU6050

代码（从陀螺仪读取数据）

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup()
{
  Serial.begin(115200);

  Serial.println("Inicjalizacja MPU6050");
  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Nie mozna znalezc MPU6050 - sprawdz polaczenie!");
    delay(500);
  }

  // Kalibracja żyroskopu
  mpu.calibrateGyro();

  // Ustawienie czułości
  mpu.setThreshold(3);
}

void loop()
{
  Vector rawGyro = mpu.readRawGyro();
  Vector normGyro = mpu.readNormalizeGyro();

  Serial.print(" Xraw = ");
  Serial.print(rawGyro.XAxis);
  Serial.print(" Yraw = ");
  Serial.print(rawGyro.YAxis);
  Serial.print(" Zraw = ");
  Serial.println(rawGyro.ZAxis);

  Serial.print(" Xnorm = ");
  Serial.print(normGyro.XAxis);
  Serial.print(" Ynorm = ");
  Serial.print(normGyro.YAxis);
  Serial.print(" Znorm = ");
  Serial.println(normGyro.ZAxis);

  delay(10);
}
```

代码（从加速度）

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup()
{
  Serial.begin(115200);

  Serial.println("Inicjalizacja MPU6050");
  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Nie mozna znalezc MPU6050 - sprawdz polaczenie!");
    delay(500);
  }

}

void loop()
{
  Vector rawGyro = mpu.readRawGyro();
  Vector normGyro = mpu.readNormalizeGyro();

  Serial.print(" Xraw = ");
  Serial.print(rawGyro.XAxis);
  Serial.print(" Yraw = ");
  Serial.print(rawGyro.YAxis);
  Serial.print(" Zraw = ");
  Serial.println(rawGyro.ZAxis);

  Serial.print(" Xnorm = ");
  Serial.print(normGyro.XAxis);
  Serial.print(" Ynorm = ");
  Serial.print(normGyro.YAxis);
  Serial.print(" Znorm = ");
  Serial.println(normGyro.ZAxis);

  delay(10);
}
```

代码

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup()
{
  Serial.begin(115200);

  Serial.println("Inicjalizacja MPU6050");

  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Nie mozna znalezc MPU6050 - sprawdz polaczenie!");
    delay(500);
  }
}

void loop()
{
  float temp = mpu.readTemperature();

  Serial.print(" Temp = ");
  Serial.print(temp);
  Serial.println(" *C");

  delay(500);
}
```

五向导航模块

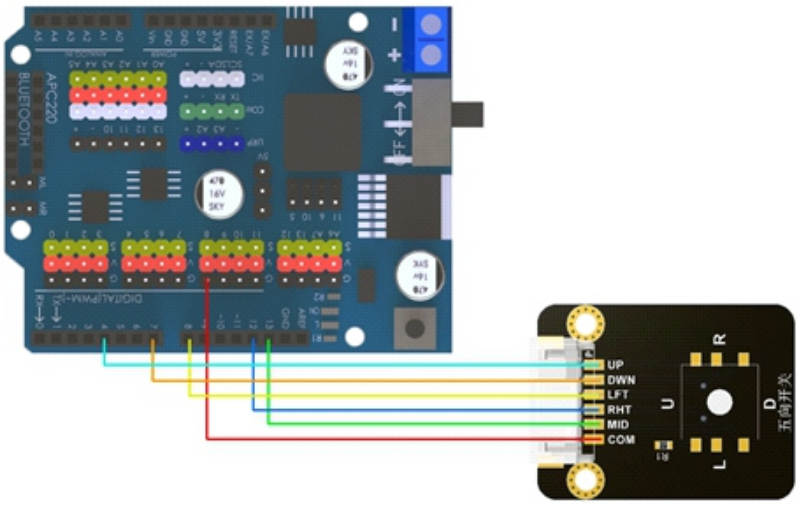
模块介绍：五向开关其最大的不同就是由五个触点。在内底部，有着中央固定触点以及公用触点外壳，还有就是在其周围有很多周边的固定触点的外壳。这些外壳都放置在活动触点簧片上。当机器想要形成一个通路的时候，就可以直接让触点簧片和公用触点连接在一起。所述活动触点簧片能够连接在一起，加上操作杆的控制，就能够使得它一个活动触点簧片可以直接与其相对应的触点连接起来。

输出类型：数字输出。

引脚定义：

Up	上边数字输出
Dow	下边数字输出
Lft	左边数字输出
Rgh	右边数字输出
Mid	中间数字输出
com	公共端(共阴)

Arduino
例子一
接线图



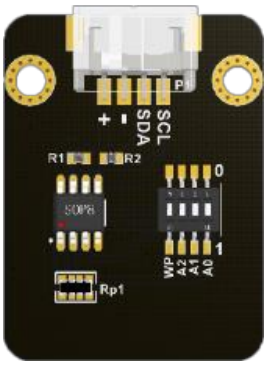
代码

