

传感技术实验报告

孙英涵

2020 年 5 月 3 日

目录

第一部分 基础实验	1
实验一 Hello Arduino	2
1.1 实验用品	2
1.2 实验步骤	2
1.3 实验程序	2
1.4 程序解读	2
1.5 实验结果	3
实验二 LED 流水灯实验	4
2.1 实验目的	4
2.2 LED	4
2.3 实验用品	4
2.4 实验步骤	5
2.5 接线图	5
2.6 实验程序	5
2.7 实验结果	7
实验三 RGB 贴片模块呼吸灯实验	8
3.1 脉冲宽度调制	8
3.2 RGB 色彩模式	9
3.3 实验用品	9
3.4 实验步骤	9
3.5 接线图	10
3.6 实验程序	10
3.7 实验结果	11
实验四 按键模块实验	12
4.1 实验目的	12
4.2 按键模块	12
4.3 实验原理	12
4.4 实验用品	12
4.5 实验步骤	13
4.6 接线图	13
4.7 实验程序	13

4.8 实验结果	14
实验五 矩阵键盘实验	16
5.1 实验目的	16
5.2 矩阵键盘	16
5.3 实验用品	17
5.4 实验步骤	17
5.5 接线图	17
5.6 实验程序	18
5.7 实验结果	19
实验六 PWM 调节 LED 灯亮度实验	20
6.1 实验目的	20
6.2 电位器	20
6.3 实验原理	21
6.4 实验用品	21
6.5 实验步骤	21
6.6 接线图	21
6.7 实验程序	22
6.8 实验结果	22
实验七 A/D 模数转换实验	24
7.1 实验目的	24
7.2 A/D 转换	24
7.3 实验原理	24
7.4 实验用品	24
7.5 实验步骤	25
7.6 接线图	25
7.7 实验程序	25
7.8 实验结果	26
实验八 蜂鸣器模块实验	27
8.1 实验目的	27
8.2 蜂鸣器	27
8.3 实验用品	28
8.4 实验步骤	28
8.5 接线图	28
8.6 实验程序	28
8.6.1 有源蜂鸣器实验程序	28
8.6.2 无源蜂鸣器实验程序	29
8.7 实验结果	30

目录	iii
第二部分 传感器模块实验	31
实验所用传感器的分类	32
实验九 四位七段数码管模块显示实验	33
9.1 实验目的	33
9.2 七段数码管	33
9.2.1 数码管的结构	33
9.2.2 四位七段数码管模块	34
9.2.3 四位数码管的动态扫描原理	34
9.3 实验用品	35
9.4 实验步骤	35
9.5 接线图	35
9.6 实验程序	35
9.6.1 显示“1234”	35
9.6.2 计时功能实现	36
9.7 实验结果	38
实验十 MAX7219 驱动单个 8×8 点阵模块	39
10.1 实验目的	39
10.2 MAX 7219	39
10.3 实验用品	40
10.4 接线图	40
10.5 实验程序	40
10.6 实验结果	41
实验十一 串行 LCD 动态显示屏实验	43
11.1 实验目的	43
11.2 LCD 显示模块	43
11.3 实验用品	43
11.4 实验步骤	43
11.5 接线图	44
11.6 实验程序	44
11.7 实验结果	45
实验十二 游戏操纵五向按键模块实验	46
12.1 实验目的	46
12.2 游戏操纵五向按键模块	46
12.3 实验用品	46
12.4 实验步骤	47
12.5 接线图	47
12.6 实验程序	47
12.7 实验结果	49

实验十三 魔术光环模块实验	50
13.1 实验目的	50
13.2 魔术光环模块	50
13.3 实验用品	50
13.4 实验步骤	50
13.5 接线图	51
13.6 实验程序	51
13.7 实验结果	52
实验十四 灰度传感器模块实验	53
14.1 实验目的	53
14.2 灰度传感器	53
14.3 主要工作参数	53
14.4 实验原理	54
14.5 实验用品	54
14.6 实验步骤	54
14.7 接线图	54
14.8 实验程序	55
14.9 实验结果	55
实验十五 触摸开关模块实验	57
15.1 实验目的	57
15.2 触摸开关模块	57
15.3 主要工作参数	57
15.4 实验原理	58
15.5 实验用品	58
15.6 实验步骤	58
15.7 接线图	58
15.8 实验程序	59
15.9 实验结果	59
实验十六 光敏电阻模块实验	61
16.1 实验目的	61
16.2 光敏电阻	61
16.3 主要工作参数	61
16.4 实验原理	62
16.5 实验用品	62
16.6 实验步骤	62
16.7 接线图	62
16.8 实验程序	63
16.9 实验结果	64

实验十七 火焰传感器模块实验	65
17.1 实验目的	65
17.2 火焰传感器	65
17.3 主要工作参数	65
17.4 实验原理	66
17.5 实验用品	66
17.6 实验步骤	66
17.7 接线图	66
17.8 实验程序	67
17.9 实验结果	68
实验十八 LM35 温度传感器实验	69
18.1 实验目的	69
18.2 LM35 温度传感器	69
18.3 主要工作参数	69
18.4 实验原理	70
18.5 实验用品	70
18.6 实验步骤	70
18.7 接线图	70
18.8 实验程序	71
18.9 实验结果	71
实验十九 湿湿度传感器模块实验	72
19.1 实验目的	72
19.2 DHT11 湿湿度传感器	72
19.3 主要工作参数	72
19.4 实验步骤	73
19.5 实验用品	73
19.6 接线图	73
19.7 实验程序	74
19.8 实验结果	74
实验二十 红外循迹模块实验	75
20.1 实验目的	75
20.2 红外循迹模块	75
20.3 主要工作参数	75
20.4 实验原理	76
20.5 实验用品	76
20.6 实验步骤	76
20.7 接线图	76
20.8 实验程序	77
20.9 实验结果	77

实验二十一 红外避障模块实验	79
21.1 实验目的	79
21.2 红外避障模块	79
21.3 主要工作参数	79
21.4 实验原理	80
21.5 实验用品	80
21.6 实验步骤	80
21.7 接线图	80
21.8 实验程序	81
21.9 实验结果	81
实验二十二 热敏电阻模块实验	83
22.1 实验目的	83
22.2 热敏电阻	83
22.3 主要工作参数	83
22.4 实验原理	83
22.5 实验用品	84
22.6 实验步骤	84
22.7 接线图	84
22.8 实验程序	85
22.9 实验结果	85
实验二十三 水深传感器模块实验	87
23.1 实验目的	87
23.2 水深传感器	87
23.3 主要工作参数	87
23.4 实验用品	88
23.5 实验步骤	88
23.6 接线图	88
23.7 实验程序	89
23.8 实验结果	89
实验二十四 雨滴传感器模块实验	90
24.1 实验目的	90
24.2 雨滴传感器	90
24.3 主要工作参数	90
24.4 实验原理	91
24.5 实验用品	91
24.6 实验步骤	91
24.7 接线图	91
24.8 实验程序	92
24.9 实验结果	92

