灰尘传感器 GP2Y1010AU0F 使用说明

GP2Y1010AU0F 是日本夏普公司开发的一款光学灰尘浓度检测传感器。此传感器内部成对脚分布的红外发光管和光电晶体管，利用光敏原理来工作。用于检测特别细微的颗粒，如香烟颗粒、细微灰尘。依靠输出脉冲的高度来判断颗粒浓度。

传感器的特点：

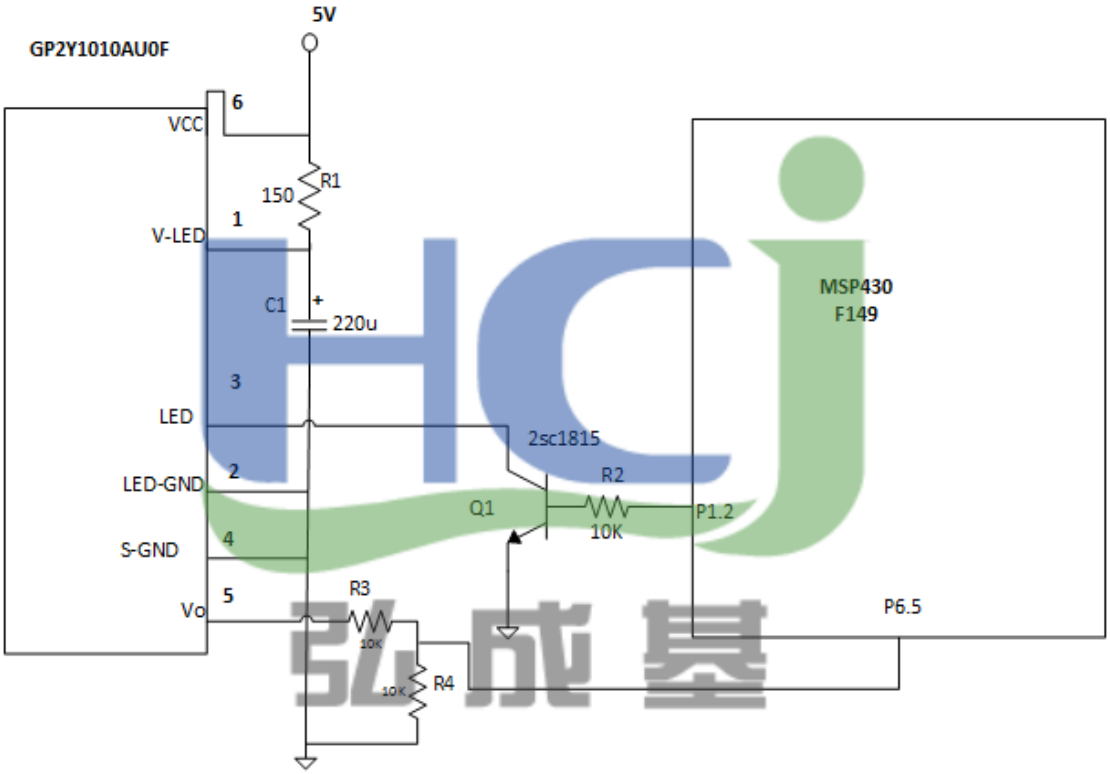
1. 尺寸：（46.0 x 30 x 17.6 mm）
2. 最大工作电流：20mA
3. 单脉冲即可检测出颗粒浓度。（待工作稳定）
4. 工作温度：-10~65℃
5. 安全无害

