

## 无源蜂鸣器实验

### 1. 实验目的

- 掌握三极管开关电路的设计：NPN 型三极管开关电路和 PNP 型三极管开关电路。
- 掌握 STC15W4K32S4 系列单片机 PWM 控制无源蜂鸣器的驱动电路设计及程序控制注意事项。

### 2. 实验内容

- 编写程序实现操作触摸按键可控制蜂鸣器是否鸣叫。

### 3. 硬件电路设计

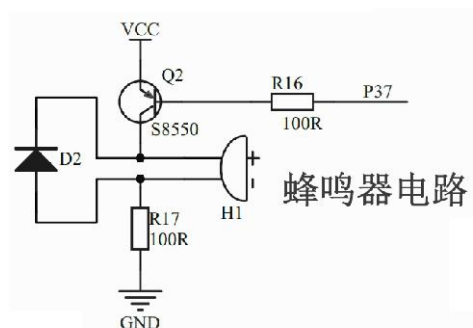
#### 3.1. 开发板无源蜂鸣器硬件电路

蜂鸣器是一种把警示电信号转换成人耳能够感知的电声转换装置，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。

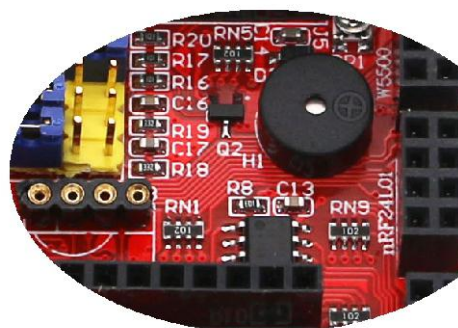
根据构造方式不同，蜂鸣器可分为：电磁式蜂鸣器和压电式蜂鸣器。根据封装不同，蜂鸣器可分为：插针式蜂鸣器和贴片式蜂鸣器。根据电流不同，蜂鸣器可分为：直流蜂鸣器和交流蜂鸣器。根据蜂鸣器自身有没有震荡源，蜂鸣器可分为：有源蜂鸣器（又叫自激式蜂鸣器）和无源蜂鸣器（又叫他激式蜂鸣器）。

有源蜂鸣器和无源蜂鸣器这里的“源”不是指电源，而是指震荡源。即有源蜂鸣器内部带震荡源，只要一通电就会“叫”。而无源蜂鸣器仅仅通电给直流信号是无法让无源蜂鸣器“叫”的，对无源蜂鸣器而言，必须要有外部震荡信号。

进取者 STC15 开发板上设计了 1 路无源蜂鸣器驱动电路，该无源蜂鸣器驱动电路的原理在后面给大家介绍，这里把无源蜂鸣器部分原理图和实物照给出。



1 无源蜂鸣器驱动电路原理图



2 无源蜂鸣器电路部分实物照

图 1：开发板无源蜂鸣器部分

✧ 1 路无源蜂鸣器电路占用的单片机的引脚如下表：

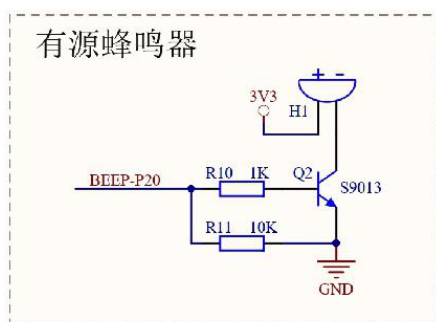
表 1：蜂鸣器电路引脚分配

BUZZER	功能描述	对应 IO 口	说明
BEEP	蜂鸣器驱动引脚	P3.7	独立 GPIO

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。

### 3.2. 无源蜂鸣器和有源蜂鸣器对比

首先，有源蜂鸣器和无源蜂鸣器的驱动电路是不一样的。STC8A8K64S4A12 开发板上设计了 1 路有源蜂鸣器驱动电路，该有源蜂鸣器部分原理图和实物照如下。



1 有源蜂鸣器驱动电路原理图

2 有源蜂鸣器电路部分实物照

图 2：STC8A8K64S4A12 开发板有源蜂鸣器部分

✧ 注：三极管 S8550 和 S9013 都是起开关作用，单片机 GPIO 连接三极管基极，GPIO 通过输出高低电平实现对三极管饱和导通和截止的控制。

不难发现无源蜂鸣器的驱动电路中在蜂鸣器两端并了一个二极管，该二极管称之为续流二极管（不可去掉）。该续流二极管作用：

- 1) 蜂鸣器本质上是一个感性元件，其电流不能瞬变，因此必须有一个续流二极管提供续流。
- 2) 如果没有续流二极管，在蜂鸣器两端会产生几十伏的尖峰电压，可能损坏三极管，并干扰整个电路系统的其他部分。