```
#define     KEY_UP          4
#define     KEY_DOWN        7
#define     KEY_LEFT        8
#define     KEY_RIGHT       12
#define     KEY_ENTER       13

int key_release_flag = 1, i;
int time_ticks = 0;

void setup()
{
  pinMode(KEY_UP, INPUT);
  pinMode(KEY_DOWN, INPUT);
  pinMode(KEY_LEFT, INPUT);
  pinMode(KEY_RIGHT, INPUT);

  pinMode(KEY_ENTER, INPUT);
  digitalWrite(KEY_UP, HIGH);
  digitalWrite(KEY_DOWN, HIGH);
  digitalWrite(KEY_LEFT, HIGH);
  digitalWrite(KEY_RIGHT, HIGH);
  digitalWrite(KEY_ENTER, HIGH);

  Serial.begin(9600);
}

void loop()
{
  if((digitalRead(KEY_UP) == LOW) || (digitalRead(KEY_DOWN) == LOW) ||
(digitalRead(KEY_LEFT) == LOW) || (digitalRead(KEY_RIGHT) == LOW) ||
(digitalRead(KEY_ENTER) == LOW))
  {
    if(key_release_flag)
    {
      delay(10);

      if((digitalRead(KEY_UP) == LOW) || (digitalRead(KEY_DOWN) == LOW) ||
(digitalRead(KEY_LEFT) == LOW) || (digitalRead(KEY_RIGHT) == LOW) ||
(digitalRead(KEY_ENTER) == LOW))
      {
        key_release_flag = 0;
        if(digitalRead(KEY_UP) == LOW)
          Serial.print("you press UP\n");
        else if(digitalRead(KEY_DOWN) == LOW)
          Serial.print("you press DOWN\n");
        else if(digitalRead(KEY_LEFT) == LOW)
          Serial.print("you press LEFT\n");
```

```
            else if(digitalRead(KEY_RIGHT) == LOW)
               Serial.print("you press RIGHT\n");
            else if(digitalRead(KEY_ENTER) == LOW)
               Serial.print("you press ENTER\n");
         }
      }
   }
   else
      key_release_flag = 1;
}
```

结果：打开串口窗口，尝试分别向上下左右中按下，串口会出现你按下模块的哪一个方向的按键。

EEPRROM模块



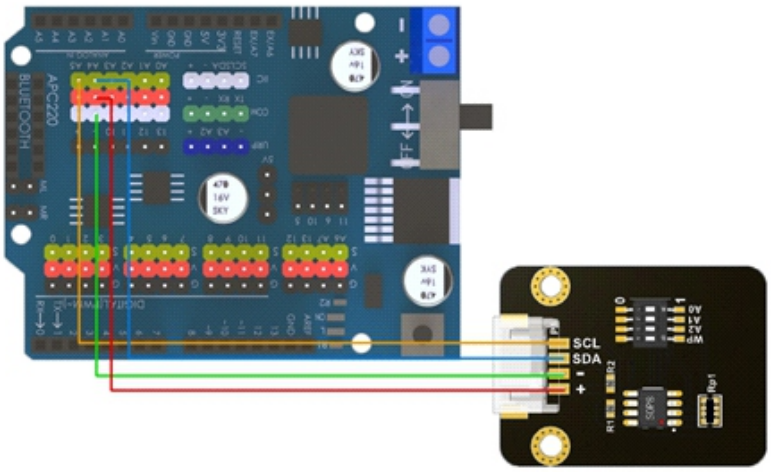
模块介绍：EEPRROM是带电可擦可编程只读存储器，是一种掉电后数据不丢失的存储芯片。

输出类型：IIC协议。
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND

Arduino
接线图



模块上有一个码盘，拨动码盘，确定设备地址。

样例一：写入数据

