

《自动化认知与实践》之 Arduino 传感器编程入门



哈尔滨工业大学（深圳）

机电学院自动化 陈浩耀

邮箱: hychen5@hit.edu.cn

地址: G栋310

□ 移动小车关键部分：

人机交互系统

- 通讯模块
- 语音识别模块

控制逻辑系统

- 单片机
- 控制器设计
- 传感器信息处理

传感器系统

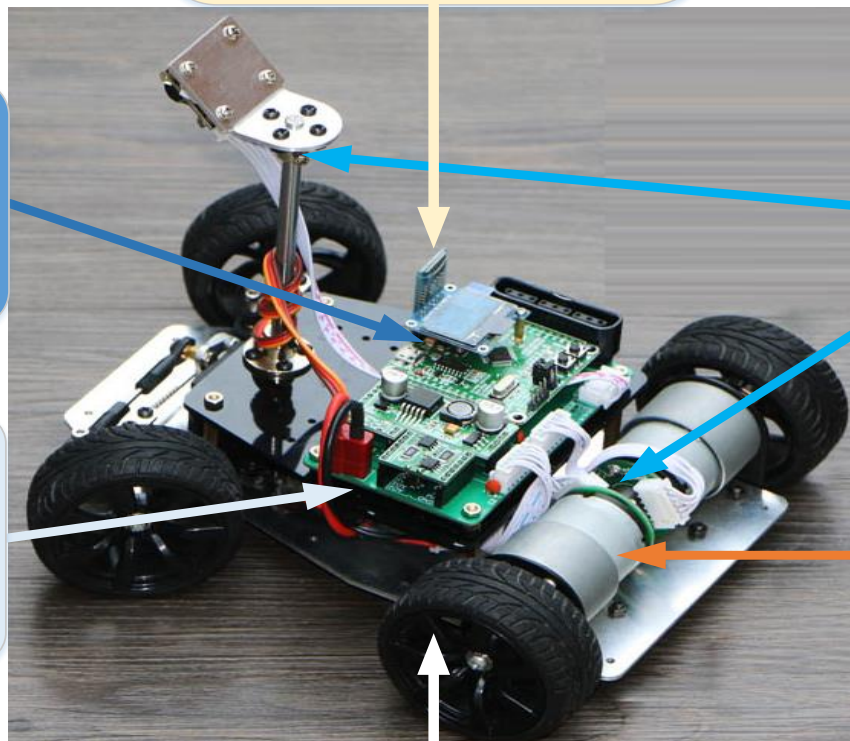
- 码盘
- 寻线传感器
- 避障

电路系统

- 电源电路
- 电机功率驱动
- 模数转换

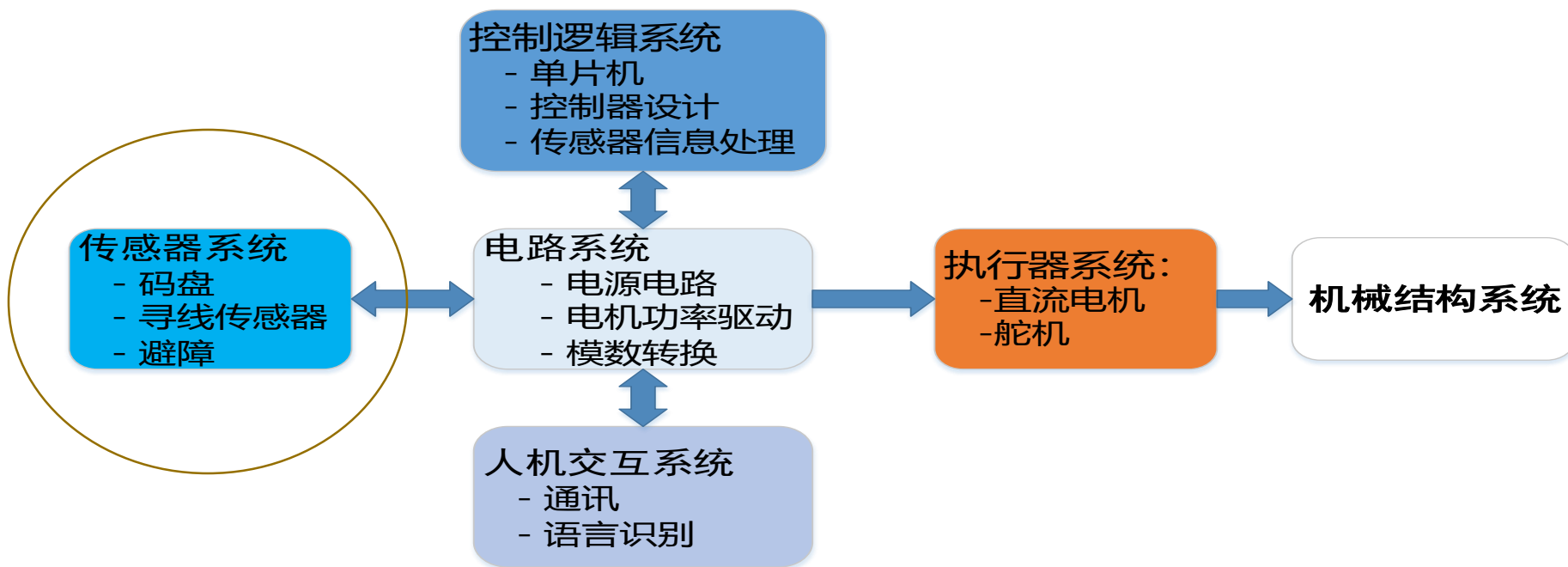
执行器系统：

- 直流电机
- 舵机



机械结构系统

□ 机器人（自动化）系统的基本组成：



□ Strong Recommendation:

自己要亲自编写代码！！！！




Arduino基本函数的使用





Arduino时间函数



□ 时间函数

- 1、`delay(ms);`
- 2、`delayMicroseconds(us);`
- 3、`millis();`
- 4、`micros();`

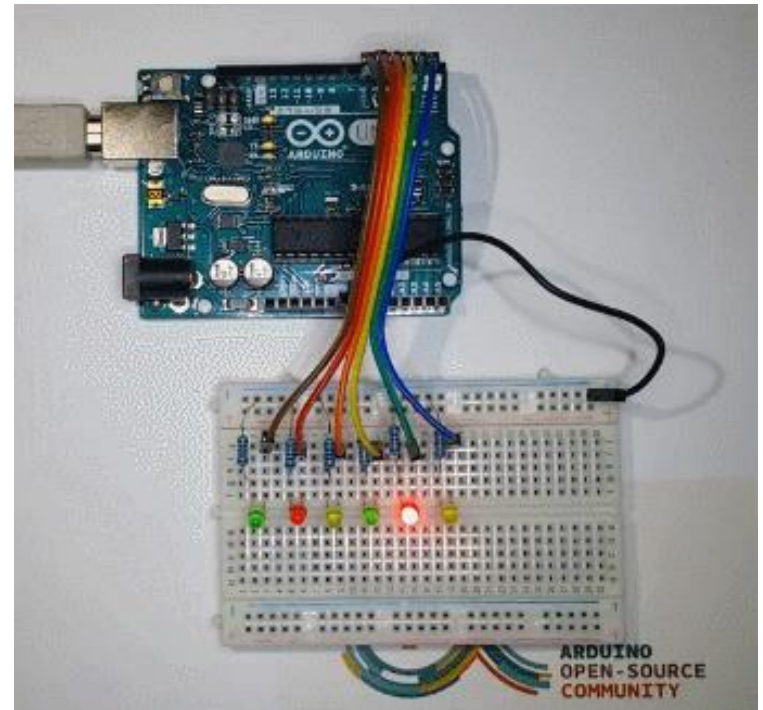
1、delay(ms);

- 延时函数，参数是延时的时长，单位是ms(毫秒)。

是一种阻塞（blocking）函数

例程-跑马灯

```
void setup(){
    pinMode(0,OUTPUT); //定义为输出
    pinMode(1,OUTPUT);
    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
}
void loop(){
    int i;
    for(i = 0; i <= 5; i++) //依次循环六盏灯
    {
        digitalWrite(i,HIGH); //点亮LED
        delay(1000); //持续1秒
        digitalWrite(i,LOW); //熄灭LED
        delay(1000); //持续1秒
    }
}
```



2、delayMicroseconds(us);

- 延时函数，参数是延时的时长，单位是us(微秒)。1ms=1000us.
- 该函数可以产生更短的延时。

3、millis()

- **timer0 的中断帮忙每次加 1（室友帮忙计数）**
- 应用该函数，可以获取单片机通电到现在运行的时间长度，单位是ms。系统最长的记录时间为**约为50天**，超出从0开始。返回值是unsigned long型。
- 该函数适合作为定时器使用，不影响单片机的其他工作。（使用**delay**函数期间无法做其他工作。）
- 1000毫秒=1秒

例程-延时10秒后自动点亮的灯

```
int LED=13;
unsigned long i, j;
void setup()
{
    pinMode(LED,OUTPUT);
    i=millis(); //读入初始值
}
void loop()
{
    j=millis(); //不断读入当前时间值
    if((j-i)>10000) //如果延时超过10秒， 点亮LED
    {
        digitalWrite(LED,HIGH);
    }
    else
        digitalWrite(LED,LOW);
}
```

4、micros()

- 该函数返回开机到现在运行的微秒值。返回值是unsigned long。**70分钟溢出。**
- 1000微秒=1毫秒


例程原理

- 连续按按钮，看你的反应有多快。
- 按钮接某个数字口；每按一次会减去上一次按的时间，看你连续按的间隔时间有多快（使用**OLED**或者串口监视器显示）。(选做课后作业，设计这个程序，然后在实验课程验证)


5、MsTimer2定时器库

- 利用单片机内部的硬件定时器实现周期性定时

```
01. //定时器库的头文件，除了使用MsTimer2，也可以使用
    millis(),但很容易受到干扰，定时不准
02. #include <MsTimer2.h>
03. void control() {
04. //这里面可以放置编码器测速算法和电机转速PID 控制算法
05. }
06. void setup() {
07. //设置每隔PERIOD 时间，执行control 函数一次
08. MsTimer2::set(PERIOD, control);
09. MsTimer2::start();
10. }
11. void loop() {
12. }
```

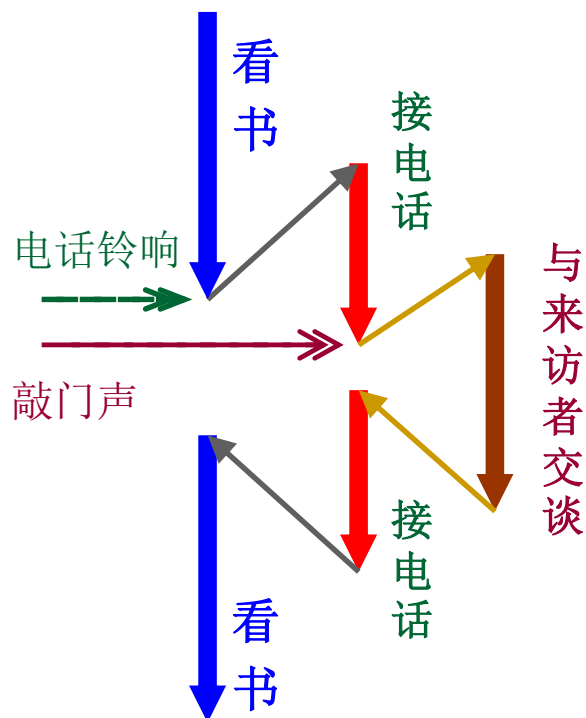


Arduino中断函数



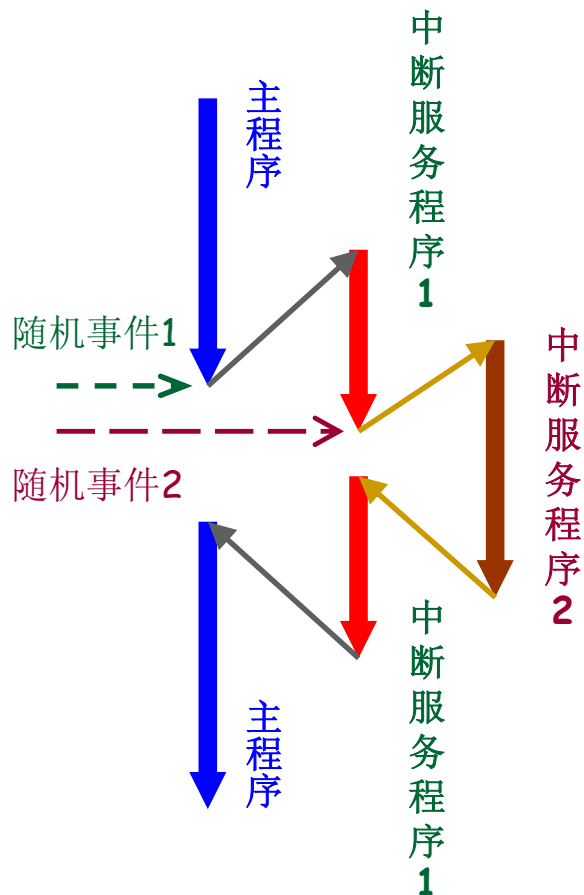
1、中断的概念

日常生活中的中断



- 你在看书，电话铃响，于是你在书上做上记号，去接电话，与对方通话；
- 门铃响了，有人敲门，你让打电话的对方稍等一下，你去开门，并在门旁与来访者交谈，谈话结束，关好门；
- 回到电话机旁，继续通话，接完电话后再回来从做记号的地方接着看书。