然后，驱动无源蜂鸣器时，需要单片机 GPIO 口输出方波信号才会有声音。驱动有源蜂鸣器的话，单片机 GPIO 输出高低电平即可控制蜂鸣器鸣叫与否。

驱动无源蜂鸣器，使用单片机 PWM 输出引脚更好，输出的方波信号一定频率范围内均可（频率不同，蜂鸣器发出的声音也有区别）。所以可知，有源蜂鸣器程序控制方便，但输出的声音不可控。另外，无源蜂鸣器价格上一般比有源蜂鸣器要便宜。

### 3.3. 三极管开关电路

三极管，全称为半导体三极管，也称双极型晶体管、晶体三极管，其有 3 种工作状态，分别是截止状态、放大状态和饱和导通状态。（每种状态的工作条件在此不做介绍）

利用三极管可工作于截止状态和饱和导通状态的电气特性，可通过控制三极管的基极电

压来控制三极管是导通还是断开，以此便可将三极管作为开关使用，形成三极管开关电路。

三极管开关电路的一大优势是可实现小电流控制大电流工作，具体而言是基极电流可以很小（几毫安）便可控制甚至几安培工作电流的负载工作。正是这样的优势，使得驱动能力有限的单片机 IO 口可以通过三极管开关电路高效控制较大功率的负载，比如蜂鸣器、继电器、液晶屏背光等。

三极管是在一块半导体基片上制作两个相距很近的 PN 结，两个 PN 结把整块半导体分成三部分，中间部分是基区，两侧部分是发射区和集电区，按两侧部分的发射区和集电区排列方式的不同可以将三极管分成 PNP 型三极管和 NPN 型三极管。常见 NPN 型三极管有 S9013、S8050 等，常见 PNP 型三极管有 S9012、S8550 等。

无论是 NPN 型三极管还是 PNP 型三极管，都有基极、集电极和发射极。基极、集电极和发射极对应着三极管实物的 3 个引脚。下图分 NPN 型和 PNP 型介绍三极管典型开关电路。