应用实例：检测空气中颗粒浓度，用于空气净化器、空气监测器

说明：具体尺寸、引脚说明、内部电路读者可以参考 datasheet，本文侧重说明传感器应用的具体电路，使用中存在的注意事项，以及相应的开发代码。

1 GP2Y1010AU0F 与 MSP430F149 的使用说明：

1.1 电路连接图：



图— 用MSP430搭建的灰尘测试系统

1.2 电路说明：

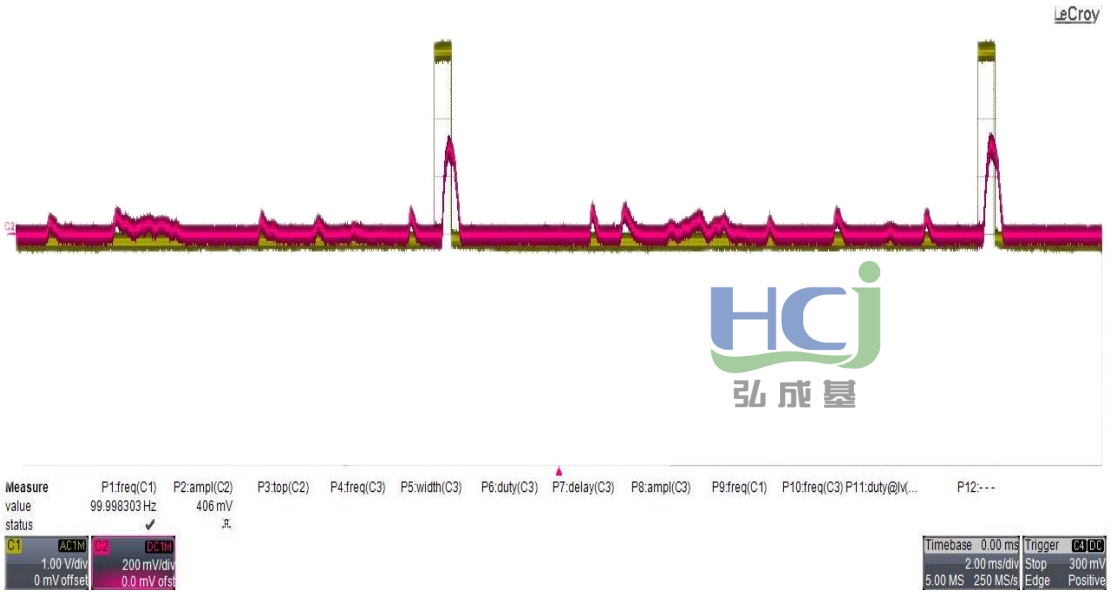
- 电阻 R2 起到限流的作用，晶体管 Q1 可以增加 3 号脚的驱动能力。R2 建议 10K，Q1 建议 2SC1815（npn 晶体管）

- 430 的 P1.2 作为脉冲的输入脚，为传感器提供输入信号
- 传感器的引脚按照 datasheet 中典型电路来接，R1、C1 分别为 150Ω和 220uF 的电解电容
- P6.5 为 430 内部 ADC 的 5 号模拟输入脚
- R3 和 R4 起分压作用，分别为 10K，因为对于 430 单片机，驱动电压为 3.2V，而对于传感器而言，需要提供 5V 的电压，输出电压有可能会超过 3.2V，当然这不仅仅取决于此。对应的 A5 进行采样时设定的参考电压正好为 3.2V 的缘故，对于 430 单片机，是有相应的引脚作为输入参考电压的。因此分压模块的设置非常灵活。如果不采用分压模块，最好在 5 号输出接一只大值电阻到地
- 电路最好采取焊接的方式，因为面包板搭建的电路会因为面包板分布电容的原因与最终结果出现偏差

1.3 信号与结果：

对于输入信号的几点说明：输入信号采用周期为 10ms±1ms；输入高脉冲的宽度为 0.32ms±0.02ms。

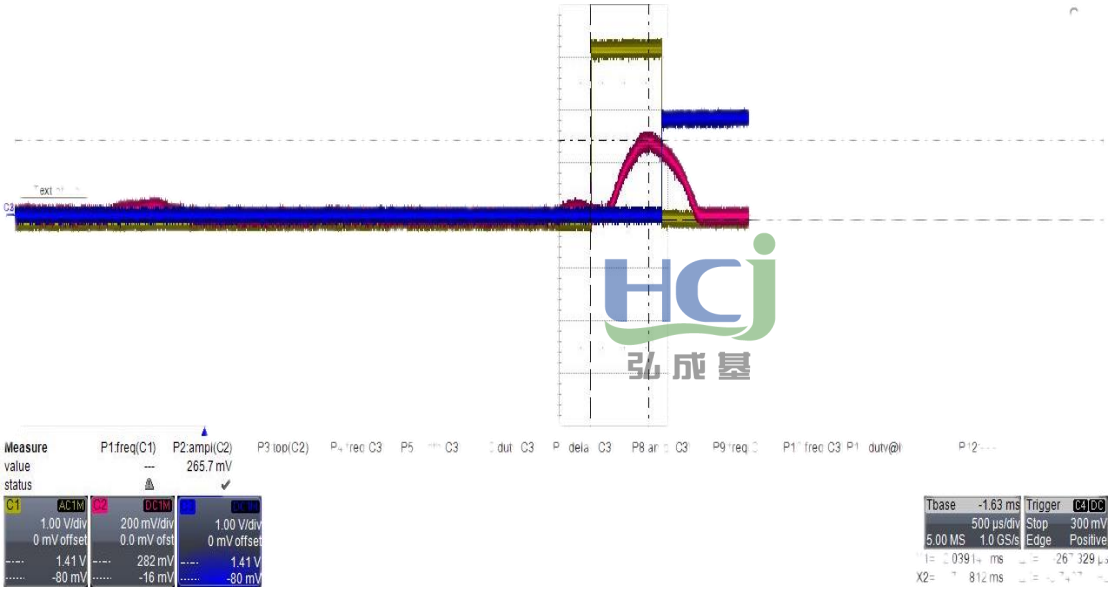
对于采样信号，需要等待输入信号出现高脉冲以后计时 0.28ms 进行采样。具体板子搭建成功后的信号示意图如下图所示。