单片机中的中断概念



- **中断**——由于某一随机事件的发生，计算机**暂停**原程序的运行，**转去**执行另一程序（随机事件），处理完毕后又自动**返回**原程序继续运行。
- **中断源**——引起中断的原因，或能发生中断申请的来源。
- **主程序**——计算机现行运行的程序。
- **中断服务子程序**——处理突发事件的程序。

实现分时操作：

提高 **CPU** 的效率 只有当服务对象向 **CPU** 发出中断申请时 才去为它服务 这样 我们就可以利用中断功能同时为多个对象服务 从而大大提高了 **CPU** 的工作效率。

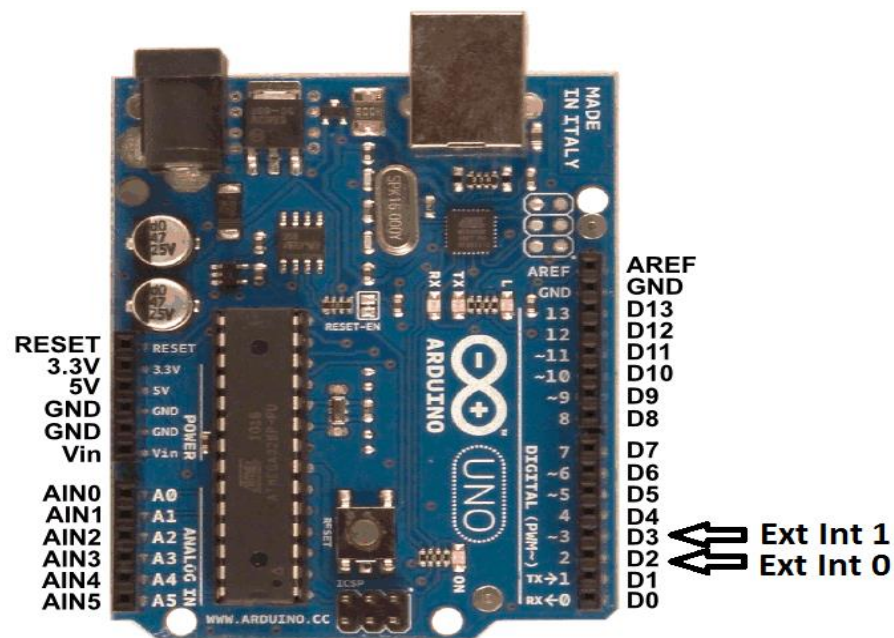
2、 中断函数

- 1、 `attachInterrupt(interrput,function,mode);`
 - 2、 `detachInterrupt(interrput);`
-

函数说明

`attachInterrupt(interrupt,function,mode);`

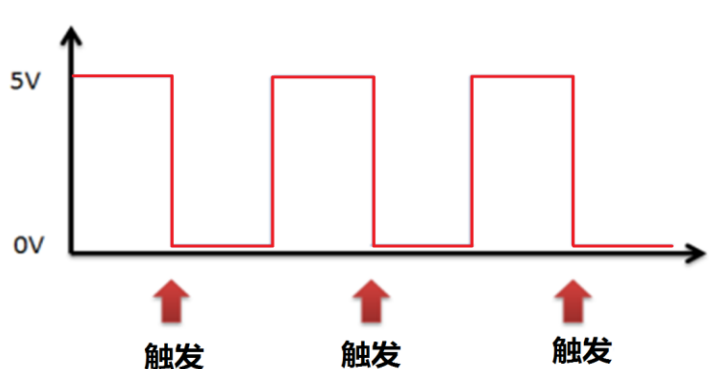
- 用于设置外部中断，函数有3个参数，分别表示中断源，中断处理函数和触发模式。
- 中断源可选0或者1，对应2或者3号数字引脚。
- 中断处理函数是一段子程序，当中断发生时执行该子程序部分。



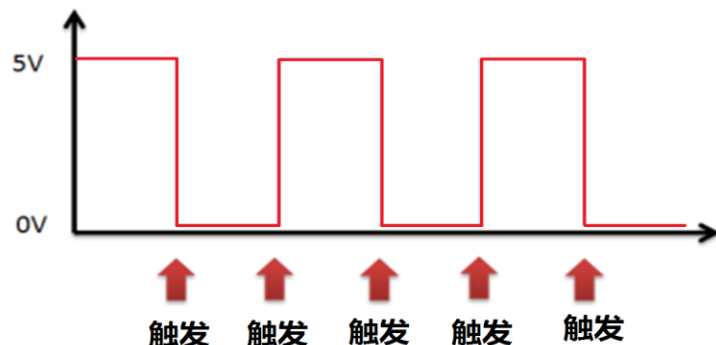
2、函数说明

`attachInterrupt(interrput,function,mode);`

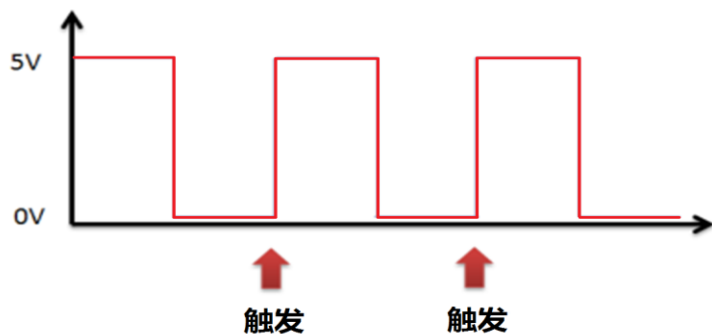
- 触发模式有四种类型，**LOW**(低电平触发)、**CHANGE**(变化时触发)、**RISING**（低电平变为高电平触发）、**FALLING**(高电平变为低电平触发)



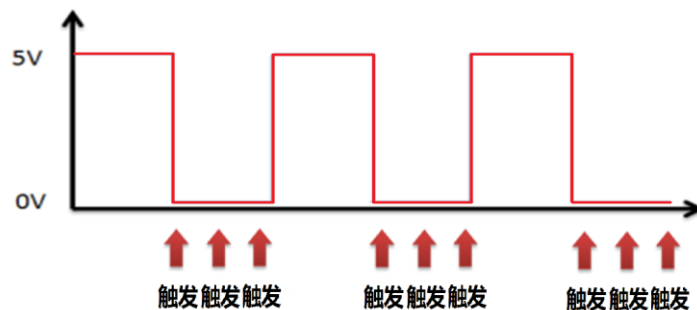
FALLING



CHANGE

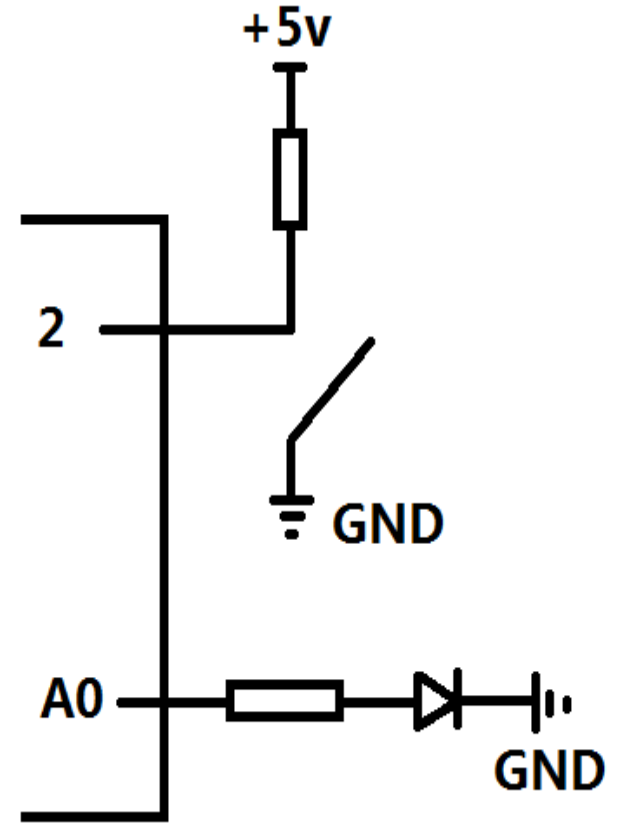
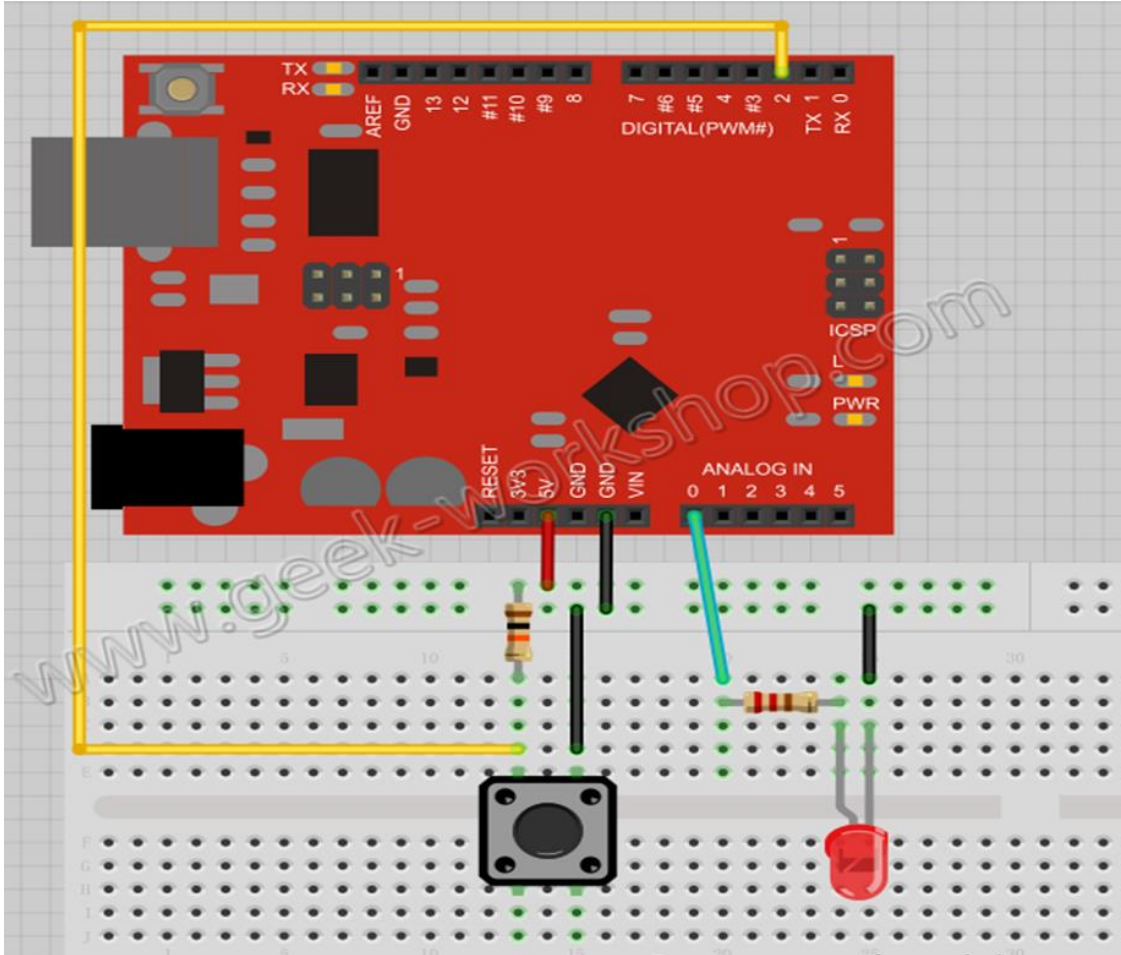