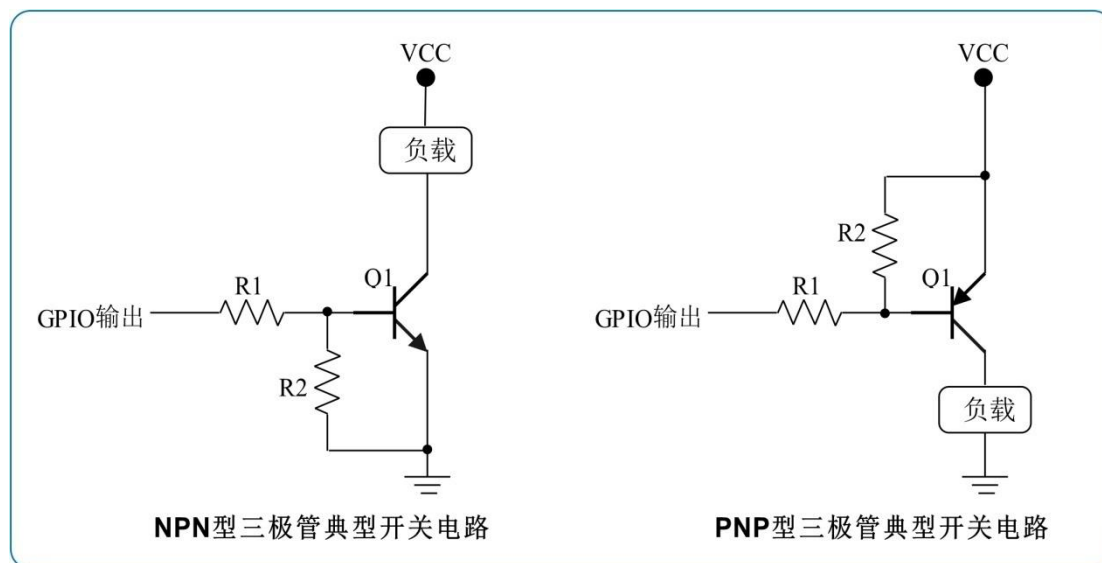


图 3：三极管典型开关电路

✧ 从上图可知，NPN 型三极管典型开关电路和 PNP 型三极管典型开关电路都用到了两个电阻，两个电阻的作用如下所述。

1、三极管典型开关电路中 R1 的作用是保护单片机 GPIO 口和三极管基极。针对 NPN 型三极管 S9013 举例，如果没有电阻 R1，当单片机 GPIO 口输出高电平时，三极管基极和发射极之间有有效的压差使三极管处于导通状态，一旦三极管饱和导通，基极和发射极间的电压差是 0.7V，发射极接地，那基极电压应是 0.7V 左右，而单片机 GPIO 口又输出的是高电平 3.3V，那么对无论单片机 GPIO 口还是三极管基极都会造成不可预知的损坏，故选择在单片机 GPIO 口和三极管基极之间加一个限流电阻 R1，至于 R1 的阻值如何选择将在接下来的对三极管开关电路的分析中讲解。

2、三极管典型开关电路中 R2 的作用是在单片机 GPIO 口呈高阻态时使晶体管可靠截止，避免出现误动作。如果没有 R2 电阻，则单片机 GPIO 呈高阻态时输出的电平是不确定的，那可能会让三极管导通或者截止，三极管截止的话，负载不工作不会有什么后果。而如果三

极管导通，控制了负载工作，则会造成不可预知的后果。故加上 R2 电阻使得三极管基极和发射极电压一致，三极管会在单片机 GPIO 口呈高阻态时始终处于截止状态。电阻 R2 还有一个作用是用于泄漏电流，即使三极管处于截止状态，也不能保证三极管基极就没有电压（尤其集成芯片上断电时），而有了 R2 电阻可以将漏电流通过 R2 流入到地，保护了三极管和整个电路。

### 3.3.1. 分析 NPN 型三极管典型开关电路

如下图所示，以硅半导体 NPN 型三极管为例介绍单片机 GPIO 口输出高电平和低电平时如何控制三极管饱和导通和截止，并分析三极管典型开关电路中电阻值的选择。（假设单片机是 3.3V 的单片机，VCC 供电也是 3.3V）

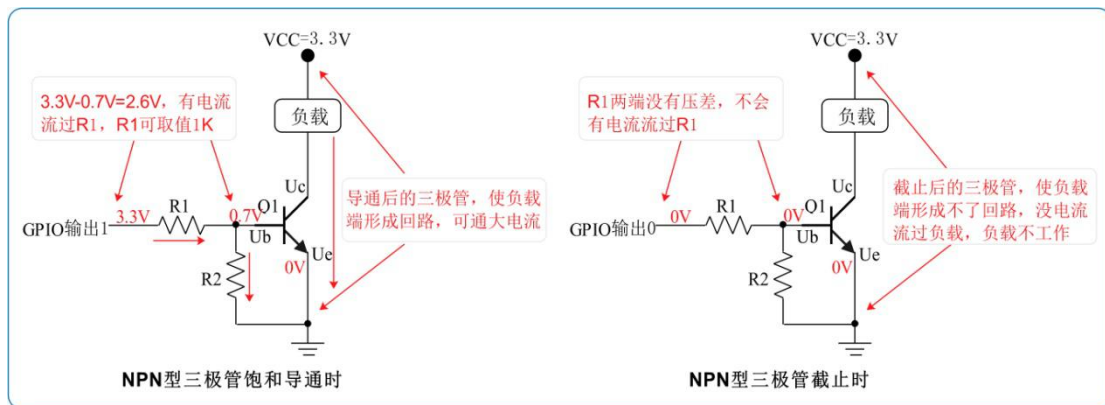


图 4：三极管典型开关电路

1、当单片机 GPIO 口由软件控制输出逻辑“1”，针对硬件电路 GPIO 口输出的就是高电平 3.3V。这个高电平会使三极管基极电压远大于三极管发射极电压 0V（对 R2 取值有要求），从而使三极管工作在饱和导通状态，而一旦三极管进入饱和导通状态，由三极管工作原理及选择的是硅半导体三极管可知三极管基极和发射极电压差  $U_{be} \approx 0.7V$ ，三极管发射极接地故可知三极管基极电压约 0.7V，如上图 3 标注。R1 取值会是一个范围，三极管有个重要的参数是电流放大系数  $\beta$ ， $\beta$  除了和所选择的三极管型号有关，还受温度等影响。在此取值 R1 为 1K，那么可计算出流过 R1 的电流约是 2.6mA，这个驱动电流是单片机 GPIO 引脚可以提供的。如果单片机是 5V 供电的，R1 是否选择 1K，用户可综合考虑计算取值。