图二 输入输出信号示意图

图中黄色的信号线是输入信号，可以看到其波形频率大概为 100HZ ，也就是周期为 10ms，另外可以看到输出分压以后的波形（红线），当输入高脉冲过后不久，便出现了红色输出的高脉冲。

为了更加清晰的观测红色输出波形，我们可以来观察图三。



图三 采样及输出的具体波形

图三反应了具体的输出波形，可以从时间轴测量，在输入上升沿到输出的峰值，时间大概是 0.28ms 左右，这与 datasheet 上的说明相符。其中蓝色的线代表 AD 采样的时刻，但是由于示波器触发设置的原因，导致蓝色的线条在输入信号的下降沿才发生跳变，其实这里相当于滞后了，读者可以心领神会。

1.4 代码说明：

由于参考代码过多，在此对于常见的代码不予讲解，下面对于核心代码进行说明。

430 单片机特点：16 位处理器，低功耗，8M 的主系统时钟决定单指令周期为 0.125us，内部集成众多外设，包括 12 位 ADC，定时器 A，定时器 B。强调一点是，由于此传感器要求采样时序较为精确，因此像 430 这种中端的微处理器或者更高端的处理器比较适合开发此传感器，也就是说指令的处理速度越快越好。

430 单片机带动传感器工作一次的时间大概为 1.5s，工作期间不间断提供输入脉冲，大概到 1s 左右，此时传感器的工作已经稳定，因此可以开始采样了，连续采样 4 次，然后将数据取平均值，这就是一次工作的整个流程。这样的设计思路既可以保证传感器的读值稳定性，又可以做到低功耗。