RISING



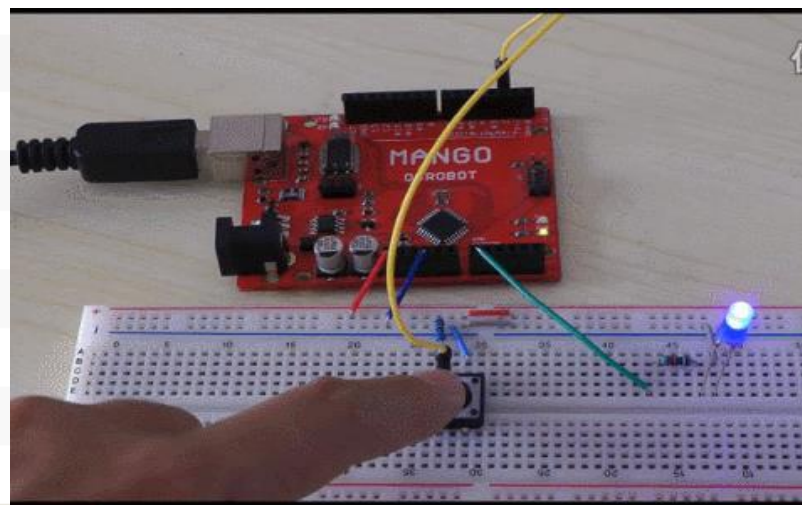
LOW

例子



```
01. int pbIn = 2; // 定义输入信号引脚
02. int ledOut = A0; // 定义输出指示灯引脚
03. int state = LOW; // 定义默认输入状态
04.
05. void setup()
06. {
07.     // 设置输入信号引脚为输入状态、输出引脚为输出状态
08.     pinMode(pbIn, INPUT);
09.     pinMode(ledOut, OUTPUT);
10. }
11.
12. void loop()
13. {
14.     state = digitalRead(pbIn); // 读取微动开关状态
15.     digitalWrite(ledOut, state); // 把读取的状态赋予LED指示灯
16.
17.     // 模拟一个长的流程或者复杂的任务
18.     for (int i = 0; i < 100; i++)
19.     {
20.         // 延时10毫秒
21.         delay(10);
22.     }
23. }
```

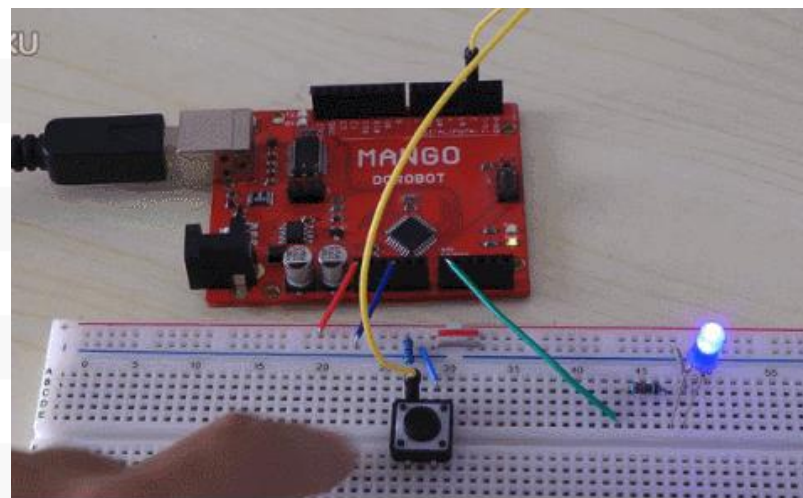
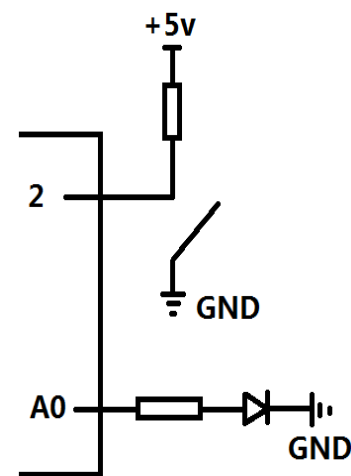
该代码的实验效果，按下按钮，**LED**状态不会立刻改变，要按住一会儿才能改变。




```

01.  int pbIn = 0;                                // 定义中断引脚为
    0, 也就是D2引脚
02.  int ledOut = A0;                            // 定义输出指示灯引
    脚
03.  volatile int state = LOW; // 定义默认输入状态
04.  void setup(){
05.      // 置ledOut引脚为输出状态
06.      pinMode(ledOut, OUTPUT); // 监视中断输入引脚的变
    化
07.      attachInterrupt(pbIn, stateChange, CHANGE);
08.  }
09.  void loop(){
10.      // 模拟长时间运行的进程或复杂的任务。
11.      for (int i = 0; i < 100; i++)
12.      {
13.          delay(10);
14.      }
15.  }
16.  void stateChange(){
17.      state = !state;
18.      digitalWrite(ledOut, state);
19.  }

```



2、detachInterrupt(interrupt);

- 该函数用于取消中断，参数interrupt表示所要取消的中断源。

3、注意事项

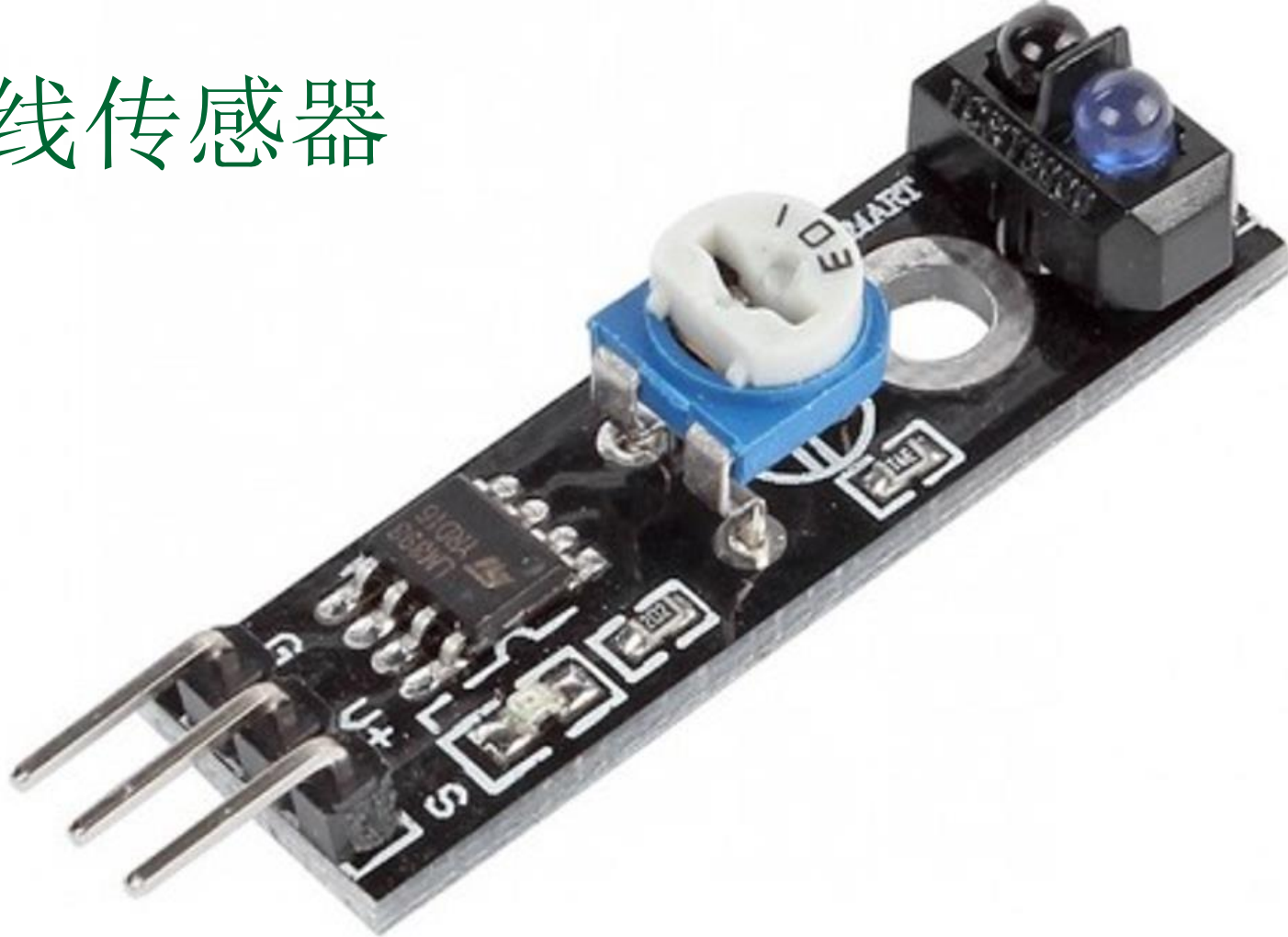
- 当中断函数发生时，**delay()**和**millis()**的数值将不会继续变化。当中断发生时，串口收到的数据可能会丢失。

□ 数学库

- 1、`min(x,y)`; 求两者最小值
- 2、`max(x,y)`; 求两者最大值
- 3、`abs(x)`; 求绝对值
- 4、`sin(rad)`; 求正弦值
- 5、`cos(rad)`; 求余弦值
- 6、`tan(rad)`; 求正切值