实验二十五 土壤湿度传感器模块实验	94
25.1 实验目的	94
25.2 土壤湿度传感器	94
25.3 主要工作参数	94
25.4 实验用品	95
25.5 实验步骤	95
25.6 接线图	95
25.7 实验程序	96
25.8 实验结果	96
实验二十六 霍尔开关模块实验	98
26.1 实验目的	98
26.2 霍尔开关	98
26.3 主要工作参数	99
26.4 实验原理	99
26.5 实验用品	99
26.6 实验步骤	99
26.7 接线图	99
26.8 实验程序	100
26.9 实验结果	101
实验二十七 倾斜开关模块实验	102
27.1 实验目的	102
27.2 倾斜开关	102
27.3 主要工作参数	102
27.4 实验原理	103
27.5 实验用品	103
27.6 实验步骤	103
27.7 接线图	103
27.8 实验程序	104
27.9 实验结果	105
实验二十八 磁簧开关模块实验	106
28.1 磁簧开关	106
28.2 主要工作参数	106
28.3 实验原理	107
28.4 实验用品	107
28.5 实验步骤	107
28.6 接线图	107
28.7 实验程序	108
28.8 实验结果	108

实验二十九 声音传感器模块实验	110
29.1 实验目的	110
29.2 声音传感器	110
29.3 主要工作参数	110
29.4 实验用品	110
29.5 实验步骤	111
29.6 接线图	111
29.7 实验程序	111
29.8 实验结果	112
实验三十 手指侦测心跳模块实验	113
30.1 实验目的	113
30.2 手指侦测心跳模块	113
30.3 主要工作参数	113
30.4 实验原理	114
30.5 实验用品	114
30.6 实验步骤	114
30.7 接线图	114
30.8 实验程序	115
30.9 实验结果	115
实验三十一 光电传感器模块实验	116
31.1 实验目的	116
31.2 光电传感器	116
31.3 主要工作参数	116
31.4 实验原理	116
31.5 实验用品	117
31.6 实验步骤	117
31.7 接线图	117
31.8 实验程序	118
31.9 实验结果	118
实验三十二 烟雾气体传感器模块实验	120
32.1 实验目的	120
32.2 MQ-2 烟雾气体传感器	120
32.3 主要工作参数	120
32.4 实验原理	121
32.5 实验用品	121
32.6 实验步骤	121
32.7 接线图	121
32.8 实验程序	122
32.9 实验结果	122

第一部分

基础实验

实验一 Hello Arduino

1.1 实验用品

Arduino UNO 开发板、USB 数据线

1.2 实验步骤

- (1) 打开 Arduino IDE，输入程序；
- (2) 在工具栏选择适用的端口以及编程器；
- (3) 点击上传按钮，当左下角提示上传成功时，说明程序已经成功烧录；
- (4) 打开串口监视器，查看结果。

1.3 实验程序

```
1 // HelloArduino.ino
2 // Date: 2020/04/17
3
4 void setup()
5 {
6     Serial.begin(9600);
7 }
8
9 void loop()
10 {
11     Serial.println("Hello Arduino!");
12     delay(1000);
13 }
```

1.4 程序解读

```
1 Serial.begin(9600);
```

初始化串口通信，并将波特率设为9600。

```
1 Serial.println("Hello Arduino!");
```

从串口输出数据，跟随一个回车 (ASCII 13, 或 'r') 和一个换行符 (ASCII 10, 或 'n')。

```
1 delay(1000);
```

delay() 函数是 Arduino 中的一个延时函数，参数的单位为毫秒，例如 delay(1000) 就是延时1秒中。

1.5 实验结果

打开串口监视器，并将波特率设置为9600，我们即可看到在串口端不断打印出的 ”Hello Arduino!”，如图1.1所示。

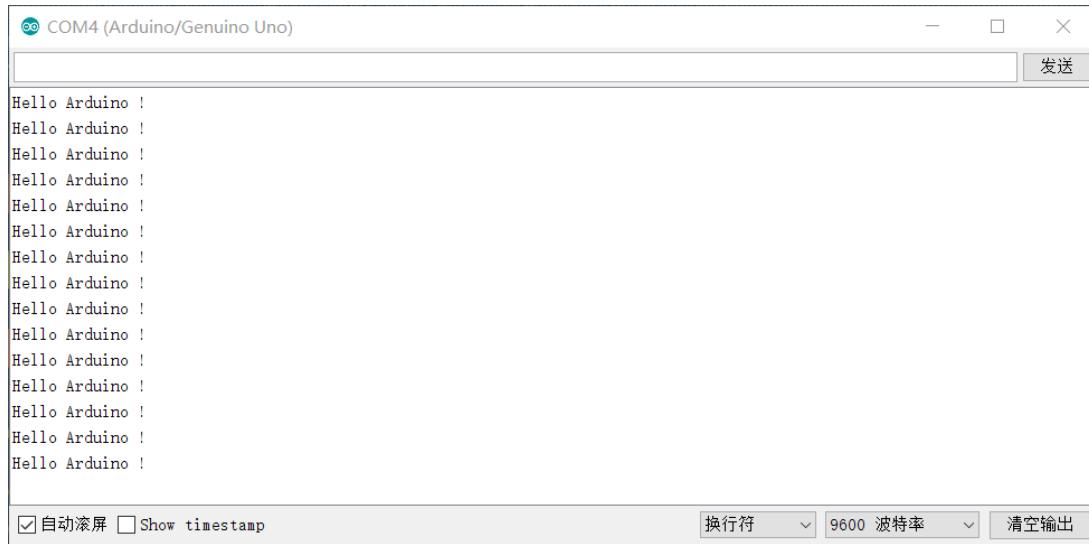


图 1.1: 串口监视器中不断打印出的 “Hello Arduino”

这里将波特率设置为9600的原因是为了和程序中的设置一样，否则打印的内容会乱码。如图1.2所示，这里我们将波特率修改为38400，可以看到它无法正常打印出 ”Hello Arduino!”。