输入信号由定时器 A 产生，P1.2（TA1）输出，ADC 采样时序的控制由定时器 B 控制，使用片内 12 位的 ADC 进行采样，采样的参考电压是系统电压 3.2V（做法比较粗糙），采样后直接转成模拟信号的表达形式，最后输出的是电压值，对于电压转换到 PM2.5 的值，在 1.5 节有简单说明。

```
#include <msp430x14x.h>
#define   uchar     unsigned char
#define   uint      unsigned int

uint count_A=0;   //对 count_A 进行计数，当 count_A>100,即传感器稳定后进行采样
uchar count_B=0; //单独工作一次 ADC 采样的次数,当 count_B=4 时完成一组工作
uint PM2_5;       //PM2.5 的数字值
uchar ptr[3];     //PM2.5 的模拟值
```

```
//这一模块可以更改输入信号的端口，读者可以在自己的系统取合适端口，在此取 P1.2 口也//就是 TA1，因为我们是利用定时器 A 来产生传感器输入信号的
#define PWM BIT2
void PWM_IO()
{
    P1SEL|=PWM;//P1.2 为 PWM 波形的输出口
    P1DIR|=PWM;
}
//---------------------GP2Y1010AU-----------------------------------------------------------------------//
//*****************************************************************************
//函数名：timerA_start_PWM()
//输入：无
//输出：无
//功能描述：产生 PWM 波形
//*****************************************************************************
void timerA_start_PWM(void)
{
    TACTL|=TACLR;
    PWM_IO();  //使 P1.2 为 PWM 的输出口
    TACTL|=TASSEL1+ID1+ID0+MC0;//选择子系统时钟且分频，时钟为 SMCLK/8;加计数模式
    CCTL1|=OUTMOD0+OUTMOD1;//置位/复位模式
    CCTL0|=CCIE;//开启捕获比较中断
    CCR0=10000;//周期为 10ms
    CCR1=9680;//高电平时间为 0.32ms
}
//*****************************************************************************
*
//函数名：timerB_start_996()
//输入：无
//输出：无
//功能描述：确定 ADC 启动采样的时间，为定时器 A 中断后，定时器 B 定时 9.96ms
//*****************************************************************************
*
void timerB_start_996()
{
   TBCTL|=TBCLR;
   TBCTL|=TBSSEL1+ID1+ID0+MC0;//选择子系统时钟且分频，时钟为 SMCLK/8;加计数模式
   TBCCR0=9950;//经过实验调整，9.95ms 采样最好，其一是启动定时器会耗费一定时间，
//其二是由于本身产品存在的一定误差，因此这个值可以由读者自己实践确定
   TBCCTL0|=CCIE; //开启比较捕获中断
}
//*****************************************************************************
*
```

```
//函数名：GP2Y1010AU_work()
//输入：无
//输出：无
//功能描述：使传感器开始工作，并且经过 ADC 采样
//*****************************************************************************
*
void GP2Y1010AU_work()
{
  timerA_start_PWM();//开启定时器 A 并产生 PWM 波
}
//*****************************************************************************
*
//函数名：ADC_Int()
//输入：无
//输出：无
//功能描述：ADC 设置的初始化
//*****************************************************************************
*
void ADC_Int()
{
  ADC12CTL0|=SHT0_2+ADC12ON;//采样保持时间选为 16 个 ADC12CLK,打开 ADC
  ADC12CTL1|=SHP+ADC12SSEL0+ADC12SSEL1;//SAMPCON 采用时序电路产生的信号,时钟
SMCLK
  ADC12MCTL0|=INCH0+INCH2;//选择 AVCC 和 AVSS 作为参考电压，选择 A5 通道
  ADC12IE|=BIT0;//允许 ADC12MEM0 中断
  ADC12CTL0|=ENC;//使能转换
  P6SEL|=BIT5;//P6.5 为模拟信号输入端
}
//*****************************************************************************
*
//函数名：Tran_Val(uint Hex_Val)
//输入：数字量
//输出：模拟量
//功能描述：ADC 数字量向模拟量转变
//*****************************************************************************
*
void Tran_Val(uint Hex_Val)  //为了不出现浮点数的计算，整个对电压放大 100 倍进行计算
{
  unsigned long caltmp;
  uint Curr_Volt;
  uchar t1;
  caltmp=Hex_Val;
  caltmp=(caltmp<<5)+Hex_Val;//caltmp=Hex_Val*33
  caltmp=(caltmp<<3)+(caltmp<<1);//caltmp=caltmp*10
```

```
    Curr_Volt=caltmp>>12;
    ptr[0]=Curr_Volt/100;//Hex->DEC 变换，百位
    t1=Curr_Volt-(ptr[0]*100);
    ptr[1]=t1/10;//十位
    ptr[2]=t1-(ptr[1]*10);//各位
}

void main()
{
    WDTCTL = WDTPW + WDTHOLD; //禁止看门狗
    InitSys();//系统初始化
    P3DIR|=BIT0;//P3.0 为输出  P3.0 和 P3.1 在此均做测试用，因为这两个引脚分别接有两个
                  //LED 灯
    P3DIR|=BIT1;//P3.1 为输出
    P3OUT&=~BIT1;
    P3OUT&=~BIT0;
    LCD_IO_set();
     LcdReset();              //LCD 初始化
     FYTOOJASK3000();
    ADC_Int();
    _EINT();//使能中断，这是一个 C 编译器支持的内部过程

    while(1)
    {
        GP2Y1010AU_work();
        Delay2s();   //延时 2 秒，延时至少超过一次传感器的工作时间，也就是 1.5 秒，主要
这里用的是 while，不断循环，要等到这次工作结束再进行下次工作
        wen_xianshi();//通过 lcd（1602 液晶）显示
    }
}
/*********************************************************************
                          中断处理
*********************************************************************/
//-------------------------------------------------------------------
//函数名：timer_B()
//输入：无
//输出：无
//功能描述：当定时器 B 计数结束后，通知 ADC 进行采样转换
//-------------------------------------------------------------------
#pragma vector=TIMERB0_VECTOR
__interrupt void Timer_B(void)
//定时器 A 的 CC0 中断处理程序
//TIMERA0_VECTOR=13*2,等于基地址 0XFFE0+26=0XFFFA
          {
```

```
            TBCTL&=~MC0;
            P3OUT^=BIT0;
            //P3OUT&=~BIT1;
            ADC12CTL0|=ADC12SC;//进行一次 AD 转换
            count_B++;
          }
//-------------------------------------------------------------------
//函数名：timer_A()
//输入：无
//输出：无
//功能描述：对定时器 A 的中断次数的统计
//-------------------------------------------------------------------
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A(void)
//定时器 A 的 CC0 中断处理程序
//TIMERA0_VECTOR=6*2,等于基地址 0XFFE0+12=0XFFEC
          {
            count_A++;
            //P3OUT&=~BIT0;
            P3OUT^=BIT1;
              switch(count_A)//每隔五次，进行采样一次
                {
                  case 201:timerB_start_996();break;
                  case 211:timerB_start_996();break;
                  case 221:timerB_start_996();break;
                  case 231:timerB_start_996();break;
                  default:
                    if(count_A>250)
                    {
                      count_A=0;
                      TACTL&=~MC0;
                    }
                }
          }

#pragma vector=ADC_VECTOR
__interrupt void ADC12_ISR(void)
//定时器 A 的 CC0 中断处理程序
//ADC_VECTOR=7*2,等于基地址 0XFFE0+12=0XFFEEH
{
  while((ADC12CTL1&ADC12BUSY)==1);
  switch(count_B)
  {
    case 1:PM2_5+=ADC12MEM0;break;
```

```
        case 2:PM2_5+=ADC12MEM0;break;
        case 3:PM2_5+=ADC12MEM0;break;
        case 4:                          //ADC 的中断优先级比较高，因此，两次工作的间隔有
                                         //要求
          {
            count_B=0;
            PM2_5+=ADC12MEM0;
            PM2_5=PM2_5>>2;//求取平均值
            Tran_Val(PM2_5);//将其转化成模拟值
          }
    }
}
```