机器人小车常用传感器编程

1、寻线传感器

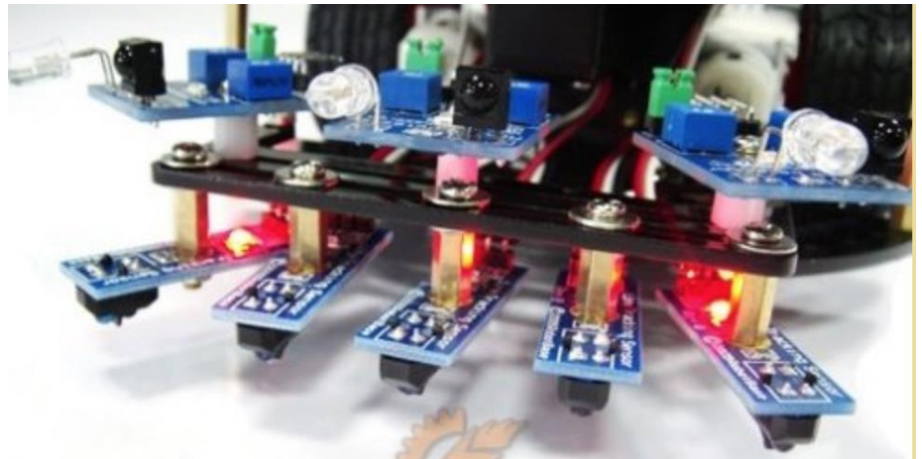


- 遇到白色反射红外，遇到黑色被吸收红外。以此来寻找地面的黑线。

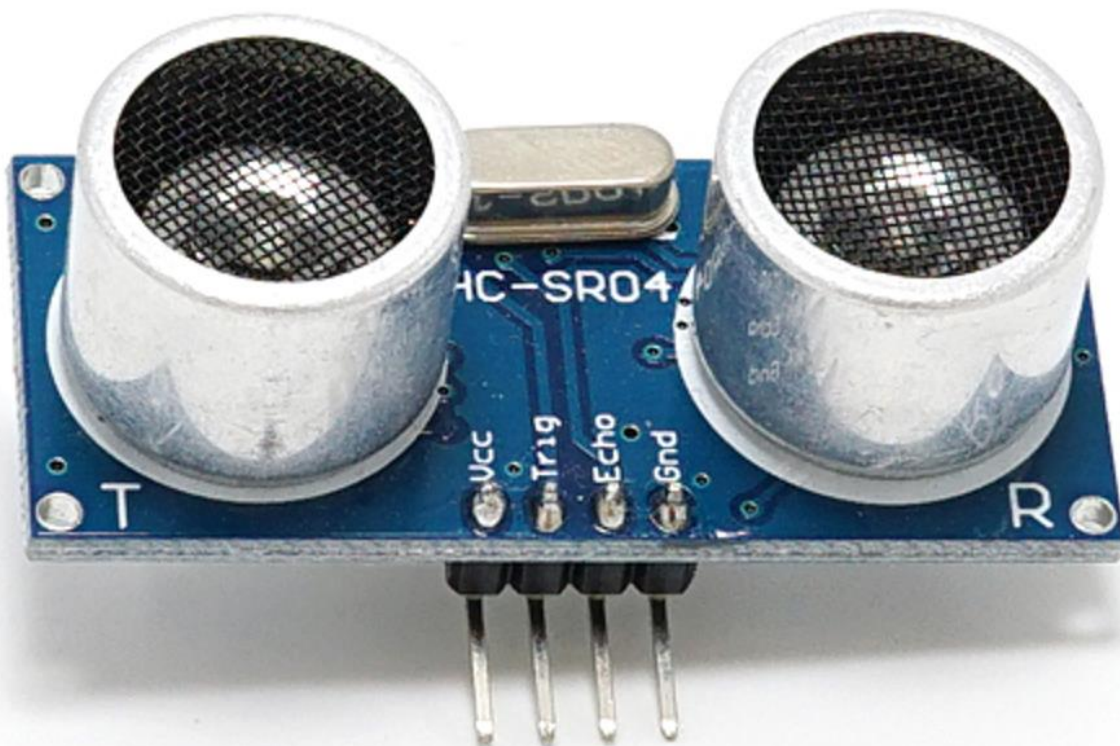
编程原理

- 寻线模块和数字13 接口自带LED 搭建简单电路，制作寻线提示灯
- 利用数字13 接口自带的LED，将寻线传感器接入数字3接口，当寻线传感器感测到有反射信号时（白色），LED 亮,反之（黑线）则灭.

```
■ int Led=13;//定义LED 接口
■ int buttonpin=3; //定义寻线传感器接口
■ int val;//定义数字变量val
■ void setup()
■ {
■     pinMode(Led,OUTPUT);//定义LED 为输出接口
■     pinMode(buttonpin,INPUT);//定义寻线传感器为输出接口
■ }
■ void loop()
■ {
■     val=digitalRead(buttonpin);//将数字接口3的值读取赋给val
■     if(val==HIGH)//当寻线传感器检测有反射信号时，LED 亮
■     {
■         digitalWrite(Led,HIGH);
■     }
■     else
■     {
■         digitalWrite(Led,LOW);
■     }
■ }
```



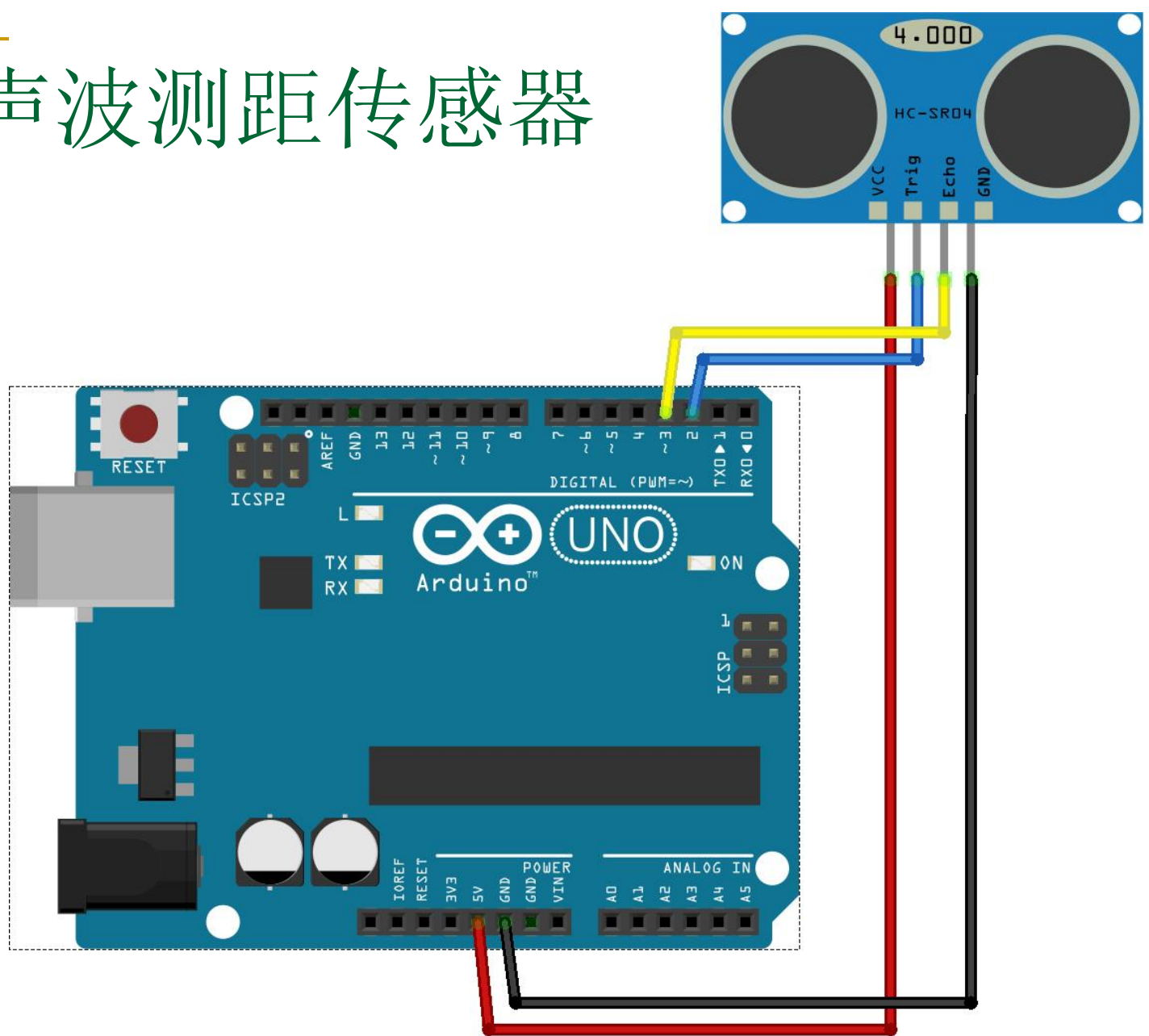
2、超声波测距传感器



Vcc -- 供5V电源
Trig -- 触发控制信号输入
Echo-- 回响信号输出
Gnd -- 为地线

- 用Trig触发测距，会发出8个 40khz的方波，自动检测是否有信号返回有信号返回，通过echo输出高电平，高电平持续的时间就是距离的2倍
- 测量距离 = （高电平时间*声速） / 2

2、超声波测距传感器



```
#define Trig 2 //引脚Trig 连接 IO D2
#define Echo 3 //引脚Echo 连接 IO D3
float cm; //距离变量
float temp; //
void setup() {
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
}
```

```
void loop() {
  //给Trig发送一个短时间脉冲,触发测距
  digitalWrite(Trig, LOW); //给Trig发送一个
  //低电平
  delayMicroseconds(2); //等待 2微妙
  digitalWrite(Trig,HIGH); //给Trig设置高电
  //平
  delayMicroseconds(10); //等待 10微妙
  digitalWrite(Trig, LOW); //给Trig设置低电
  //平
}
```

```
temp = float(pulseIn(Echo, HIGH));
//存储回波等待时间,
//pulseIn函数会等待引脚变为HIGH,开始计算时
//间,再等待变为LOW并停止计时
//返回脉冲的长度
```

```
//声速是:340m/1s 换算成 34000cm /
//1000000μs => 34 / 1000
//因为发送到接收,实际是相同距离走了2回,所以
//要除以2
//距离(厘米) = (回波时间 * (34 / 1000)) / 2
//简化后的计算公式为 (回波时间 * 17) / 1000

cm = (temp * 17 ) / 1000; //把回波时间换算成cm

delay(100);
}
```

3、旋转编码器

将机械转动的模拟量（位移）转换成以数字代码形式表示的电信号。

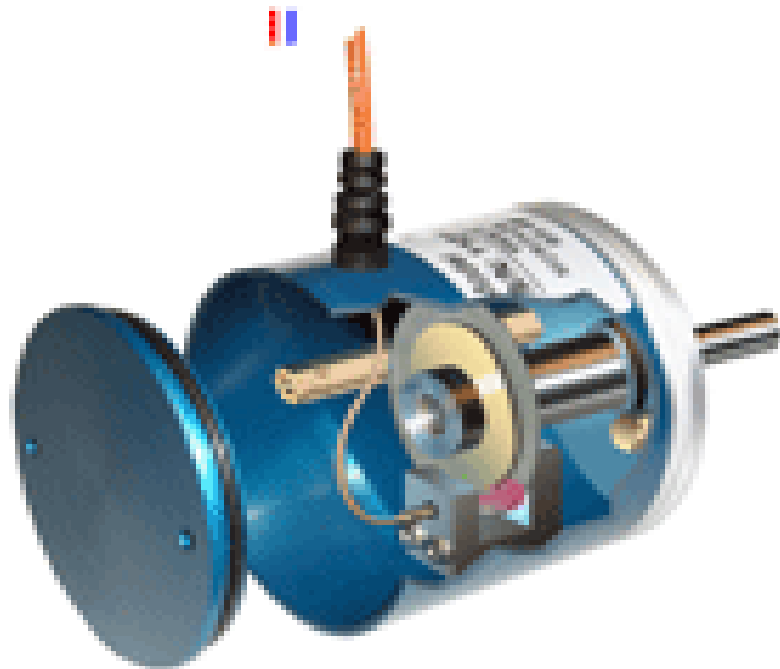
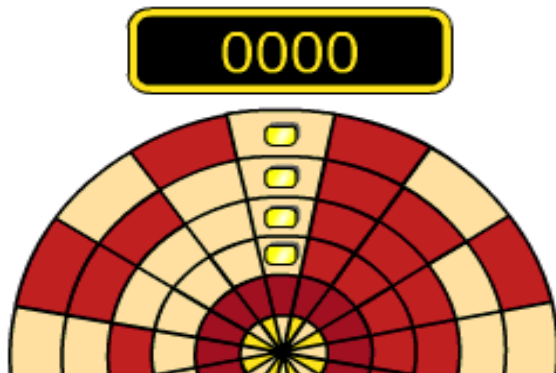


角编码器

角编码器是一种旋转式位置传感器，它的转轴通常随被测轴一起旋转，能将检测轴的角位移转换成二进制编码或一串脉冲

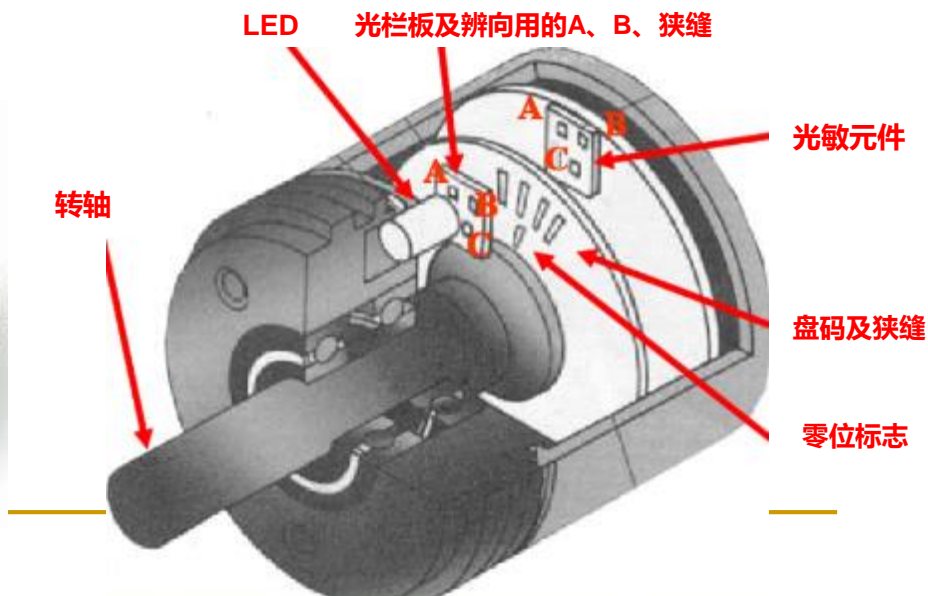
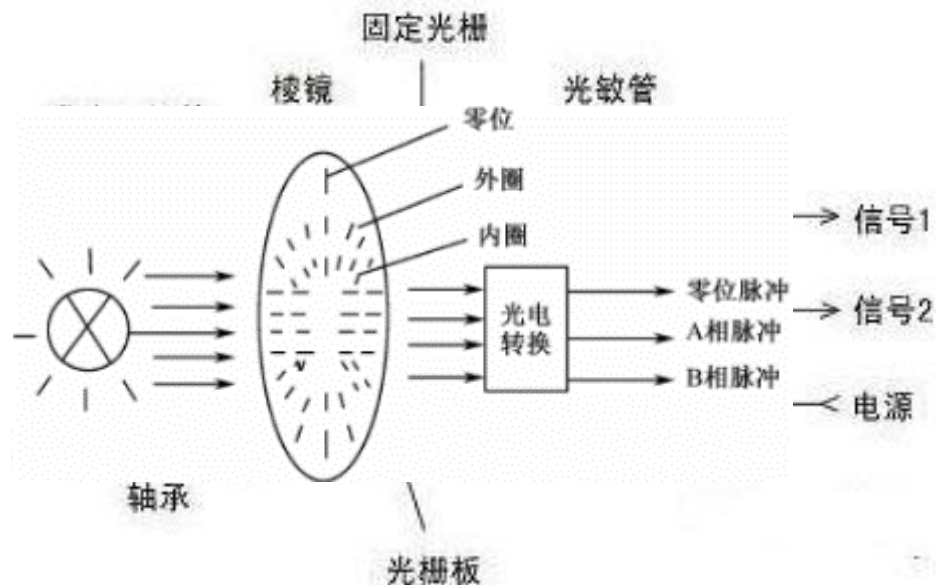
角编码器 {
绝对编码器
增量编码器