之前有说过 R2 的作用，但没有说明 R2 的阻值是如何取的。R2 的取值至少要满足其与基极限流电阻 R1 分压后能够满足三极管的临界饱和，实际选择时会大大高于这个最小值，通常外界干扰越小、负载越重，允许 R2 的取值就越大，一般采用 10K 量级的电阻。在此仅分析下 R2 取值过小为什么不行，假设 R2 取值  $100\Omega$ （R1 取值按照上面的 1K 取值），三极管刚开始是截止的，电流全部从 R2 流到地，那么易算出 R2 与 R1 分压后基极端的电压是 0.3V，这个电压不能使三极管进入饱和导通状态。另一个因素，即使三极管处于截止状态，R2 阻值过小都会导致可能产生的漏电流过大，所以综合考虑取值 R2 为 10K，当然这个值不是唯一的。

2、当单片机 GPIO 口由软件控制输出逻辑“0”，针对硬件电路 GPIO 口输出的就是低电平 0V。这个低电平毫无疑问不会使三极管基极有超过 0.7V 的电压，那么三极管会处于截



止状态。三极管集电极和发射极之间不会有电流流过，负载端无法形成回路，负载不会工作。上图 3 中有直观分析 NPN 型三极管截止时的工作情况。

### 3.3.2. 分析 PNP 型三极管典型开关电路

如下图所示，以硅半导体 PNP 型三极管为例介绍单片机 GPIO 口输出高电平和低电平时如何控制三极管饱和和导通和截止，并分析三极管典型开关电路中电阻值的选择。（假设单片机是 3.3V 的单片机，VCC 供电也是 3.3V）

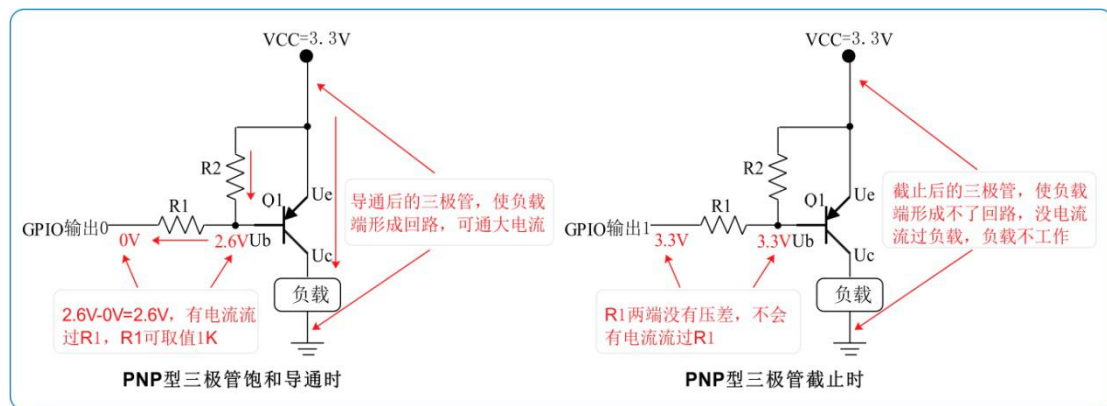


图 5：三极管典型开关电路

1、当单片机 GPIO 口由软件控制输出逻辑“0”，针对硬件电路 GPIO 口输出的就是低电平 0V。这个低电平会使三极管基极电压远低于三极管发射极电压 3.3V（对 R2 取值有要求），从而使三极管工作在饱和和导通状态，而一旦三极管进入饱和和导通状态，由三极管工作原理及选择的是硅半导体三极管可知三极管基极和发射极电压差  $U_{be} \approx 0.7V$ ，三极管发射极接 3.3V 故可知三极管基极电压约 2.6V，如上图 4 标注。此时负载端形成回路，负载工作。