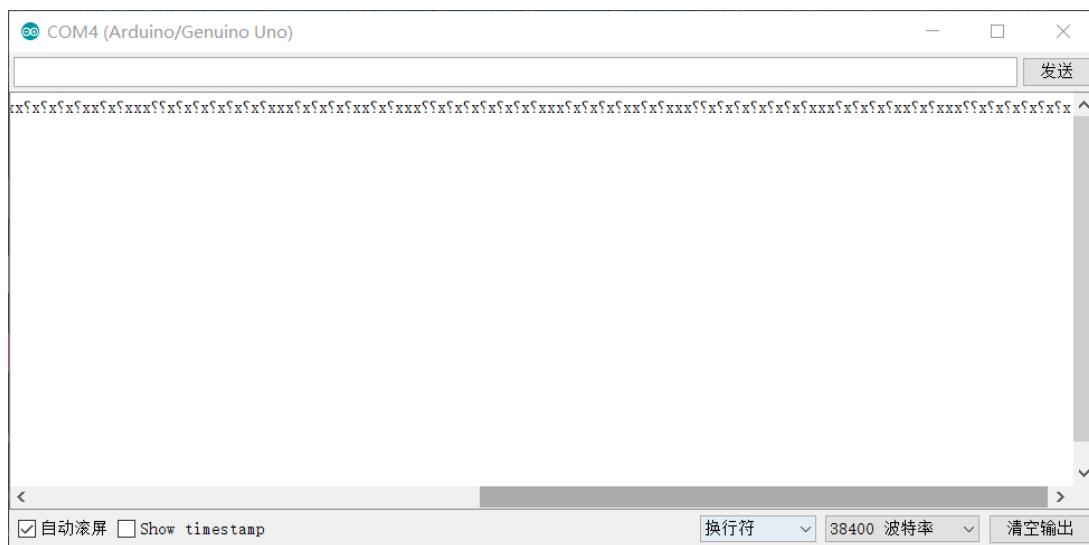


图 1.2: 当波特率不一致时会乱码

实验二 LED 流水灯实验

2.1 实验目的

流水灯即让灯光像流水一样亮起来。在本实验中，我们希望 LED 灯可以像流水一样从左至右依次亮起，然后从右向左依次熄灭。

2.2 LED

LED (Light Emitting Diode)，又名发光二极管，是一种能够将电能转化为可见光的固态的半导体器件，它可以直接把电转化为光。LED 的心脏是一个半导体的晶片，晶片的一端附在一个支架上，一端是负极，另一端连接电源的正极，使整个晶片被环氧树脂封装起来。



图 2.1: LED 示意图，其中长脚为正脚，短脚为负角

LED 的反向击穿电压为5V。其正伏安特性曲线太陡，必须与限流电阻串联，以便在使用时控制流过管道的电流。限流电阻 R 可通过以下公式获得：

$$R = \frac{E - V_F}{I}$$

式中 E 代表电源电压， V_F 代表 LED 的正向电压降， I 为 LED 的一般工作电流。LED 的工作电压一般为 1.5 - 2.0V，工作电流通常为 10 - 20mA。因此在 5v 的数字逻辑电路中，我们可以使用 220Ω 电阻作为限流电阻。

2.3 实验用品

Arduino UNO 开发板、USB 数据线、面包板、LED 灯及 220Ω 电阻各8个、若干杜邦线

2.4 实验步骤

- (1) 按照如图2.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 观察 LED 灯的状态。

2.5 接线图

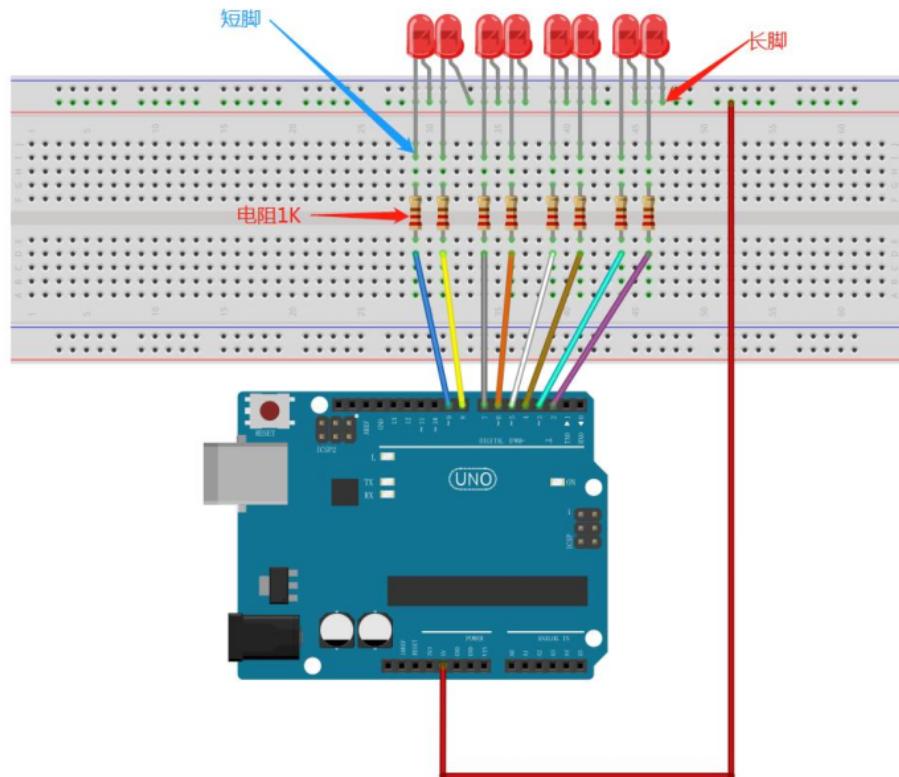


图 2.2: 实验接线图

2.6 实验程序

```
1 // LED_Running_Lights.ino
2 // Date: 2020/04/17
3
4 int led_array[8] = { 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 } ;
5 int flash_speed = 500 ;
6
7 /* flash led from left to right one by one */
8 void led_flash(void)
```

```
9  {
10     for(int i = 0; i < 8; i++)
11     {
12         digitalWrite(led_array[i], LOW);
13         delay(flash_speed);
14         digitalWrite(led_array[i], HIGH);
15     }
16 }
17
18 /* turn on all led form left to right */
19 void led_turn_on(void)
20 {
21     for(int i = 0; i < 8; i++)
22     {
23         digitalWrite(led_array[i], LOW);
24         delay(flash_speed);
25     }
26 }
27
28 /* turn off all led */
29 void led_turn_off(void)
30 {
31     for(int i = 0; i < 8; i++ )
32     {
33         digitalWrite(led_array[i], HIGH);
34         delay(flash_speed);
35     }
36 }
37
38 void setup()
39 {
40     Serial.begin(115200);
41     for(int i = 0; i < 8; i++)
42     {
43         pinMode(led_array[i], OUTPUT);
44         // set led control pin defalut HIGH turn off all LED
45         digitalWrite(led_array[i], HIGH);
46     }
47 }
48
49 void loop()
50 {
51     Serial.println("start flash led!");
52     led_flash();
53     led_turn_off();
54     led_turn_on();
55 }
```