```
#include <EEPROM.h>

/** the current address in the EEPROM (i.e. which byte we're going to write to next) **/
int addr = 0x55;

void setup() {
}

void loop() {

   int val = analogRead(0) / 4;

   EEPROM.write(addr, 50);

   addr = addr + 1;
   if (addr == EEPROM.length()) {
      addr = 0;
   }

   delay(100);
}
```

结果：写入数据50，直到无无存储空间才停止写入。

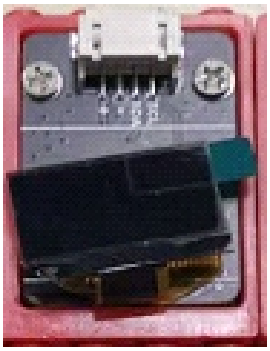
样例二：读取数据

```
/*
 * EEPROM Read
 *
 * Reads the value of each byte of the EEPROM and prints it
 * to the computer.
 * This example code is in the public domain.
 */

#include <EEPROM.h>

// start reading from the first byte (address 0) of the EEPROM
int address = 0x55;
byte value;

void setup() {
   // initialize serial and wait for port to open:
   Serial.begin(9600);
}
```

```
void loop() {
    // read a byte from the current address of the EEPROM
    value = EEPROM.read(address);

    Serial.print(address);
    Serial.print(" t");
    Serial.print(value, DEC);
    Serial.println();
    address = address + 1;
    if (address == EEPROM.length()) {
        address = 0;
    }

    delay(10);
}
```

结果：串口窗口显示内存所有的数据。

OLED屏模块



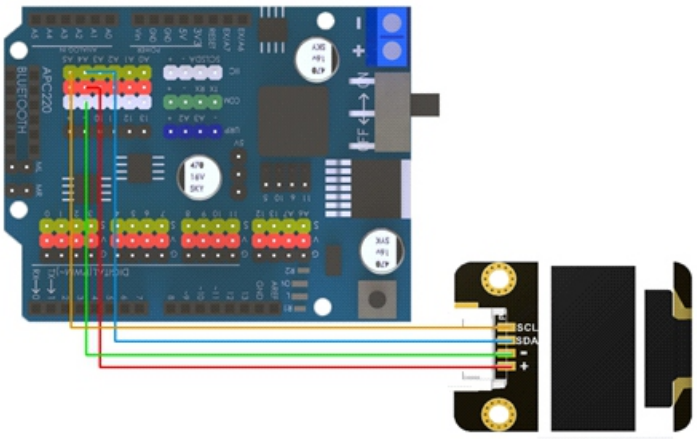
模块介绍：OLED（Organic Light-Emitting Diode）又称为有机电激光显示、有机发光半导体（Organic Electroluminesence Display，OLED）。OLED属于一种电流型的有机发光器件，是通过载流子的注入和复合而致发光的现象，发光强度与注入的电流成正比。OLED在电场的作用下，阳极产生的空穴和阴极产生的电子就会发生移动，分别向空穴传输层和电子传输层注入，迁移到发光层。当二者在发光层相遇时，产生能量激子，从而激发发光分子最终产生可见光。一般而言，OLED可按发光材料分为两种：小分子OLED和高分子OLED（也可称为PLED）。OLED是一种利用多层有机薄膜结构产生电致发光的器件，它很容易制作，而且只需要低的驱动电压，这些主要的特征使得OLED在满足平面显示器的应用上显得非常突出。OLED显示屏比LCD更轻薄、亮度高、功耗低、响应快、清晰度高、柔性好、发光效率高，能满足消费者对显示技术的新需求。

输出类型：IIC协议。
接口模式：pH2.0-4P

引脚定义：

SCL	时钟线
SDA	双向数据线
+	VCC
-	GND

Arduino
接线图



安装u8glib.h

样例一

```
#include "U8glib.h"    //加载显示库文件
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);
// I2C / TWI  实例化

void setup() {
}

void loop() {
   u8g.firstPage();    //一下是显示实现部分
   do {
        u8g.setFont(u8g_font_fub14);//设置字体和自号
        u8g.setPrintPos(0, 50); //显示的位置
        u8g.print("Hello World!");
   }
   while( u8g.nextPage() );
}
```

结果：屏幕显示"Hello World!"的文字。

样例二