编码器

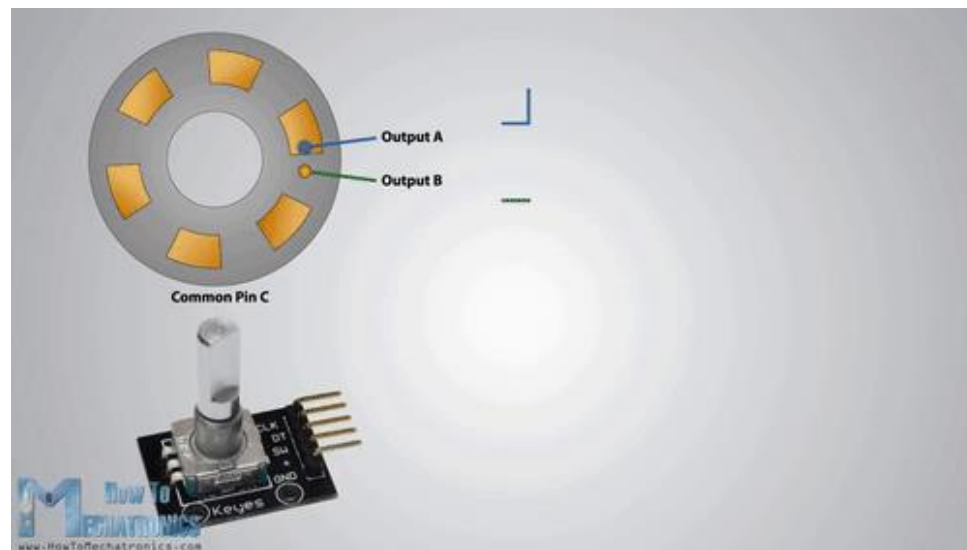
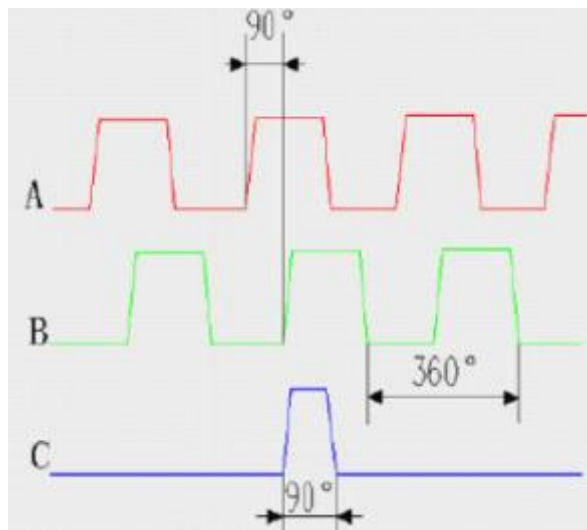


增量式角编码器

增量式编码器（即脉冲盘式编码器）是直接利用光电转换原理输出三组方波脉冲A、B和C相；A、B两组脉冲相位差 90° ，从而可方便地判断出旋转方向，而C相为码盘每转一圈就产生一个脉冲，用于基准点定位。

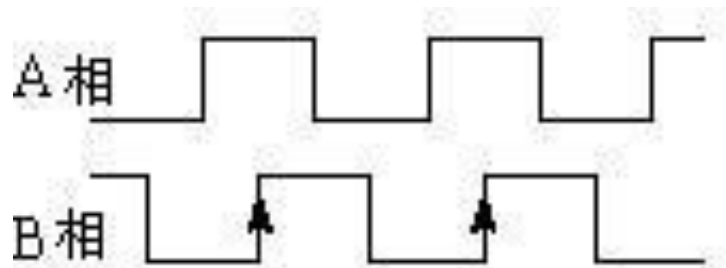


辨向信号



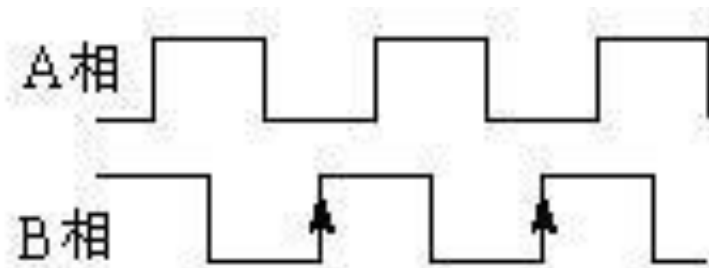
光电编码器的光栏板上有A组与B组两组狭缝，彼此错开1/4节距，两组狭缝相对应的光敏元件所产生的信号A、B彼此相差90°相位，用于辨向。当编码正转时，A信号超前B信号90°；当码盘反转时，B信号超前A信号90°。