2.7 实验结果

如图2.3所示，LED灯如期望的那样像流水一样从左至右依次亮起，然后从右向左依次熄灭，如此循环往复。

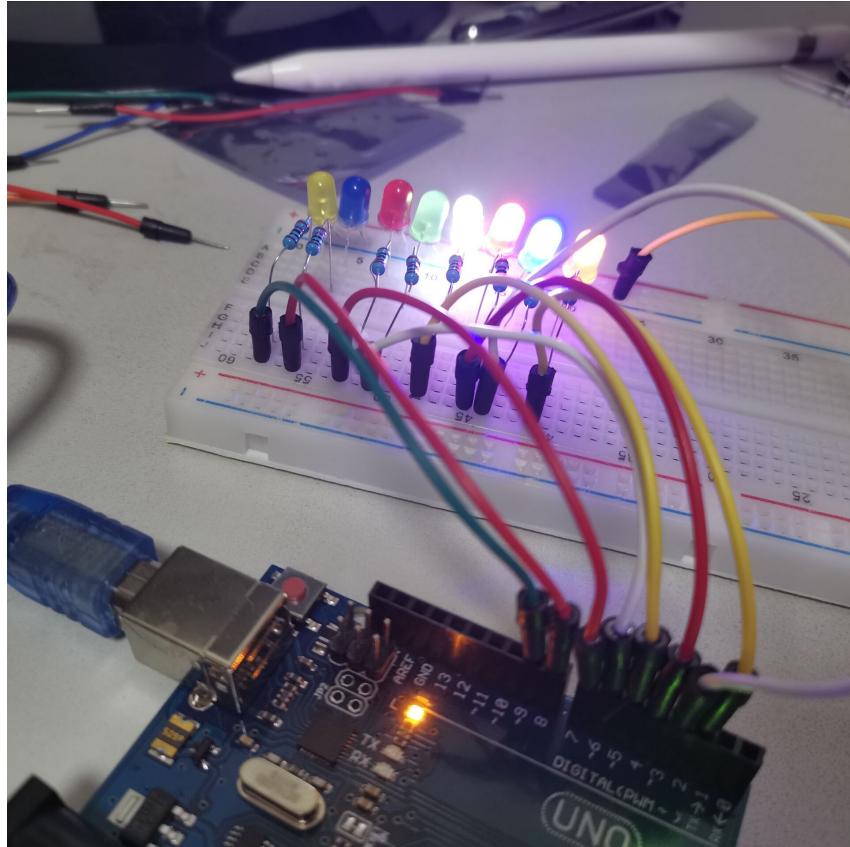


图 2.3: LED 灯流水灯正在工作

实验三 RGB 贴片模块呼吸灯实验

3.1 脉冲宽度调制

脉冲宽度调制 (Pulse Width Modulation, PWM)，简称脉宽调制，是将模拟信号变换为脉冲的一种技术，一般变换后脉冲的周期固定，但脉冲的工作周期会依模拟信号的大小而改变。

在模拟电路中，模拟信号的值可以连续进行变化，在时间和值的幅度上都几乎没有限制，基本上可以取任何实数值，输入与输出也呈线性变化。所以在模拟电路中，电压和电流可直接用来进行控制对象，例如家用电器设备中的音量开关控制、采用卤素灯泡灯具的亮度控制等等。但模拟电路有诸多的问题：例如控制信号容易随时间漂移，难以调节；功耗大；易受噪声和环境干扰等等。

与模拟电路不同，数字电路是在预先确定的范围内取值，在任何时刻，其输出只可能为 ON 和 OFF 两种状态，所以电压或电流会通/断方式的重复脉冲序列加载到模拟负载。PWM 技术是一种对模拟信号电平的数字编码方法，通过使用高分辨率计数器（调制频率）调制方波的占空比，从而实现对一个模拟信号的电平进行编码。其最大的优点是从处理器到被控对象之间的所有信号都是数字形式的，无需再进行数模转换过程；而且对噪声的抗干扰能力也大大增强（噪声只有在强到足以将逻辑值改变时，才可能对数字信号产生实质的影响），这也是 PWM 在通讯等信号传输行业得到大量应用的主要原因。

模拟信号能否使用 PWM 进行编码调制，仅依赖带宽，这即意味着只要有足够的带宽，任何模拟信号值均可以采用 PWM 技术进行调制编码，一般而言，负载需要的调制频率要高于 10Hz，在实际应用中，频率约在 1kHz 到 200kHz 之间。在信号接收端，需将信号解调还原为模拟信号，目前在很多微型控制器 (MCU) 内部都包含有 PWM 控制器模块。

PWM 的相关术语：

- (1) **On-Time (导通时间)**: 时间信号的持续时间较长。
- (2) **Off-Time (关断时间)**: 时间信号的持续时间较短。
- (3) **Period (周期)**: 表示为 PWM 信号的导通时间和关断时间的总和。
- (4) **Duty Cycle (占空比)**: 它表示为在 PWM 信号周期内保持导通的时间信号的百分比。
- (5) **周期**: 如图3.1所示， T_{on} 表示导通时间， T_{off} 表示信号的关断时间。周期是导通和关断时间的总和，并按照以下公式计算：

$$T_{total} = T_{on} + T_{off}$$

(6) 占空比：占空比用于计算为一段时间的导通时间。使用上面计算的周期，占空比计算为：

$$D = \frac{T_{on}}{T_{on} + T_{off}} = \frac{T_{on}}{T_{total}}$$

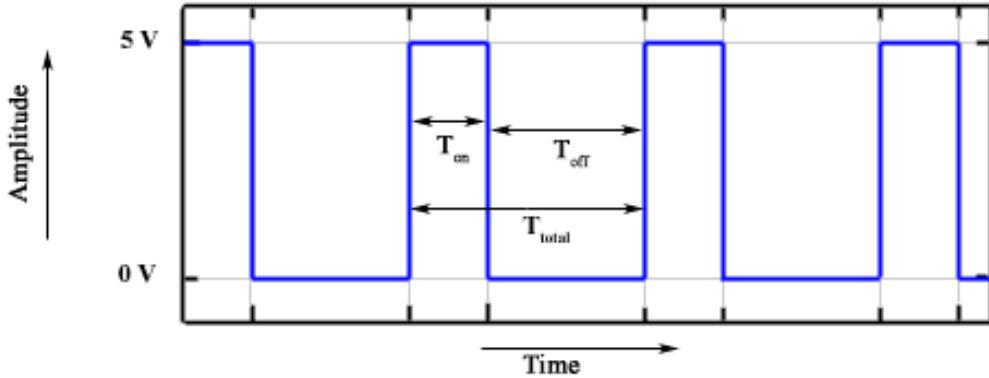


图 3.1: PWM 信号示意图

3.2 RGB 色彩模式

RGB 色彩模式是工业界的一种颜色标准，是通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的，R、G、B 分别代表红、绿、蓝三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是运用最广的颜色系统之一。

RGB 是从颜色发光的原理来设计定的，通俗点说它的颜色混合方式就好像有红、绿、蓝三盏灯，当它们的光相互叠合的时候，色彩相混，而亮度却等于三者亮度之总和，越混合亮度越高，即加法混合。

红、绿、蓝三个颜色通道每种色各分为256阶亮度，在0时“灯”最弱——是关掉的，而在255时“灯”最亮。当三色灰度数值相同时，产生不同灰度值的灰色调，即三色灰度都为0时，是最暗的黑色调；三色灰度都为255时，是最亮的白色调。

3.3 实验用品

Arduino UNO 开发板、USB 数据线、面包板、RGB LED 模块、3个 1k 电阻、若干杜邦线。

3.4 实验步骤

- (1) 按照如图3.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 观察 RGB 呼吸灯模块的变化。