关于 R1 和 R2 阻值的取值与 NPN 型三极管典型开关电路中已有介绍，在此不再赘述。

2、当单片机 GPIO 口由软件控制输出逻辑“1”，针对硬件电路 GPIO 口输出的就是高电平 3.3V。这个高电平不会使得三极管基极和发射极之间有超过 0.7V 的电压差，那么三极管会处于截止状态。三极管发射极和集电极之间不会有电流流过，负载端无法形成回路，负载不会工作。上图 4 中有直观分析 PNP 型三极管截止时的工作情况。

### 3.3.3. 对比 NPN 型典型开关电路和 PNP 型典型开关电路

无论 NPN 型典型开关电路还是 PNP 型典型开关电路本质上是一样的，只是选择的三极管型号及单片机 GPIO 口输出高低信号不同而已。大多数情况下都是可以相互替换的，但这是建立于单片机 GPIO 口输出的高电平值与负载供电的电平值是一样的。如果负载的供电是 5V，而单片机供电是 3.3V，那么情况会有所不同。

下面将分析 NPN 型典型开关电路和 PNP 型典型开关电路在负载是 5V 供电单片机是 3.3V 供电下，单片机 GPIO 能否控制三极管有效进入截止状态和饱和和导通状态。

1、NPN 型三极管开关电路在单片机 GPIO 输出高低电平时：

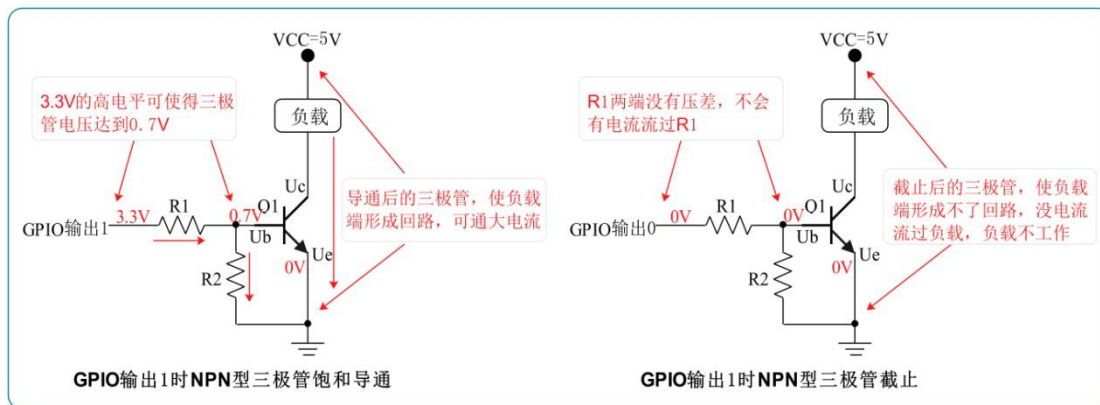


图 6：三极管典型开关电路

由上图分析可知，虽然负载的供电是 5V，但单片机 GPIO 口输出高电平 3.3V 可以使 NPN 三极管导通，单片机 GPIO 口输出低电平 0V 可以使 NPN 三极管截止。因为 NPN 三极管发射极连接的是 GND，NPN 三极管能否工作取决于三极管基极和发射极的电压差，这样单片机 GPIO 口输出高低电平均可有效控制 NPN 三极管导通或截止。