旋转方向



a) 正转

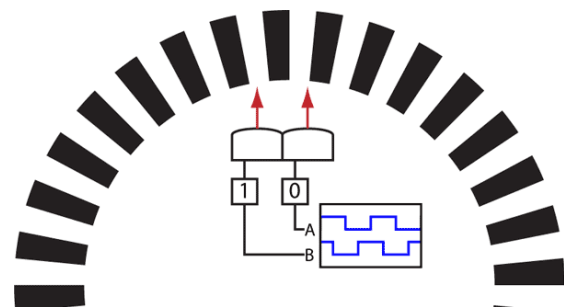
A路波形引前B路波形90度，即当B路脉冲由0上跳为1时，A路脉冲已是高电平



b) 反转

A路波形滞后B路波形90度，即当B路脉冲由0上跳为1时，A路脉冲是低电平

增量式编码器的分辨率及分辨率



增量式光电编码器的测量精度取决于它所能分辨的最小角度，而这与码盘圆周上的狭缝条纹数 n 有关，即最小能分辨的角度及分辨率为：

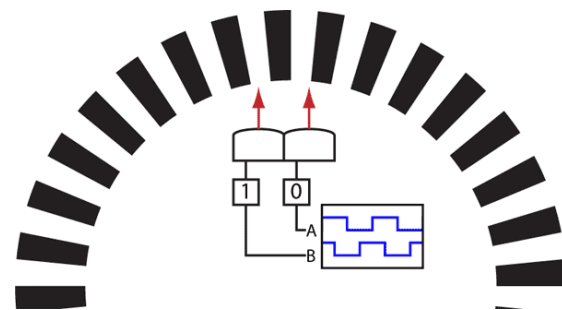
$$\alpha = \frac{360^{\circ}}{n}$$

$$\text{分辨率} = \frac{1}{n}$$

增量式编码器的分辨力及分辨率

编码盘

电动机

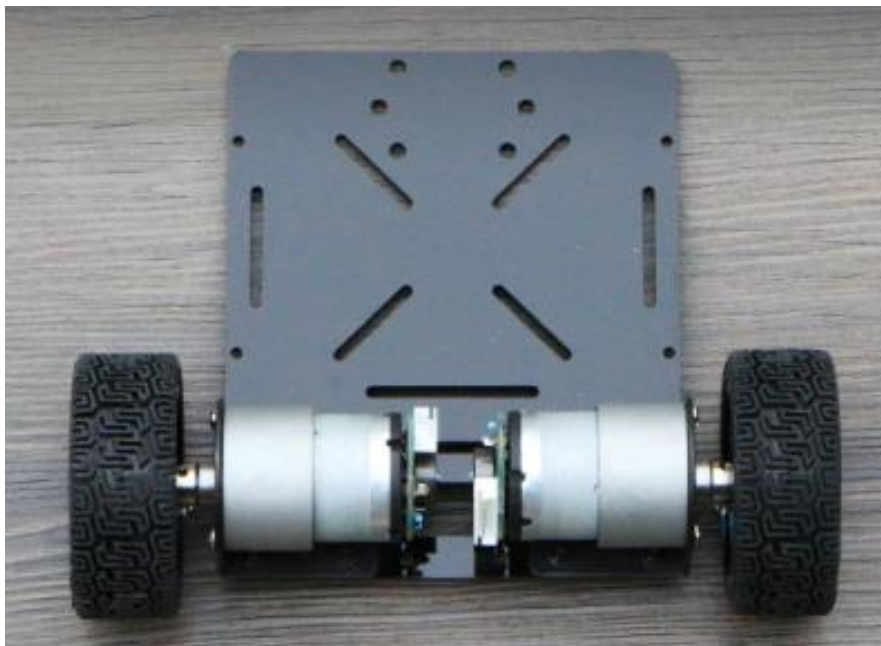


减速器



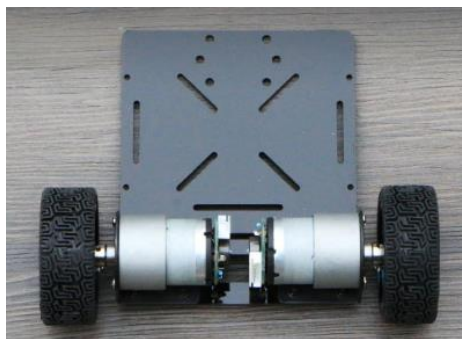
问题来了：轮子安装在输出轴上，轮子转动一圈，编码器的转了几圈？

编码器与电机轴直连，轮子的转动与编码器的转动比：减速比（减速器的减速比率）



上次课知识：
PWM可以控制电机转速。

如何利用**Arduino**测量
轮子的转速？

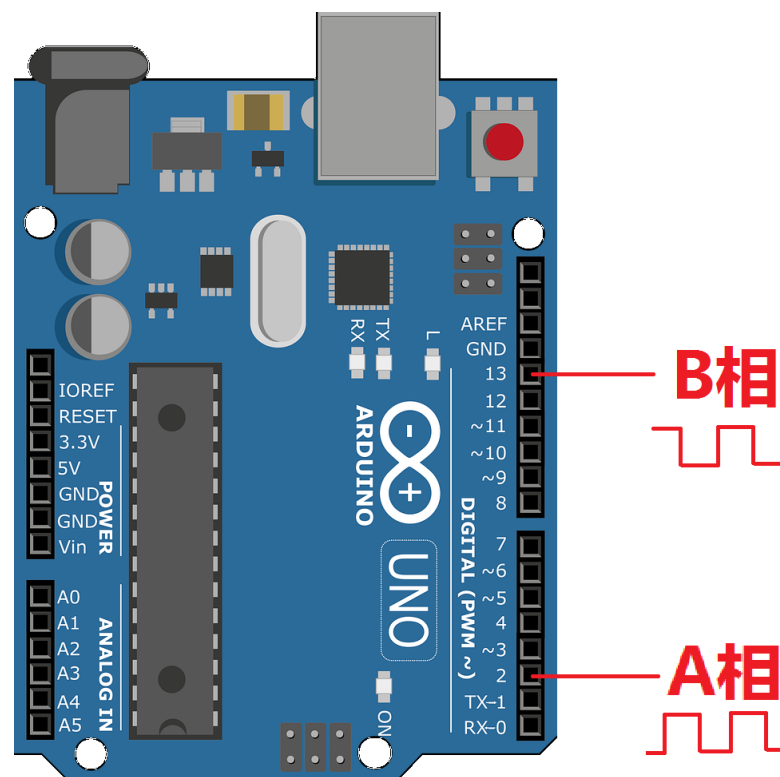


利用**Arduino**的中断：
每来一个跳变，就触发一次中断
，在中断处理函数里计数一次

原理：

把编码器A相输出接到单片机的外部中断入口，这样就可以通过跳变沿触发中断，然后在对应的外部中断服务函数里面，通过B相的电平来确定正反转。

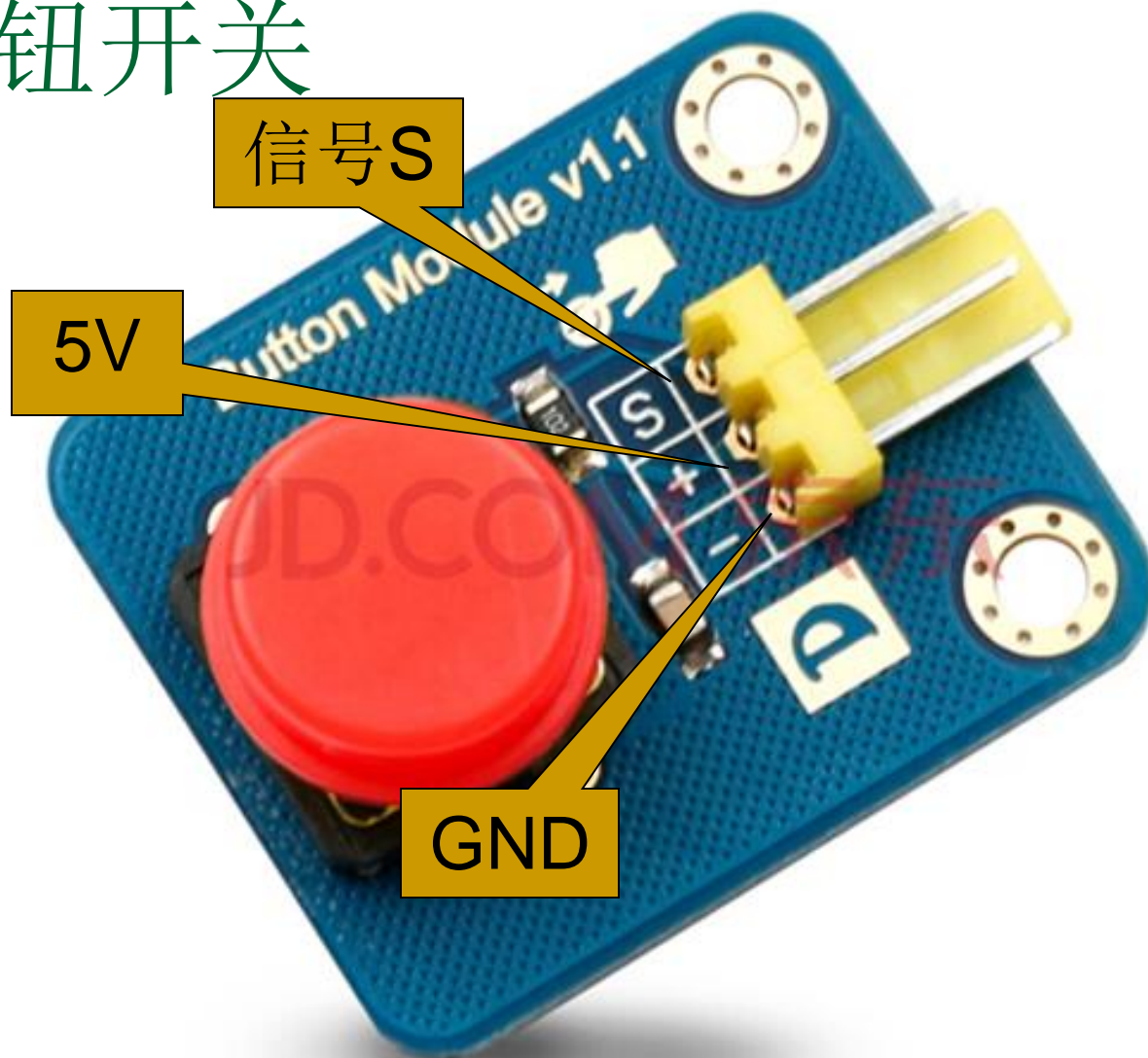
如当A相来一个上升跳变沿的时候，如果B相是低电平就认为是正转，高电平就认为是反转。



```
1 //定义左右编码器A B相引脚
2 #define ENCODER_A  2
3 #define ENCODER_B  13
4 void getEncoder(void) {
5     if (digitalRead(ENCODER_B) == LOW) {
6         encoderVal++; //根据另外一相电平判定方向
7     }
8     else {
9         encoderVal--;
10    }
11 }
12 void setup() { //编码器引脚都设置输入
13     pinMode(ENCODER_A, INPUT);
14     pinMode(ENCODER_B, INPUT);
15     attachInterrupt(0, getEncoder, RISING); //使能编码器引脚外部中断
16     Serial.begin(9600);
17 }
18 void loop() {
19     delay(1000);
20     Serial.print("encoder val = ");
21     Serial.print(encoderVal);
22     Serial.print("\r\n");
23 }
```

**思考：如果把RISING
改成CHANGE会怎么
样？**

4、按钮开关

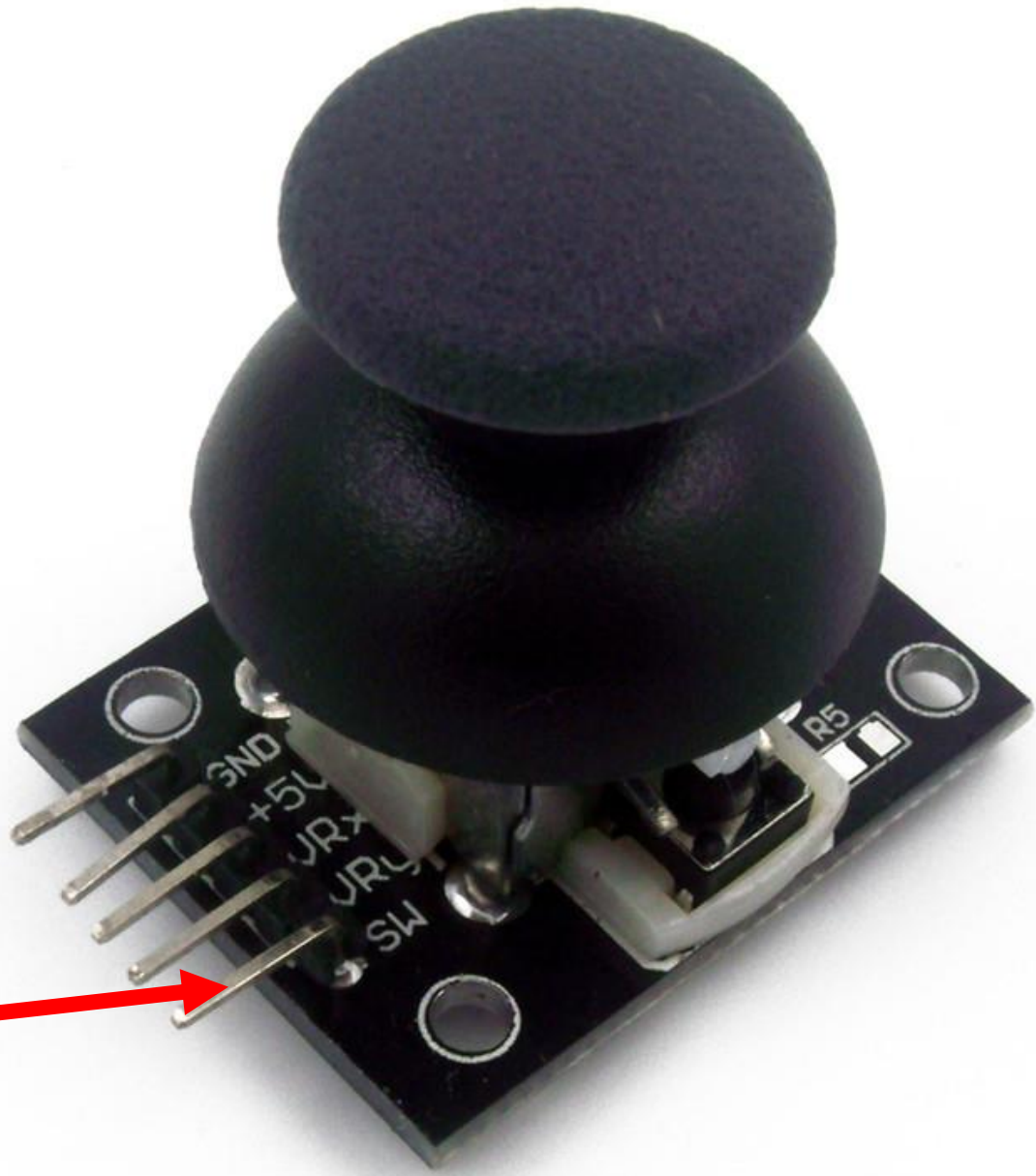


开关例程

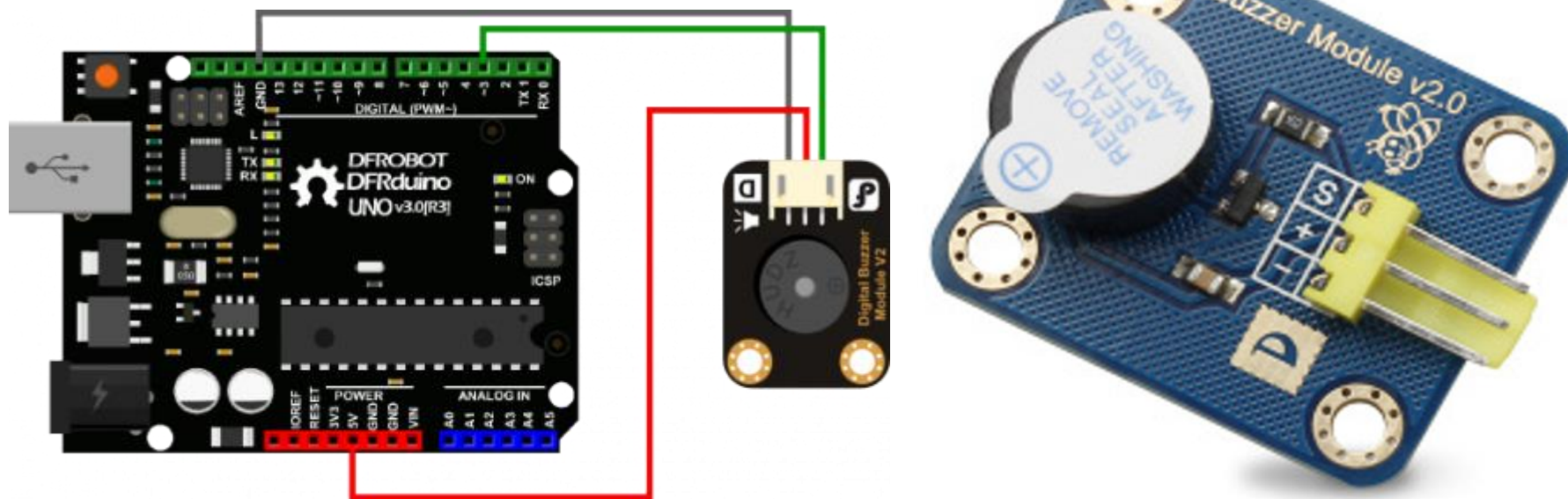
- 按键开关模块和数字**13** 接口自带**LED** 搭建简单电路，制作按键提示灯
 - 利用数字**13** 接口自带的**LED**，将按键开关传感器接入数字**3**接口，当按键开关传感器感测到有按键信号时，**LED** 亮,反之则灭.

```
■ int LED=13; //定义LED 接口
■ int BUTTON=3; //定义按键开关传感器接口
■ int val; //定义数字变量val
■ void setup()
■ {
■     pinMode(LED,OUTPUT);//定义LED 为输出接口
■     pinMode(BUTTON,INPUT);//定义按键开关传感器为输入接口
■ }
■ void loop()
■ {
■     val = digitalRead(BUTTON);//将数字接口3的值读取赋给val
■     if(val == HIGH) //当按键开关传感器检测有信号时，LED 点亮
■     {
■         digitalWrite(LED,HIGH)
■     }
■     else
■     {
■         digitalWrite(LED,LOW)
■     }
■ }
```


SW



5、数字蜂鸣器



■蜂鸣器是一种一体化结构的电子鸣响器，采用直流电压供电，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件；