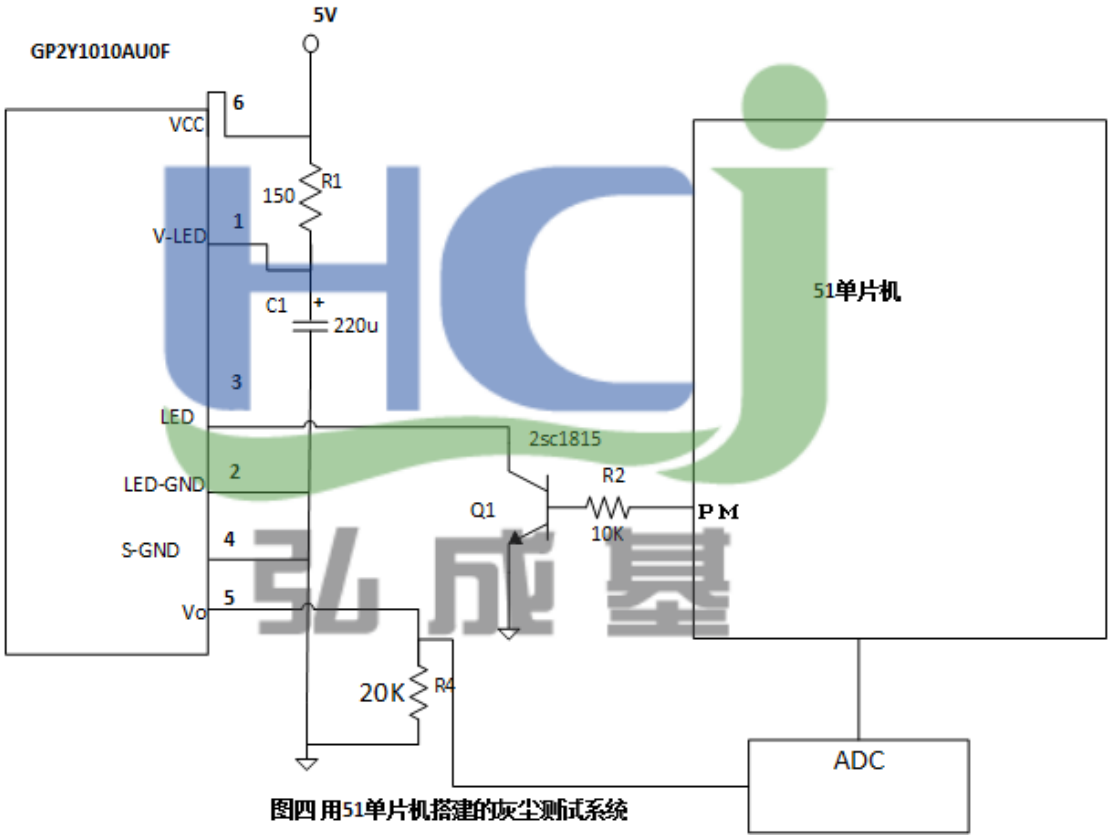
整个代码有液晶模块没有写入，但是液晶模块过于简单，且现成代码居多，还有系统初始化以及延时模块没有具体写入，只写出代码的关键部分，其次，代码的思路简单，适合大多数使用。