## 2、PNP 型三极管开关电路在单片机 GPIO 输出高低电平时：

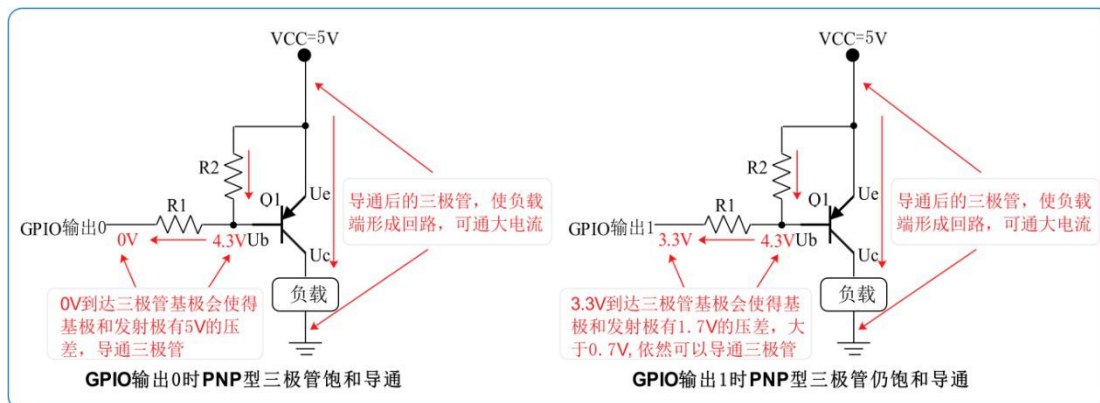


图 7：三极管典型开关电路

由上图分析可知，由于负载的供电是 5V，而 PNP 三极管发射极连接的是 5V，这样单片机 GPIO 口输出高电平 3.3V 或者低电平 0V 均可使 PNP 三极管导通，此种情况，3.3V 供电的单片机 GPIO 口不能控制 PNP 三极管截止。

## 4. 软件设计

### 4.1. 触摸按键输入检测实验

✧ 注：本节的实验源码是在“实验 2-9-1: PWM2 和 PWM3 呼吸灯实验”的基础上修改。本节对应的实验源码是：“实验 2-17: 蜂鸣器实验”。

#### 4.1.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 2: 实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	beep	.c	无源蜂鸣器驱动有关的用户自定义函数。
2	delay	.c	包含用户自定义延时函数。

#### 4.1.2. 头文件引用和路径设置

##### ■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "beep.h"
```

##### ■ 需要包含的头文件路径

本例需要包含的头文件路径如下表:

表 3: 头文件包含路径

序号	路径	描述
1	..\ Source	beep.h 和 delay.h 头文件在该路径, 所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径, 所以要包含。

MDK 中点击魔术棒, 打开工程配置窗口, 按照下图所示添加头文件包含路径。

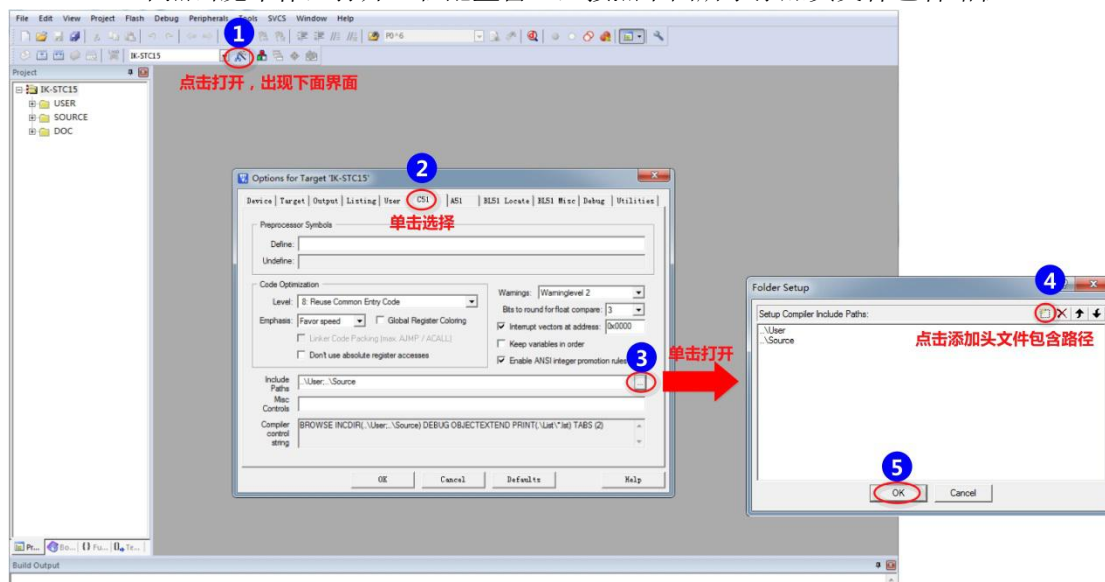


图 8: 添加头文件包含路径

#### 4.1.3. 编写代码

首先, 在 beep.c 文件中对蜂鸣器用 PWM 口进行初始化 PWM2\_P37\_Configuration, 代码如下。

## 程序清单：PWM2 初始化函数