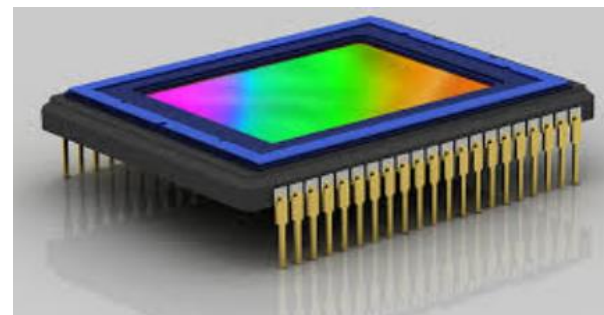
■高电平控制发声，低电平停止发声。

```
int SpeakerPin = 8; //控制喇叭的引脚
int Value = 10; //控制喇叭响的时间，可自行更改
```

```
void setup()
{
    pinMode(SpeakerPin, OUTPUT);
}
void loop()
{
    digitalWrite(speakerPin, HIGH);
    delay(value); //调节喇叭响的时间;
    digitalWrite(speakerPin, LOW);
    delay(value); //调节喇叭不响的时间;
}
```

思考：电池电量不足了，如何利用蜂鸣器来发出警报。

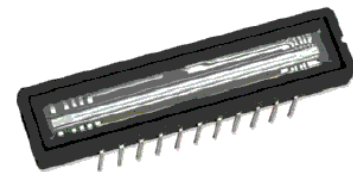
6、线性CCD TSL1401



常规摄像头：2D图像(数字图像处理)

线性CCD或摄像头：1D图像

传真机，扫描仪，条形码读取卡



6、线性CCD TSL1401CL

TSL1401是一种128个光电二极管组成的感光阵列，阵列后面有一排积分电容，光电二极管在光能量冲击下产生光电流，构成有源积分电路，那么积分电容就是用来存储光能转化后的电荷。积分电容存储的电荷越多，说明前方对应的那个感光二极管采集的光强越大。反映在像素点上就是，像素灰度低。光强接近饱和，像素点灰度趋近于全白，则呈白电平。



VCC: 接电源正极 3.3v-5v

GND: 接地

CLK: 时钟信号线

SI: 逻辑信号线

AO: 模拟信号输出

6、线性CCD TSL1401

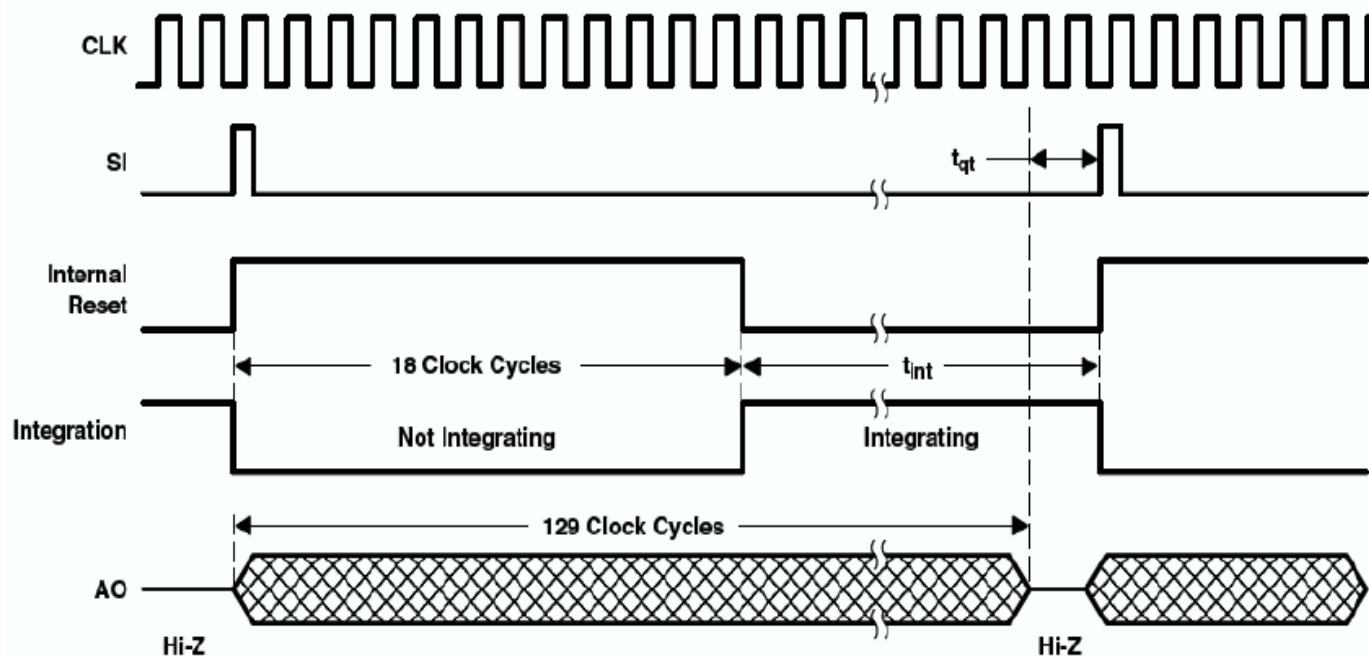


图 1。时序波形

前18个时钟周期是像素复位时间，不进行积分与曝光。而且，第一个逻辑时钟SI必须出现在下一个时钟信号CLK上升沿之前。从时序图可清晰的看出CCD的操作过程，SI信号相当于一个标志，当它变为高电平后，我们就可以在每个CLK信号高电平到来后进行数据的AD采样。

6、线性CCD TSL1401

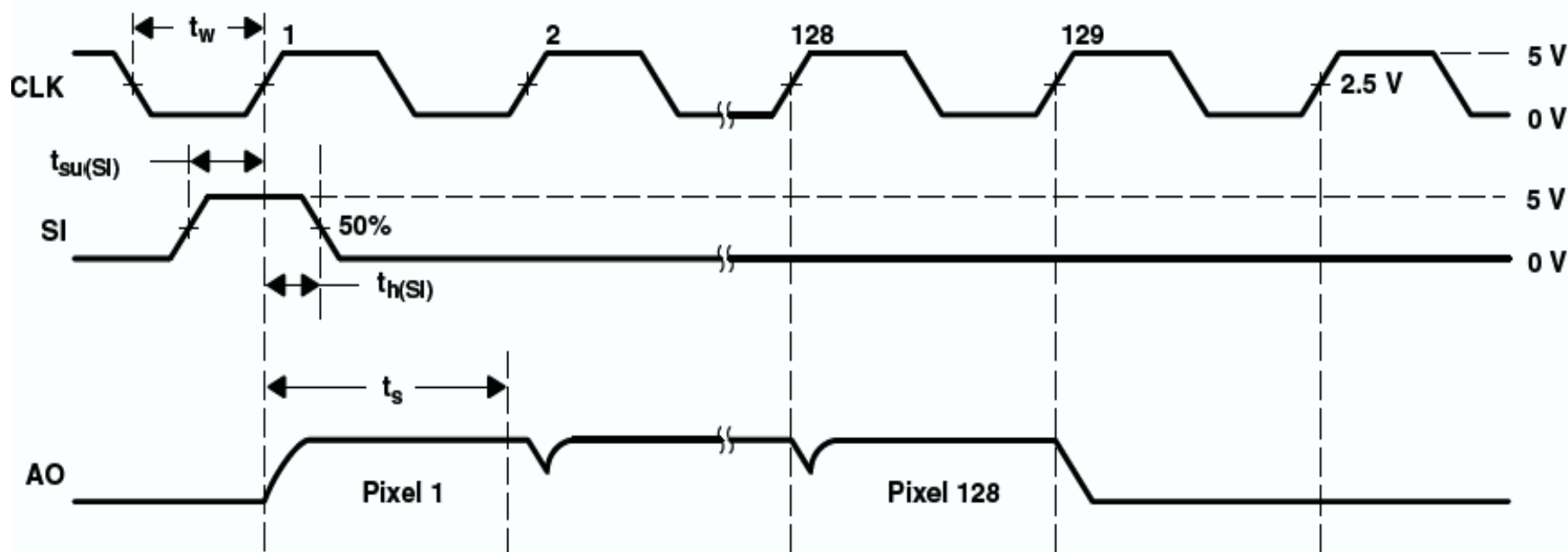


图 2。操作波形

TSL1401的数据读取包括两个部分：1. 曝光；2. 读取数据
两者都遵从图2中的操作波形，操作基本一样，区别在于是否读数据


```

1  digitalWrite(CCD_SI,HIGH);//SI拉高电平
2  digitalWrite(CCD_CLK,HIGH);//时钟高电平
3  digitalWrite(CCD_SI,LOW);//SI低电平
4  digitalWrite(CCD_CLK,LOW);//时钟低电平
5  for(i=0;i<129;i++)
6  {
7      digitalWrite(CCD_CLK,HIGH);
8      digitalWrite(CCD_CLK,LOW);
9  }//从这里结束曝光
10 delayMicroseconds(exp_time);//曝光时间
11 digitalWrite(CCD_SI,HIGH);
12 digitalWrite(CCD_CLK,HIGH);
13 digitalWrite(CCD_SI,LOW);
14 pixe1[0]=analogRead(A0)>>2;
15 if(pixe1[0]==255) pixe1[i]=254;
16 digitalWrite(CCD_CLK,LOW);
17 for(i=1;i<128;i++)
18 {
19     digitalWrite(CCD_CLK,HIGH);
20     pixe1[i]=analogRead(A0);
21     digitalWrite(CCD_CLK,LOW);
22 }
23 digitalWrite(CCD_CLK,HIGH);
24 digitalWrite(CCD_CLK,LOW);

```

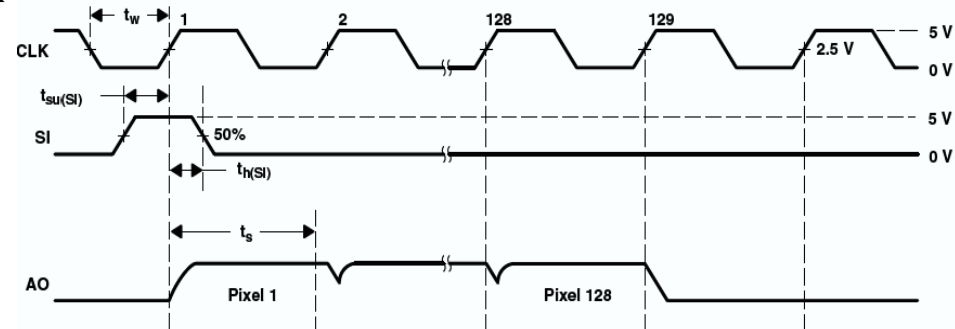
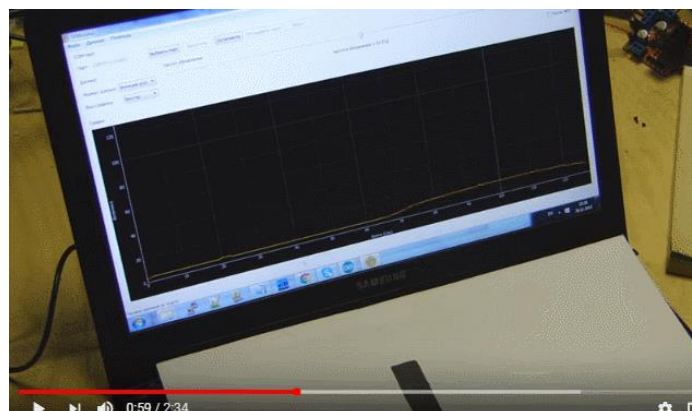


图 2。操作波形



```
1  digitalWrite(CCD_SI,HIGH);//SI拉高电平
2  digitalWrite(CCD_CLK,HIGH);//时钟高电平
3  digitalWrite(CCD_SI,LOW);//SI低电平
4  digitalWrite(CCD_CLK,LOW);//时钟低电平
5  for(i=0;i<129;i++)
6  {
7      digitalWrite(CCD_CLK,HIGH);
8      digitalWrite(CCD_CLK,LOW);
9  }//从这里结束曝光
10 delayMicroseconds(exp_time);//曝光时间
11 digitalWrite(CCD_SI,HIGH);
12 digitalWrite(CCD_CLK,HIGH);
13 digitalWrite(CCD_SI,LOW);
14 piexl[0]=analogRead(A0)>>2;
15 if(piexl[0]==255) piexl[i]=254;
16 digitalWrite(CCD_CLK,LOW);
17 for(i=1;i<128;i++)
18 {
19     digitalWrite(CCD_CLK,HIGH);
20     piexl[i]=analogRead(A0);
21     digitalWrite(CCD_CLK,LOW);
22 }
23 digitalWrite(CCD_CLK,HIGH);
24 digitalWrite(CCD_CLK,LOW);
```