```
#include "U8glib.h"    //加载显示库文件
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);
// I2C / TWI  实例化
void setup() {
}
void loop() {
   for(int i=1;i<101;i++){
        u8g.firstPage();
   do {
      //显示实现部分
        u8g.setFont(u8g_font_fub14);
        //设置字体和自号，目前测试字号有 fub14,17,20,30
        u8g.setPrintPos(50, 50); //显示的位置
        u8g.print(i);//显示变量 i 的值
        u8g.setFont(u8g_font_fub14);//设置字体和自号
        u8g.setPrintPos(95, 50); //显示的位置
        u8g.print("cm");//显示 cm 字样
   }
   while( u8g.nextPage() );
   delay(100);//显示的时间间隔。
 }
}
```

结果：屏幕循环显示1-100cm的数字及英文字母。

U8glib库函数说明：
u8g.firstPage() 表示图像循环的开始
u8g.nextPage() 表示图像循环的结束
setRot90();　　//旋转屏幕90°
setRot180();　//旋转屏幕180°
setRot270();　//旋转屏幕270°

画几何图形
画点：void U8GLIB::drawPixel(unit8_t x, unit8_t y)
参数为：（x:点的横坐标 y:点的纵坐标）

画线：void U8GLIB::drawLine(u8g_uint_t x1, u8g_uint_t y1, u8g_uint_t x2, u8g_uint_t y2)
参数为：（x1:线段起始点横坐标 y1:线段起始点纵坐标 x2:线段结束点横坐标 y2:线段结束点纵坐标）

画线段（水平线段、垂直线段）：
水平线段void U8GLIB::drawHLine(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w)
垂直线段void U8GLIB::drawVLine(u8g_uint_t x, u8g_uint_t y, u8g_uint_t h)
参数为：（x:线段起始点 y:线段结束点 w:水平宽度 h:垂直高度）
(高度单位为像素点)

画实心三角形：void U8GLIB::drawTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)
参数为：（x0:一个角的横坐标 y0:一个角的纵坐标 x1:另一个角的横坐标 y1:另一个角的纵坐标 x2:最后一个角的横坐标 y2:最后一个角的纵坐标）

画空心矩形：void U8GLIB::drawFrame(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)
参数是：（x:矩形左上角横坐标 y:矩形左上角纵坐标 w:矩形的宽 h:矩形的高）
(高和宽的单位都是像素)

画实心圆：void U8GLIB::drawCircle(u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rad, uint8_t opt = U8G_DRAW_ALL)