```

1.  /*****
2.  功能描述：对 PWM2（P3.7 引脚）进行初始化
3.  入口参数：无
4.  返回值：无
5.  *****/
6.  void PWM2_P37_Configuration(void)
7.  {
8.      PWMCFG &= 0xBF;      //将 CBTADC 位置 0，即 PWM 计数器归零时不触发 ADC 转换
9.      PWMIF &= 0xBF;      //将 CBIF 位置 0，PWM 计数器归零中断标志位，需软件清零
10.
11.     P_SW2 |= 0x80;      //将 EAXSFR 位置 1，以访问 PWM 在扩展 RAM 区的特殊功能寄存器
12.     //对 PWM2 的初始化部分
13.     PWM2CR &= 0xF7;      //将 PWM2_PS 位置 0，选择 PWM2 的输出引脚是 P3.7
14.     PWMCR |= 0x01;      //将 ENC2O 位置 1，PWM2 的端口为 PWM 输出口，受 PWM 波形发生器
        控制
15.     PWMCFG &= 0xFE;      //将 C2INI 位置 0，设置 PWM2 输出端口的初始电平为低电平
16.     PWMIF &= 0xFE;      //将 C2IF 位置 0，PWM2 中断标志位，需软件清零
17.     PWM2CR |= 0x04;      //将 EPWM2I 位置 1，使能 PWM2 中断
18.     PWM2CR &= 0xFD;      //将 EC2T2SI 位置 0，关闭 T2 翻转时中断
19.     PWM2CR &= 0xFE;      //将 EC2T1SI 位置 0，关闭 T1 翻转时中断
20.     //对 PWM2 和 PWM3 翻转计数器赋初值
21.     PWM2T1 = 0x007D;      //赋值 PWM2 第一次翻转计数器值
22.     PWM2T2 = 0x00FA;      //赋值 PWM2 第二次翻转计数器值
23.
24.     //对 PWM 波形发生器时钟源进行初始化
25.     PWMCKS |= 0x10;      //将 SELT2 位置 1，PWM 时钟源为定时器 2 溢出脉冲
26.     PWMCH = 0x00FA;      //PWM 计数器赋值（同时对 PWMCH 和 PWMCL 进行了赋值）
27.     AUXR |= 0x04;      //定时器 2 时钟为 Fosc,即 1T
28.     T2L = 0xE0;      //设定定时初值
29.     T2H = 0xFE;      //设定定时初值
30.     AUXR |= 0x10;      //启动定时器 2
31.     P_SW2 &= 0x7F;      //将 EAXSFR 位置 0，恢复访问 XRAM
32.
33.     //PWM 外部异常控制寄存器的操作
34.     PWMFDCR &= 0xDF;      //将 ENFD 位置 0，关闭 PWM 外部异常检测功能
35.     PWMFDCR &= 0xF7;      //将 ENDI 位置 0，关闭 PWM 异常检测中断
36.     PWMFDCR &= 0xFB;      //将 FDCMP 位置 0，比较器与 PWM 无关
37.     PWMFDCR &= 0xFD;      //将 FDIO 位置 0，P2.4 的状态与 PWM 无关
38.     PWMFDCR &= 0xFE;      //将 FDIF 位置 0，PWM 异常检测中断标志位，需软件清零
39.
40.     IP2 |= 0x40;      //将 PPWM 位置 1，使能 PWM 中断为最高优先级中断
41.     //使能 PWM 波形发生器

```



```

42.  PWMCR |= 0x80;           //将 ENPWM 位置 1，使能 PWM 波形发生器，PWM 计数器开始计数
43.  PWMCR &= 0xBF;          //将 ECBI 位置 0，禁止 PWM 计数器归零中断
44.  }

```

然后，编写控制无源蜂鸣器启动函数和停止函数，实际就是对 PWM 外设进行控制，代码如下。

#### 程序清单：开启蜂鸣器函数

```

1.  /*****
2.  * 描 述: 开启蜂鸣器
3.  * 入 参: 无
4.  * 返回值: 无
5.  *****/
6.  void BEEP_on(void)
7.  {
8.      PWM2_P37_Configuration();
9.      EA = 1;           //允许总中断
10. }

```

#### 程序清单：关闭蜂鸣器函数

```

1.  /*****
2.  * 描 述: 关闭蜂鸣器
3.  * 入 参: 无
4.  * 返回值: 无
5.  *****/
6.  void BEEP_off(void)
7.  {
8.      EA = 0;           //关闭总中断
9.      PWMCR &= 0x7F;    //将 ENPWM 位置 0，禁止 PWM 波形发生器，PWM 计数器停止计数
10. }

```

最后，主函数中先调用蜂鸣器用 PWM 口的初始化函数，然后检测触摸按键状态实现对无源蜂鸣器的控制。具体代码如下。

#### 代码清单：主函数