1.5 后续数据处理：可以看出代码中得出的为电压值，而不是所谓的 PM2.5 的值。其实在 datasheet 中含有电压和颗粒浓度的对应关系，但是整个图显得很粗糙。有几个问题需要说明：一是对于不同的传感器，图中显示的截距可能不同，我在相同条件下测量的结果是最大会有 0.2V 左右的极差；二是，图中的曲线并不一定完全精确，重要的是它不是一条直线，我建议可以将其打印出来，放大，利用尺子规划成三段直线来使用。综合以上两个问题，我个人认为对于精确使用，校正工作还是很必要的，而且最好对每个传感器都进行校正再使用，这和电子血压计比较相像，电子类的应用产品普遍都会有这种毛病，因为过于灵敏，当然这些也不会太麻烦，由于实验条件的限制，校验的具体说明我在此并未给出。

2 GP2Y1010AU0F 与 51 单片机的使用说明：

51 单片机是为最简单的微处理器，存在好用易学，性价比超高，适合简单的控制使用，当然，也可以用于开发这款传感器。至于 51 的资料非常多，易查询，在此我就不多说了。

电路连接如图四所示。



图四 用51单片机搭建的灰尘测试系统

电路的输出接个 20K 的电阻，直接给 ADC，这是因为，51 系统一般是 5V 系统，不需要分压环节。其次 51 单片机自身是不带 ADC 的，因此 ADC 的采样芯片是需要另外加的，ADC 有很多款，具体选型依据读者自身来定，因此在接下来的程序中不涉及 ADC 的采样时序，读者可以依据自身的应用来编写程序。

此处的程序只给出输入信号的产生程序，程序中 PM 为传感器的输入信号，利用定时器 0 工作在方式一来不断改变 PM 的值，最终产生输入信号。

程序：

```
 PM=0;//传感器的输入信号
 TMOD=0x11;//设置定时器 0 为工作方式 1
 TH0=(65536-40)/256;
 TL0=(65536-40)%256;
 ET0=1;//开定时器 0 中断
 TR0=1;//启动定时器 0
 EA=0; //开总中断

void timer0() interrupt 1
{

 TH0=(65536-40)/256;
 TL0=(65536-40)%256;
 N++;
 //if(N==7)
   //PM=1;
```

```
  if(N==8)
    PM=0;
  if(N==250)
  {
    PM=1;
    N=0;
   }
}
```

在此输入信号下的输出与图二、图三相似，在此不再贴图，读者可以参考图二图三，至于后续数据处理，在前一个例子中（1.5 节）也给出了说明。

3 疑问解答

讲述上述两个例子，还有一些读者经常询问的问题，我在此做出强调：

- 传感器是光敏原理，我们可以看到内部的光线吗？
  传感器是利用红外光进行工作的，因此是肉眼不可见的
- 面包板可以进行电路的搭建吗？
  面包板由于寄生电容的原因，很少能够成功的，建议采用焊接的电路板
- 为了测试这个传感器，我应该需要什么设备？
  最好有单片机系统，其次配套示波器观察波形
- 传感器的稳定性如何？
  这款传感器是相当稳定的，当然电子类的这种东西本身是比较敏感的，如果要求超高的稳定性，不妨从算法入手，因为实际中会有 0.1V 的波动
- 传感器的灵敏度如何？
  这款传感器是相当灵敏的，稍微向小口哈一下气，可以看到脉冲迅速升高
- 传感器可以随便拆卸吗？内部灵敏度的调节螺丝可以自己调整吗？
  不建议读者这样做，因为一般的拆卸往往会导致传感器的失效，直接用就可以了