3.5 接线图

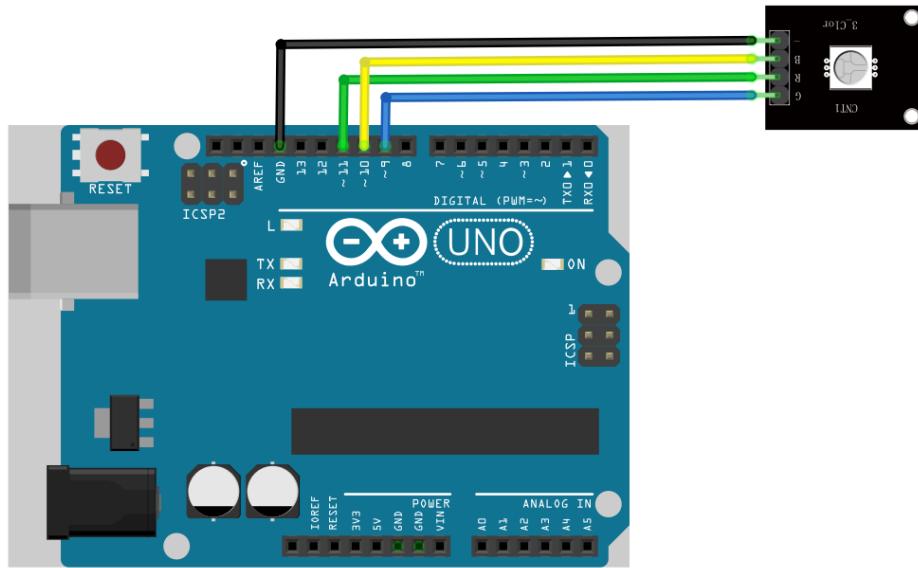


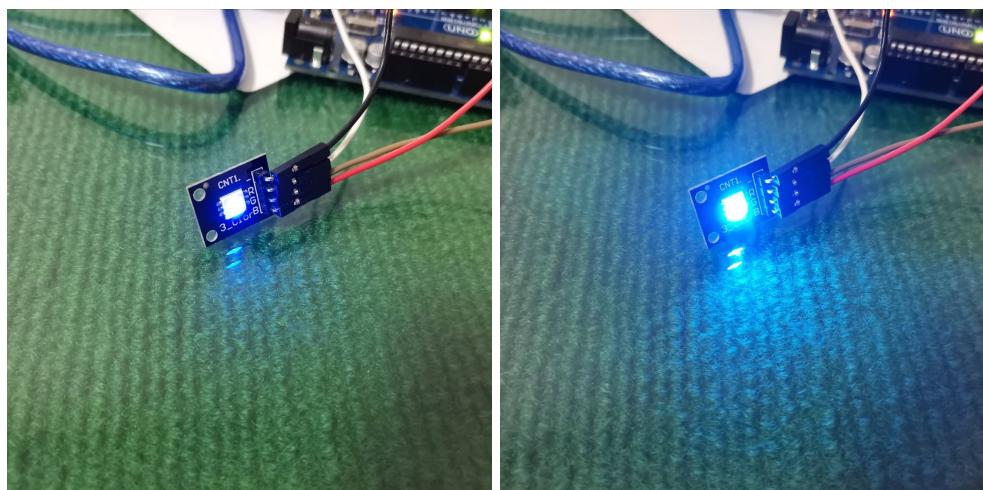
图 3.2: 实验接线图

3.6 实验程序

```
1 // RGB_Breathing_Light.ino
2 // Date: 2020/04/17
3
4 #define RGB_RED    11
5 #define RGB_GREEN  10
6 #define RGB_BLUE   9
7
8 void setup()
9 {
10     pinMode(RGB_RED, OUTPUT);
11     pinMode(RGB_GREEN, OUTPUT);
12     pinMode(RGB_BLUE, OUTPUT);
13 }
14
15 void setColor(int r, int g, int b)
16 {
17     analogWrite(RGB_RED, r);
18     analogWrite(RGB_GREEN, g);
19     analogWrite(RGB_BLUE, b);
20 }
21
22 void loop()
```

```
23  {
24      for(int i=0; i<256; i++)
25      {
26          setColor(i,0,0);
27          delay(10);
28      }
29      delay(1000);
30
31      for(int i=0; i<256; i++)
32      {
33          setColor(0,i,0);
34          delay(10);
35      }
36      delay(1000);
37
38      for(int i=0; i<256; i++)
39      {
40          setColor(0,0,i);
41          delay(10);
42      }
43      delay(1000);
44 }
```

3.7 实验结果



实验四 按键模块实验

4.1 实验目的

在本实验中，我们希望通过按键模块来点亮和熄灭 LED 灯。

4.2 按键模块

按键模块采用手感舒适的轻触按键，当按下按键，直接将按键的两极导通连接，而按键两极直接引出连接 Arduino 的信号线与负极，所以按下按键就表示信号线与负极导通。

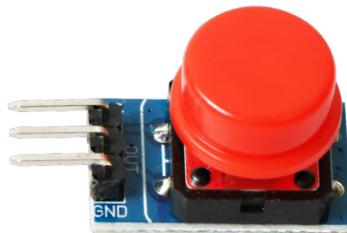


图 4.1: 按键模块示意图

数字 I/O 端口的含义是 INPUT 和 OUTPUT 接口，在以前的 LED 灯实验中，我们只使用 GPIO 的 OUTPUT 功能。现在让我们尝试在 Arduino 中使用 I/O 的 INPUT 功能，即该功能是在本实验中从外部设备读取输出值。我们使用按键和 LED 灯来完成使用 INPUT 和 OUTPUT 作为组合的实验，即按下按钮时 LED 灯点亮，松开按钮时 LED 灯熄灭。

4.3 实验原理

当按下按钮时，D7 号接口处于高电平，它将 D11 号输出引脚置高电平，这可以点亮 LED 灯。当松开按钮时，D7 号接口读为低电平时，D11 号输出保持低电平，此时 LED 灯熄灭。

4.4 实验用品

Arduino UNO 开发板、USB 数据线、面包板、按键模块、LED灯、 220Ω 电阻、若干杜邦线。

4.5 实验步骤

- (1) 按照如图4.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 分别观察在按下和松开按键这两种情况下 LED 灯的状态。

4.6 接线图

我们将按键连接到 D7 号接口，红灯连接到 D11 号接口（Arduino 控制器中的所有 D0-D13 数字 I/O 接口可用于连接按键和指示灯，但尽量不要选择数字 D0 和 D1 接口，因为 D0 和 D1 功能复用，除了 I/O 端口功能外，它们还用作串行通信接口）。下载程序时，设备正在与 PC 通信。所以我们应该保留 D0 和 D1 接口为了避免插入线路的麻烦，我们不选择 D0 号和 D1 号接口。