//参数是：（x0:圆心的横坐标 y0:圆心的纵坐标 rad:圆的半径 opt:见下表）

U8G_DRAW_UPPER_RIGHT	上部右侧 1/4 圆弧
U8G_DRAW_UPPER_LEFT	上部左侧 1/4 圆弧
U8G_DRAW_LOWER_LEFT	下部左侧 1/4 圆弧
U8G_DRAW_LOWER_RIGHT	下部右侧 1/4 圆弧
U8G_DRAW_ALL	整圆（默认）

画实心矩形：void U8GLIB::drawBox(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)

参数是：（x:矩形左上角横坐标 y:矩形左上角纵坐标 w:矩形宽度h:圆角矩形高度

画空心圆角矩形：void U8GLIB::drawRFrame(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h, u8g_uint_t r)

//参数是：（x:圆角矩形左上角横坐标 y:圆角矩形左上角纵坐标 w:圆角矩形宽度 h:圆角矩形高度 r:圆角弧度的半径）

注意：这里的r因矩形的宽度和高度的大小而有范围变化。w>2*r且h>2*r。

画实心圆角矩形：void U8GLIB::drawRBox(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h, u8g_uint_t r)

画空心椭圆：void drawEllipse(u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry)

//参数是：（x0:椭圆圆心的横坐标 y0:椭圆圆心的纵坐标 rx:水平方向半径 ry:垂直方向半径

画实心椭圆：void drawFilledEllipse(u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry, uint8_t opt)

//参数是：（x0:椭圆圆心的横坐标 y0:椭圆圆心的纵坐标 rx:水平方向半径 ry:垂直方向半径 rad:圆的半径 opt:见下表）

U8G_DRAW_UPPER_RIGHT	上部右侧 1/4 椭圆弧
U8G_DRAW_UPPER_LEFT	上部左侧 1/4 椭圆弧
U8G_DRAW_LOWER_LEFT	下部左侧 1/4 椭圆弧
U8G_DRAW_LOWER_RIGHT	下部右侧 1/4 椭圆弧
U8G_DRAW_ALL	整椭圆（默认）

画字符、字符串

方法一：u8g_uint_t U8GLIB::drawStr(u8g_uint_t x, u8g_uint_t y, const char *s)
参数为：（x:字符左下角的横坐标 y:字符左下角的纵坐标 s:要画出的字符）
注意：使用drawStr函数之前，需要使用setFont函数来设置一下要画出的字符的显示字体。setFont();设置字体和字号，目前测试字号有u8g_font_fub14, u8g_font_fub17, u8g_font_fub20, u8g_font_fub30

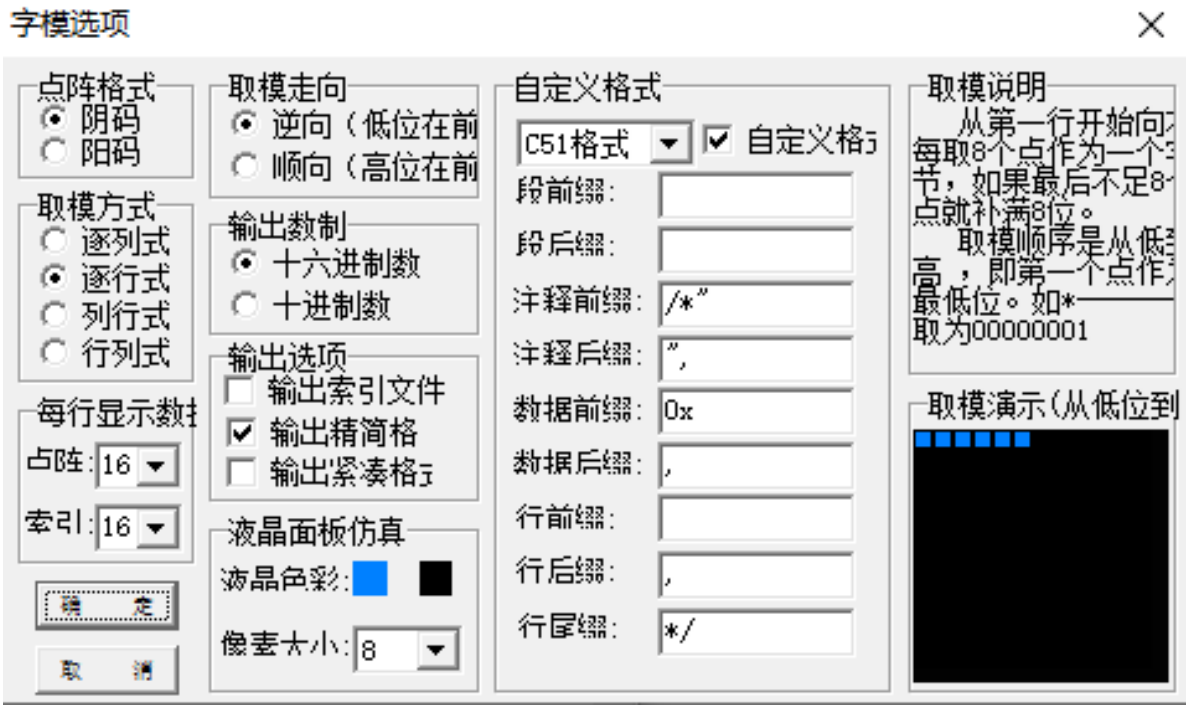
方法二：
U8GLIB::print(...) 参数为要打印的内容
注意：使用print函数之前，需要用 setPrintPos(u8g_uint_t x, u8g_uint_t y) 来设置位置，参数为（x:字符左下角的横坐标 y:字符左下角的纵坐标）

同时drawStr函数还有三种变形：