思考：与前面的寻线
传感器有什么不同？
孰优孰劣？

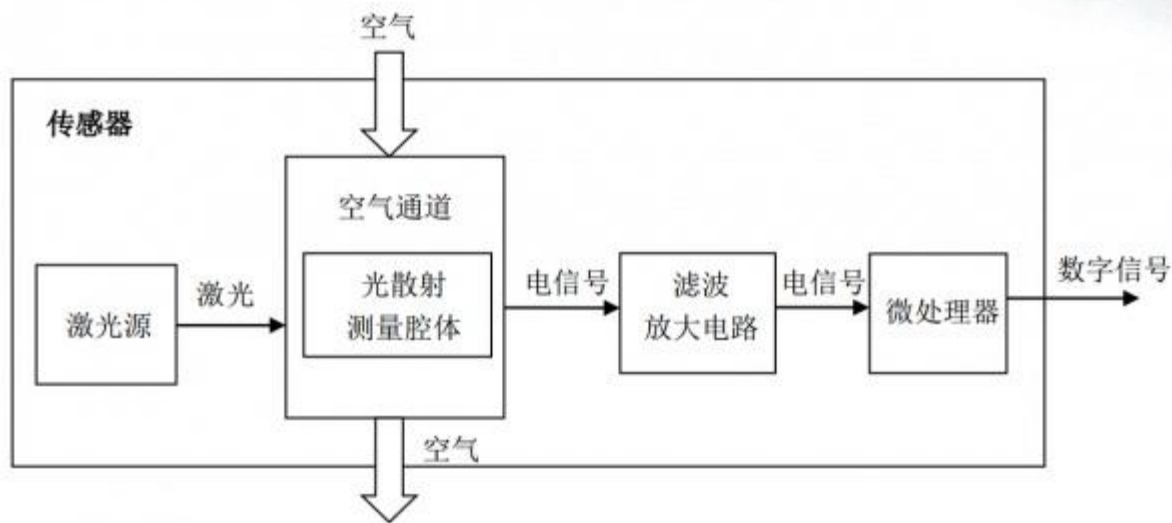
7、人体红外传感器



HC-SR501是一个红外线感应模块，它依靠特定温度（36-38）的物体运动来判断人体。因此可以作为报警器的关键模块。在关键的地方，如门口，放上这么一个传感器，可以起到防盗的作用。它有两个调节旋钮，一个调节最远探测距离，一个调节延时时间，具体参数如上。当人走过或停留在感应范围中，模块通过D0发送高电平信号。

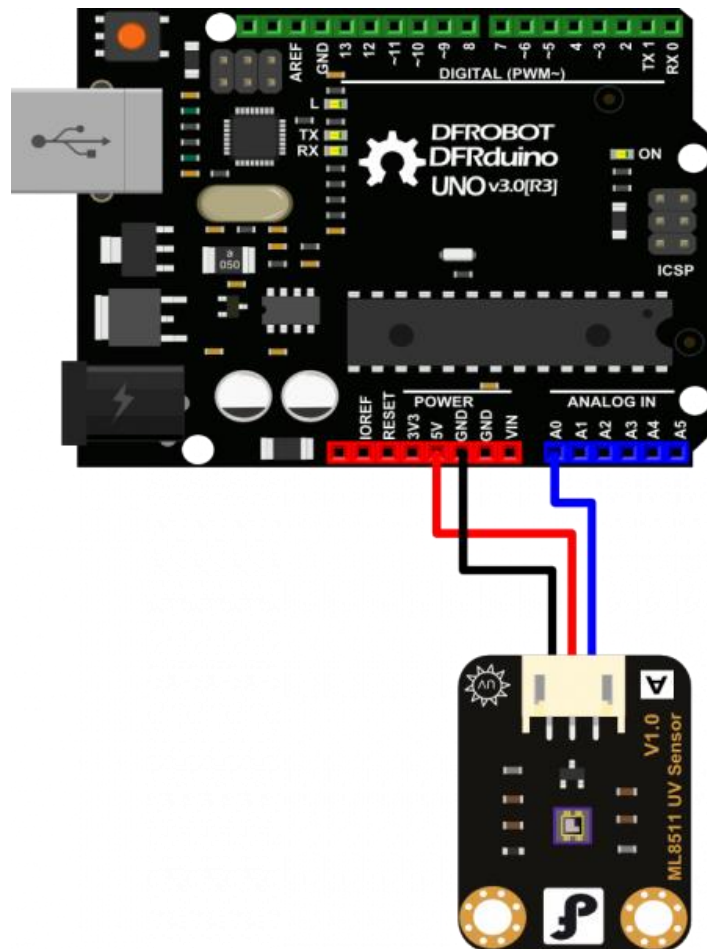
8、PM2.5灰尘传感器

- 用于获得单位体积内空气中0.3~10微米悬浮颗粒物个数，即颗粒物浓度

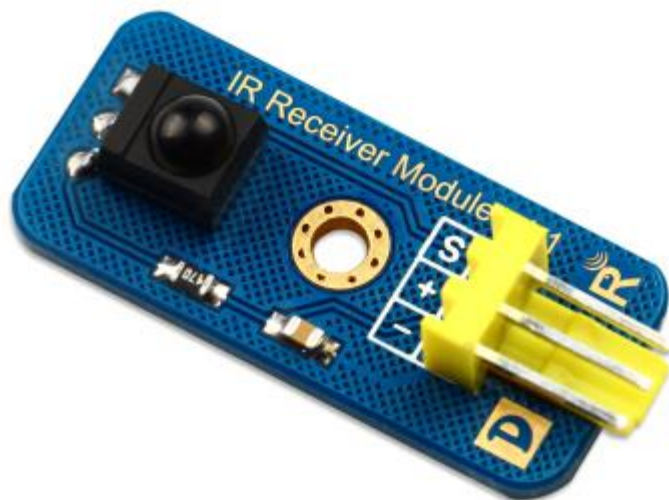


9、紫外线传感器

- 用来检测室内或室外的紫外线密度

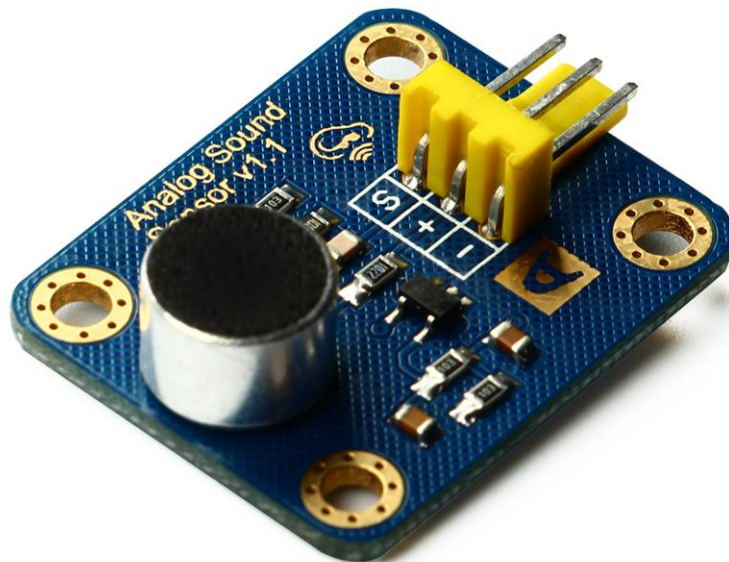


10、红外接收头



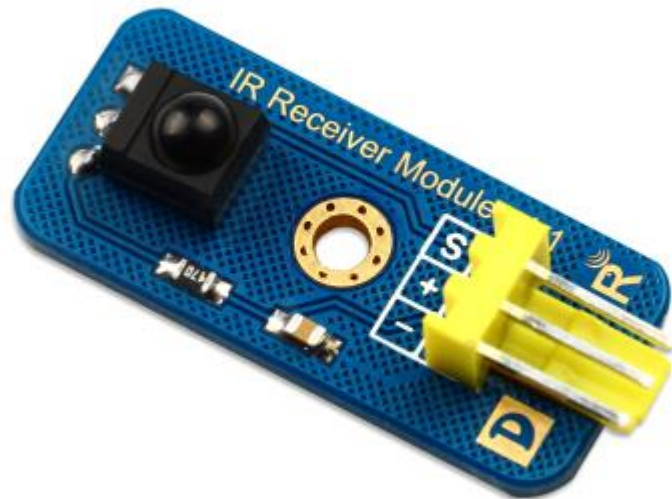
- 红外接收头，和家用遥控器使用的红外接收头一致，接收38K的调制信号。通过Arduino解码。

11、麦克风声音传感器



- 输出模拟电压值，使得您只需采集模拟量电压就可以读出声音的幅值，判断声音的大小。

12、火焰传感器



- 可以用来探测火源或其它一些波长在760纳米~1100纳米范围内的热源，探测角度达60度，其中红外光波长在940纳米附近时，其灵敏度达到上限。此火焰传感器在灭火机器人比赛或者搜救机器人比赛中起着非常重要的作用。

13、酒精含量传感器



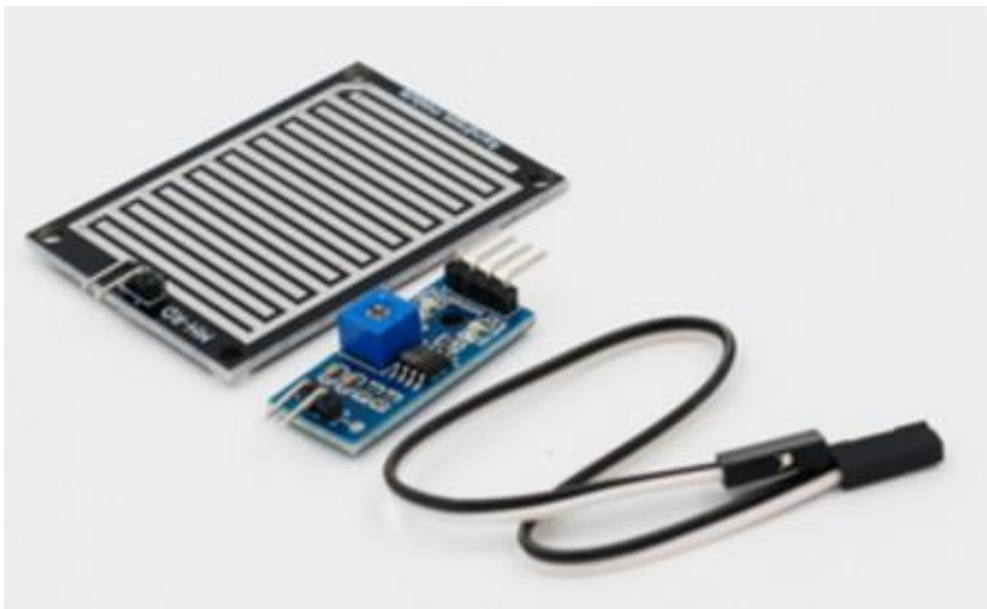
- 使用气敏材料二氧化锡(SnO_2)。当传感器所处环境中存在酒精蒸汽时，传感器的电导率随空气中酒精气体浓度的增加而增大。气体传感器对酒精的灵敏度高，可以抵抗汽油、烟雾、水蒸气的干扰。

14、土壤湿度传感器



- 可以检测土壤的湿度。

15、雨水检测传感器



- 可以检测是否下雨。

16、气压传感器



- **BMP180**是一款高精度、小体积、超低能耗的压力传感器，可以应用在移动设备中它的性能卓越，精度最低可以达到0.03hPa，并且耗电极低，只有3 μ A。**BMP180**采用强大的8-pin陶瓷无引线芯片承载（LCC）超薄封装，可以通过I2C总线直接与各种微处理器相连。

■ 认认真真课后思考

- 什么叫做中断，中断有何用处？
- 使用中断设计一个电路和程序，当有人按下按钮**A**后，直流电机正转将教室的投影屏幕卷起，按下按钮**B**后，控制直流电机反转将教室投影屏幕放下。
- 利用micros()设计一个按键游戏，比比谁按键按得快，给出代码；
- 结合本堂课程，预习下次实验课程内容。（大家提前敲好代码，实验时考到实验电脑上）
- 如何利用蜂鸣器来给电池电量报警？

■ 课后学习参考网站

<https://www.arduino.cc/en/>

注意事项:

- 大家**一定**要预习实验指导书
- 不理解代码就一顿编译，是对自己不负责任的行为，也是折腾实验老师的一种“好”方式

针对下一次课：

- 复习“自动化专业导论”关于自动控制的知识点
- 预习机器人系统组成部分
- 以循迹机器人为例，预习比例微分积分控制器

网上资料参考：

<https://blog.csdn.net/tuxinbang1989/article/details/83058243>

<https://zh.wikipedia.org/wiki/PID%E6%8E%A7%E5%88%B6%E5%99%A8> PID