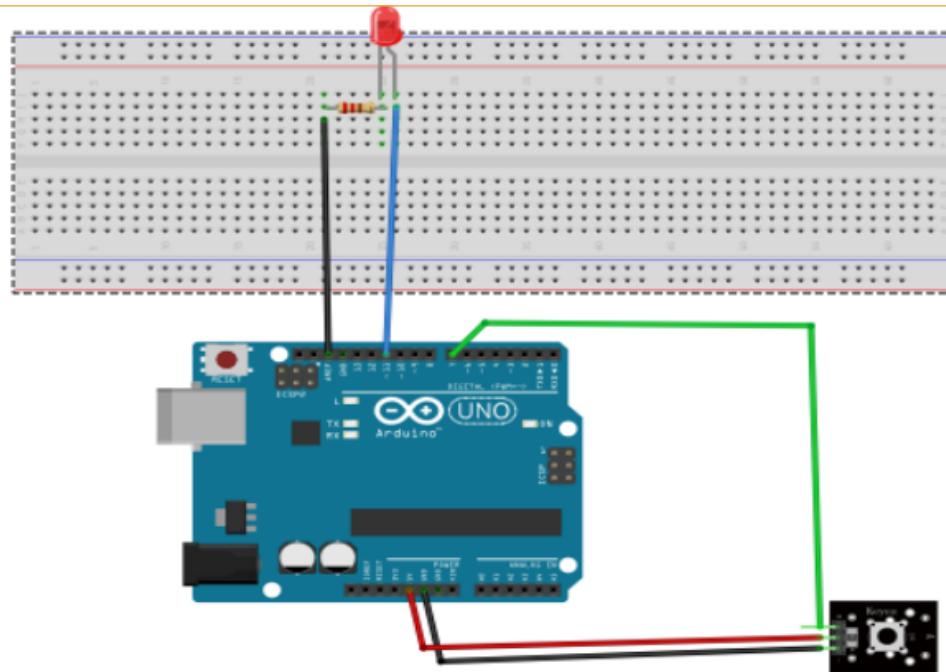


图 4.2: 实验接线图

4.7 实验程序

```
1 // Button.ino
2 // Date: 2020/04/18
3
4 int led_out = 11 ;
5 int keypad_pin = 7;
6
```

```
7 int getState(int iterator)
8 {
9     for(int i = 0; i < iterator; i++)
10    {
11        if(digitalRead(keypad_pin)==0)
12        {
13            return 0;
14        }
15    }
16    return 1;
17 }
18
19 void setup()
20 {
21     pinMode(led_out, OUTPUT);
22     pinMode(keypad_pin, INPUT);
23     digitalWrite(led_out, LOW);
24     Serial.begin(9600);
25 }
26
27 void loop()
28 {
29     int state = getState(100);
30
31     delay(1000);
32     if(state == 0)
33     {
34         digitalWrite(led_out, HIGH);
35         Serial.println("turn on the LED");
36     }
37     else
38     {
39         digitalWrite(led_out, LOW);
40         Serial.println("turn off the LED");
41     }
42 }
```

4.8 实验结果

如图4.3所示，当程序烧录成功后，我们按下按键，可以观察到 LED 灯被点亮，而松开按键时，LED 灯被熄灭。

打开串口监视器，将波特率设置为9600 (与程序中一致)，即可看到串口监视器中不断打印出的 LED 灯的状态信息，如图4.4所示。

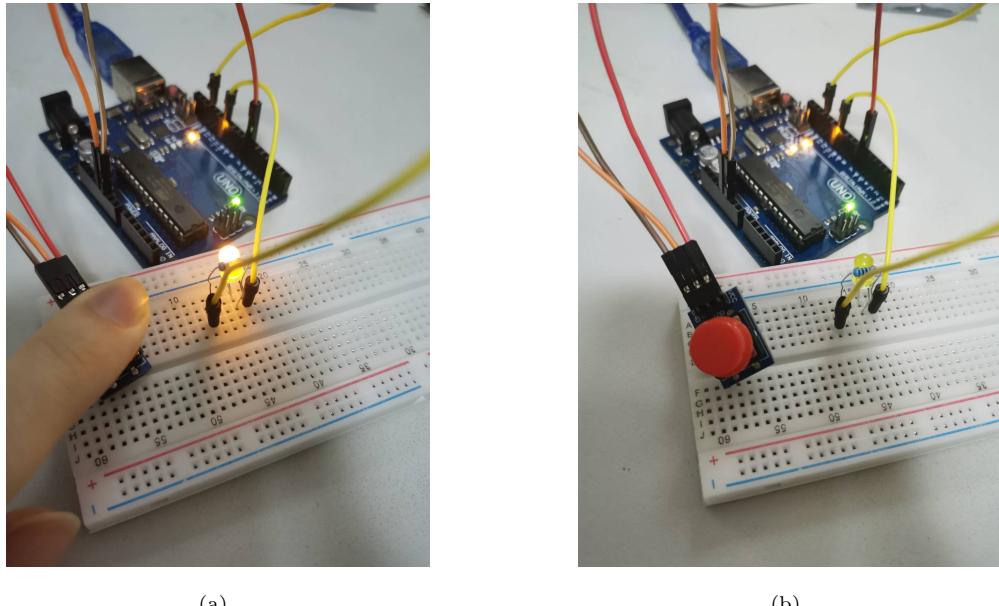


图 4.3: 按键模块实验结果示意图: (a) 按下按键时, LED 灯点亮; (b) 松开按键时, LED 灯熄灭。

A screenshot of a serial monitor window titled "COM4 (Arduino/Genuino Uno)". The window displays a series of timestamped messages indicating the state of the LED. The messages alternate between "turn off the LED" and "turn on the LED", with each message having a timestamp. At the bottom of the window, there are several control buttons: "自动滚屏" (Auto scroll), "Show timestamp" (which is checked), "换行符" (Line feed), "9600 波特率" (9600 baud rate), and "清空输出" (Clear output).

Timestamp	Action
20:32:16.796	turn off the LED
20:32:17.774	turn off the LED
20:32:18.783	turn off the LED
20:32:19.792	turn off the LED
20:32:20.771	turn off the LED
20:32:21.784	turn off the LED
20:32:22.796	turn off the LED
20:32:23.774	turn off the LED
20:32:24.786	turn on the LED
20:32:25.793	turn on the LED
20:32:26.772	turn on the LED
20:32:27.784	turn on the LED
20:32:28.793	turn on the LED
20:32:29.772	turn on the LED
20:32:30.786	turn on the LED
20:32:31.795	turn on the LED

图 4.4: 串口监视器中不断打印出的 LED 灯的状态信息

实验五 矩阵键盘实验

5.1 实验目的

学习矩阵键盘的原理，并基于 Arduino 实现对矩阵键盘的按键识别。

5.2 矩阵键盘

矩阵键盘是单片机外部设备中所使用的排布类似于矩阵的键盘组。矩阵式结构的键盘显然比直接法要复杂一些，识别也要复杂一些，而当键盘中需要的键数比较多时，采用矩阵法做键盘是合理的。列线通过电阻接正电源，并将行线所接的单片机的 I/O 口作为输出端，而列线所接的 I/O 口则作为输入。本实验中我们使用 4×4 的矩阵键盘，其原理图如图5.1所示。

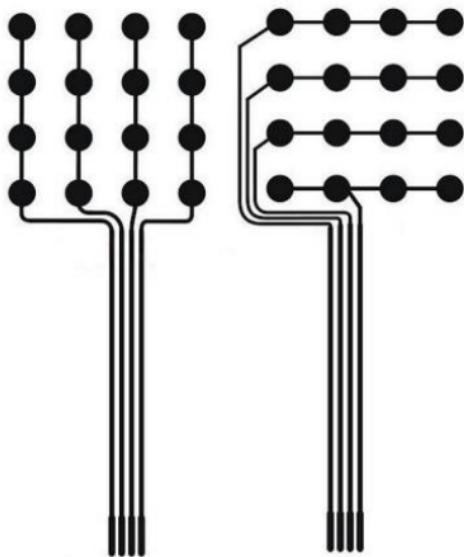


图 5.1: 本实验所用 4×4 矩阵键盘原理图

行扫描法是一种最常用的按键识别方法，以图5.1中所示的矩阵键盘为例，它的具体流程如下：

- (1) 判断键盘中有无键按下：将全部行线置于低电平，然后检测列线的状态。只要有一列的电平为低，则表示键盘中有键被按下，而且闭合的键位于低电平线与4根行线相交叉的4个按键之中。若所有列线均为高电平，则键盘中无键按下。
- (2) 判断闭合键所在的位置：在确认有键按下后，即可进入确定具体闭合键的过程。其方法是：依次将行线置为低电平，即在置某根行线为低电平时，其它线为高电平。在确定某根行线位置为