```

1.  int main()
2.  {
3.      ///////////////////////////////////////////////////
4.      //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.      //    高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.      //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.      //    P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.      ///////////////////////////////////////////////////
9.      P2M1 &= 0xDF; P2M0 &= 0xDF; //设置 P2.5 为准双向口
10.     P3M1 &= 0x7F; P3M0 &= 0x7F; //设置 P3.7 为准双向口
11.
12.     while(1)

```

```

13. {
14.   if(TOUCH_KEY == 1)    //检测触摸按键 S4 对应引脚 P2.5 是否是低电平 (有手指触摸按键 S4 触摸
                           区域, 引脚为高电平)
15.   {
16.     delay_ms(10);       //软件延时 10ms, 如果延时后按键 S4 的电平依然没有变化, 说明按键确实被
                           有效操作, 简称按键消抖
17.     if(TOUCH_KEY == 1)  //检测触摸按键 S4 对应引脚 P2.5 是否依然是高电平
18.     {
19.       BEEP_on();        //开启蜂鸣器
20.       while(TOUCH_KEY == 1) //等待按键 S4 释放, 即如果 P2.5 一直为高电平, 会一直执行空命令
21.       {
22.         ;                //条件 TOUCH_KEY == 1 成立, 会执行这个空命令
23.       }
24.       BEEP_off();       //关闭蜂鸣器
25.     }
26.   }
27. }
28. }

```

#### 4.1.4. 硬件连接

本实验需要用到触摸按键，所以触摸按键用 IO 口要选择，实验连接图如下。

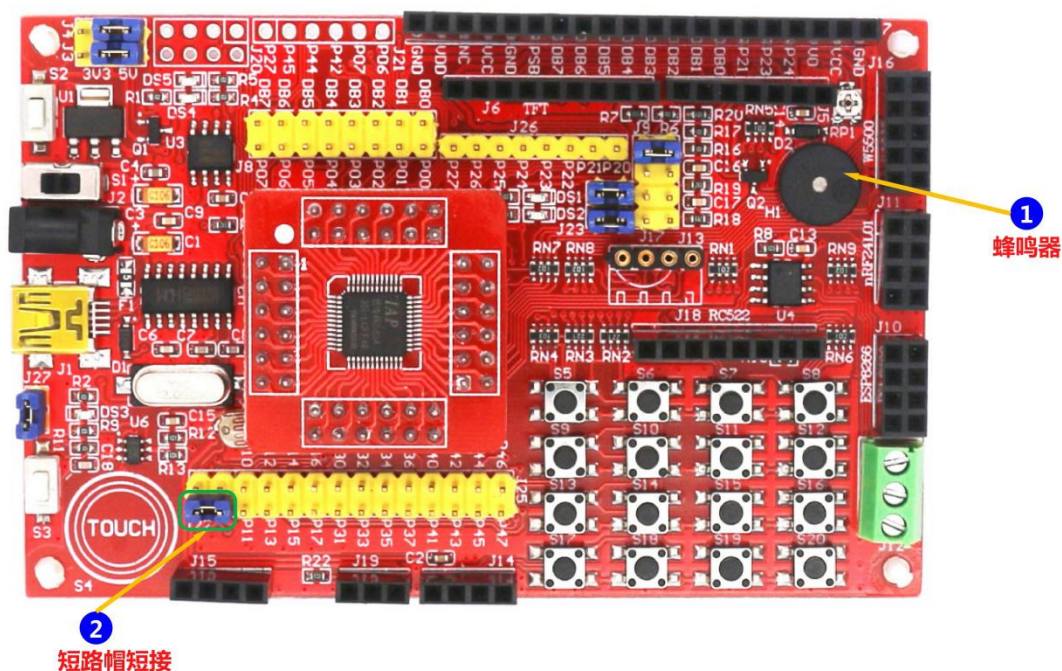


图 9：无源蜂鸣器实验连接图

#### 4.1.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\1 - 基础实验程序\”目录下的压缩文件“实验 2-17:

蜂鸣器实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。

2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“...\BEEP\projec”目录下的工程“BEEP.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“BEEP.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，把手指放在触摸按键 S4 识别区域蜂鸣器鸣叫，把手拿开，蜂鸣器停止鸣叫。