```
drawStr90();     //字符顺时针旋转响应90°
drawStr180();    //字符顺时针旋转响应180°
drawStr270();    //字符顺时针旋转响应270°
```

画出图像
方法一：void U8GLIB::drawXBMP(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h, const u8g_pgm_uint8_t *bitmap)
//参数为：（x:位图左上角的横坐标 y:位图左上角的纵坐标 w:位图的宽 h:位图的高 *bitmap:位图对象）

打开“PCtoLCD2002“图像取模软件，设置选项。



文件->打开->bmp后缀的图像文件，点击“生成字模“，复制生成的字模数据，将数据存放在位图对象中。

样例三

```
#include "U8glib.h"    //加载显示库文件
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);
// I2C / TWI  实例化

static unsigned char test[] U8G_PROGMEM = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xD8,0x85,0x43,0x40,0xE1,0x80,0x5E,0x30,
0xB8,0x17,0x10,0x00,0x08,0x84,0x44,0x40,0x91,0x80,0x80,0x48,0x04,0x14,0x10,0x00,
0xF8,0x47,0x4C,0x40,0x19,0x81,0x80,0x84,0x04,0x10,0x10,0x00,0x08,0x6C,0x48,0x40,
0x09,0x83,0x80,0x84,0x84,0x15,0x10,0x00,0x08,0xEC,0x5F,0x41,0xFD,0x83,0xC0,0xFE,
0x0D,0x14,0x10,0x00,0xF8,0x17,0x50,0x3E,0x05,0x84,0x7F,0x01,0xFA,0xE3,0x0F,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xF8,0xFF,0x3F,0x7C,0xF8,0x00,0xF0,0x01,0xE0,0xE3,0x03,0x00,0x00,0x7C,0x00,0x7C,
0xFF,0x03,0xF0,0x01,0xF0,0xC1,0x07,0x00,0x00,0x7E,0x00,0x3C,0xFA,0x02,0xF0,0x01,
0xF8,0x80,0x0F,0x00,0xF0,0x01,0x1F,0x3C,0xF8,0xE0,0xFF,0xFF,0x00,0x7F,0x00,0x00,
0xF0,0x01,0x3E,0x7E,0xF8,0x00,0xF0,0x01,0x80,0xFF,0x00,0x00,0xF0,0x01,0x1E,0x7E,
0xFF,0x07,0xF0,0x01,0xE0,0xF7,0x03,0x00,0xF0,0x01,0x1E,0x7C,0x78,0x00,0xF8,0x03,
0xF8,0xC1,0x07,0x00,0xF0,0xFF,0x1F,0x7C,0xF8,0x00,0xFC,0x07,0xF8,0xFF,0x0F,0x00,
0xF0,0x01,0x1E,0x7C,0xFF,0x07,0xFE,0x0F,0xF0,0xC1,0x07,0x00,0xF0,0x01,0x1E,0x7C,
0xF8,0x00,0x3F,0x1F,0xF0,0x81,0x07,0x00,0xF0,0x01,0x3E,0x7C,0xF8,0x80,0x1F,0x3E,
0xF0,0xC1,0x07,0x00,0xF0,0xFF,0x1F,0x7C,0xFF,0xC7,0x0F,0x7C,0xF0,0xFF,0x07,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00
};

void draw(){
   u8g.drawXBMP( 0, 0, 90, 47, test);
```

```
}
void
setup() {
   // put your setup code here, to run once:
   //旋转屏幕 180°
   u8g.setRot180();// rotate screen
}

void
loop() {
   // put your main code here, to run repeatedly:
   u8g.firstPage();
   do{
       draw();
   }while(u8g.nextPage());
}
```

结果：屏幕显示“百佳大谷”的图片logo。

方法二：void U8GLIB::drawBitmapP(u8g_uint_t x, u8g_uint_t y, u8g_uint_t cnt, u8g_uint_t h, const u8g_pgm_uint8_t *bitmap)

//参数为：（x:位图左上角的横坐标 y:位图左上角的纵坐标 cnt:在水平方向上位图的字节数 h:位图的高 *bitmap:位图对象），所以位图的宽，就是(cnt * 8)，1字节=8位。

样例四

```
const
uint8_t rook_bitmap[] U8G_PROGMEM = {
   0x00,            // 00000000
   0x55,            // 01010101
   0x7f,            // 01111111
```

```
    0x3e,              // 00111110
    0x3e,              // 00111110
    0x3e,              // 00111110
    0x3e,              // 00111110
    0x7f               // 01111111
};
void
draw(){
      u8g.drawBitmapP(0,0, 1, 8, rook_bitmap);
}
void
setup() {
   // put your setup code here, to run once:
   //旋转屏幕 180°
   u8g.setRot180();// rotate screen
}

void
loop() {
   // put your main code here, to run repeatedly:
   u8g.firstPage();
   do{
       draw();
   }while(u8g.nextPage());
}
```

结果：屏幕显示一个  的图案。