低电平后，再逐行检测各列线的电平状态。若某列为低，则该列线与置为低电平的行线交叉处的按键就是闭合的按键。

5.3 实验用品

Arduino UNO 开发板、USB 数据线、矩阵键盘、若干杜邦线。

5.4 实验步骤

- (1) 按照如图5.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 打开串口监视器，在矩阵键盘上按下不同的按钮，观察串口监视器中打印出的内容。

5.5 接线图

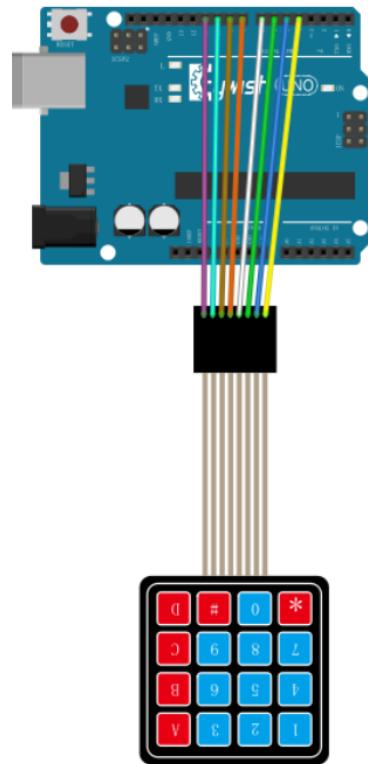


图 5.2: 实验接线图

5.6 实验程序

本实验使用到了由 Mark Stanley 与 Alexander Brevig 编写的外部库 Keypad。

```
1 // MatrixKeyboard.ino
2 // Date: 2020/04/18
3
4 #include "Keypad.h"
5
6 #define ROW_1 4
7 #define ROW_2 5
8 #define ROW_3 6
9 #define ROW_4 7
10 #define COL_1 8
11 #define COL_2 9
12 #define COL_3 10
13 #define COL_4 11
14
15 const byte ROWS = 4; //four rows
16 const byte COLS = 4; //four columns
17
18 char hexaKeys[ROWS][COLS] = {
19     {'1','2','3','A'},
20     {'4','5','6','B'},
21     {'7','8','9','C'},
22     {'*','0','#','D'}
23 };
24 byte rowPins[ROWS] = {ROW_1, ROW_2, ROW_3, ROW_4};
25 byte colPins[COLS] = {COL_1, COL_2, COL_3, COL_4};
26
27 Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
28
29 void setup()
30 {
31     for(int i=0; i < ROWS; i++)
32     {
33         pinMode(rowPins[i], OUTPUT);
34         pinMode(colPins[i], OUTPUT);
35     }
36     Serial.begin(115200);
37 }
38
39 void loop()
40 {
41     char customKey = customKeypad.getKey();
42     if (customKey){
43         Serial.println(customKey);
44     }
45 }
```

5.7 实验结果

如图5.3所示，当程序成功烧录后，我们打开串口监视器，并将波特率设置为 115200 (与程序中保持一致)。当我们按下矩阵键盘上的任意键时，都会看到在串口监视器中打印出了该键上的内容。

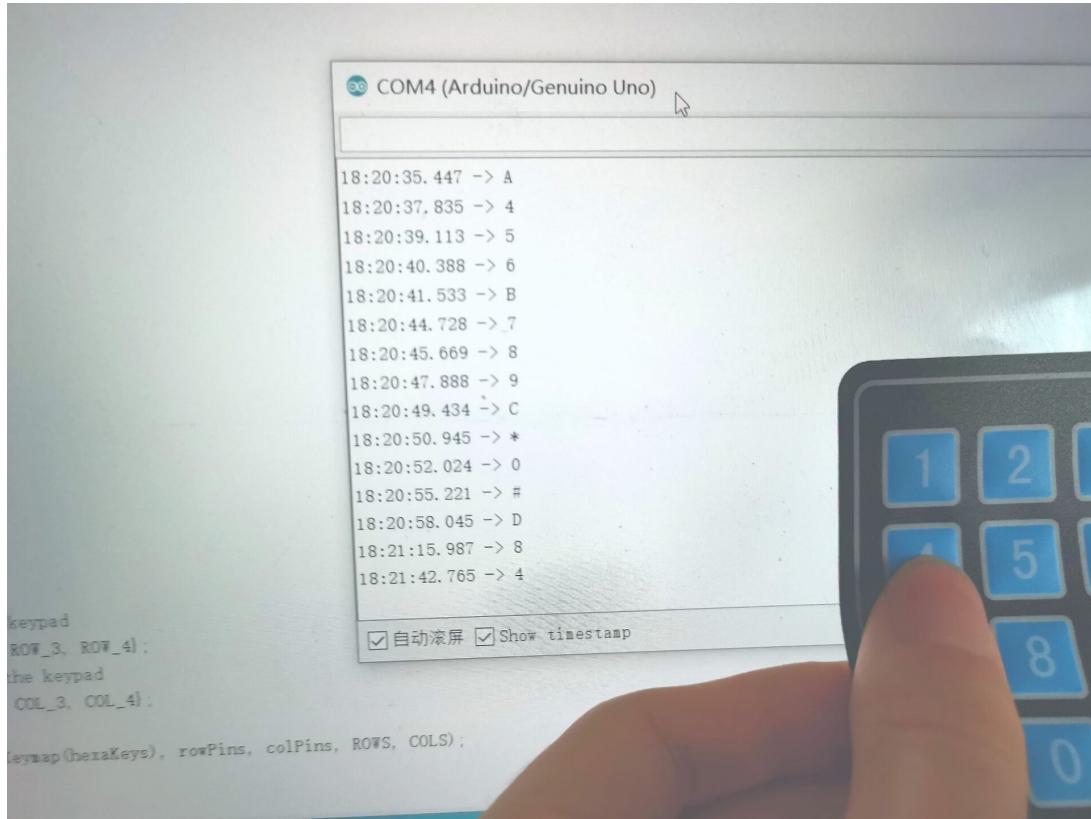


图 5.3: 当按下矩阵键盘上的任意键时，串口监视器中会打印出相应的内容

实验六 PWM 调节 LED 灯亮度实验

6.1 实验目的

学习电位器的原理使用，并实现使用电位器来控制 LED 灯的亮度。

6.2 电位器

电位器 (英文：Potentiometer，通俗上也简称 Pot，少数直译成电位计)，中文通常又称为可变电阻器 (VR，Variable Resistor) 或简称可变电阻，是一种多数具有三个端口，其中有两个固定接点与一个滑动接点，可经由滑动而改变滑动端与两个固定端间电阻值的电子零件，属于被动元件，使用时可形成不同的分压比率，改变滑动点的电位，因而得名。



图 6.1: 电位器实物图

电位器在电路中的作用主要有如下三个方面：

- (1) **用作分压器：**电位器是一个连续可调的电阻器，当调节电位器的转柄或滑柄时，动触点在电阻体上滑动。此时在电位器的输出端可获得与电位器外加电压和可动臂转角或行程成一定关系的输出电压。如果将电位器作为可变分压电阻用，则应将其一端接输入电压，中间端接输出，另一端接地。
- (2) **用作变阻器：**如果将电位器作为可变电阻用，应把它接成两端器件，即将其一端接输入电压，中间端接输出，另一端悬空或与中间端连接。这样当调节电位器的转柄或滑柄时，便可获得一个平滑连续变化的电阻值。

(3) 用作电流控制器：当电位器作为电流控制器使用时，其中一个选定的电流输出端必须是滑动触点引出端。

6.3 实验原理

在本实验中，我们不断通过 `analogRead(indexOfAnalog)` 函数来读取 AD 接口的采集值，并将其转换为电压值，然后将其转换为相应的 PWM 占空比，并使用 `analogWrite(pwm.pin, value)` 函数导出其平均电压值。我们可以通过观察 LED 灯的亮度变化来判断电压的变化。

6.4 实验用品

Arduino UNO 开发板、USB 数据线、面包板、10k 电位器、LED、 220Ω 电阻、若干杜邦线。

6.5 实验步骤

- (1) 按照如图6.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 不断旋转电位器的转柄，观察并记录 LED 灯的亮度变化；同时打开串口监视器，查看打印出的电压值和其对应的 PWM 占空比信息。

6.6 接线图

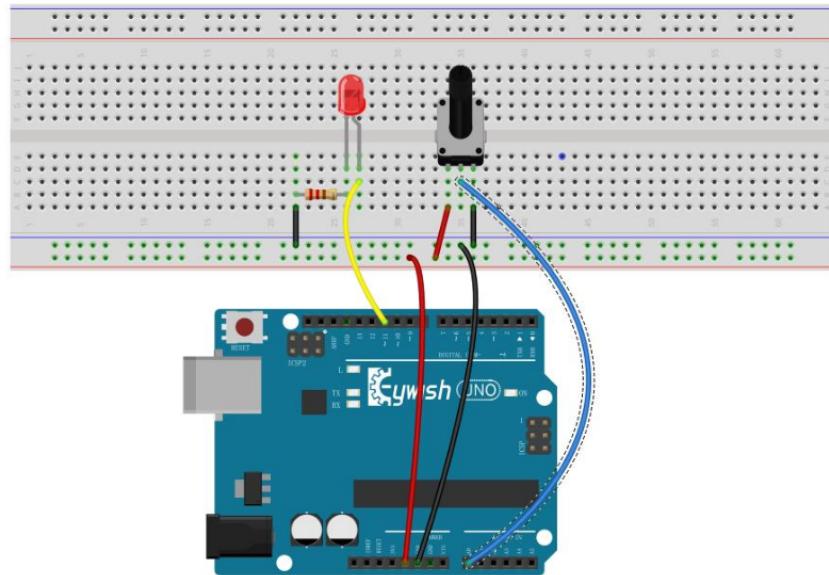


图 6.2: 实验接线图

6.7 实验程序

```
1 // pwm.ino
2 // Date: 2020/04/18
3
4 int ADPIN      = A0 ;
5 int PWM_LEDPIN = 11 ;
6 int value      = 0 ;
7 float voltage   = 0.0 ;
8
9 void setup()
10 {
11     pinMode(ADPIN, INPUT);           // define ADPIN input PWM_LEDPIN output
12     pinMode(PWM_LEDPIN, OUTPUT);
13     Serial.begin(9600);            //Serial Baud rate is 115200
14 }
15
16 void loop()
17 {
18     value = analogRead(ADPIN);       //read analog pin raw data
19
20     voltage = (float)value / 1023;
21     Serial.print("voltage==");
22     Serial.println(voltage);
23
24     value = (int)(voltage * 256);    //covert to voltage to PWM duty cycle
25     Serial.print("value==");
26     Serial.println(value);
27
28     analogWrite(PWM_LEDPIN,value);
29     delay(1000);
30 }
```

6.8 实验结果

如图6.3所示，当程序烧录成功后，我们开始转动电位器的转柄，可以观察到 LED 逐渐变亮。同样地，向相反方向不断转动电位器的转柄，可以观察到 LED 逐渐变暗。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的电压值和其对应的 PWM 占空比，当不断转动电位器的转柄时，打印出的这两个数值会相应地变化，如图6.4所示。

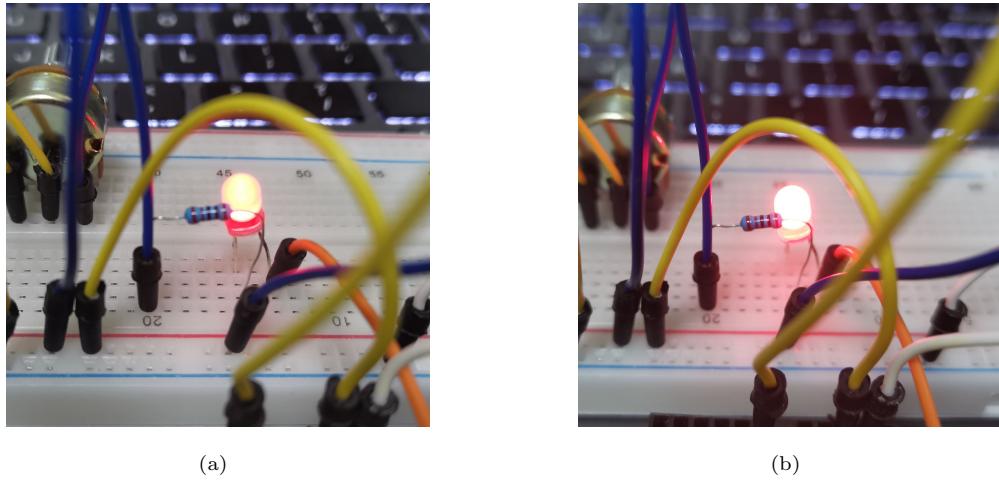


图 6.3: LED 亮度变化观察结果示意图: (a) 开始转动电位器的转柄, 起初 LED 灯亮度较小; (b) 继续向相同方向转动转柄, LED 灯的亮度逐渐变大。

```
00:08:47.022 -> value==0
00:08:47.998 -> voltage==0.08
00:08:48.032 -> value==21
00:08:49.006 -> voltage==0.18
00:08:49.006 -> value==45
00:08:50.016 -> voltage==0.23
00:08:50.016 -> value==58
00:08:50.994 -> voltage==0.30
00:08:51.028 -> value==77
00:08:52.006 -> voltage==0.39
00:08:52.006 -> value==99
00:08:53.017 -> voltage==0.49
00:08:53.017 -> value==124
00:08:53.991 -> voltage==0.53
00:08:54.025 -> value==135
00:08:54.999 -> voltage==0.53
```

图 6.4: 串口监视器中不断打印出的电压值和其对应的 PWM 占空比

实验七 A/D 模数转换实验

7.1 实验目的

在这个实验中，我们希望将电位器的电阻值转换为模拟值并将其读出，然后将其打印在串口监视器中。

7.2 A/D 转换

A/D 转换就是模数转换，也可以是整流。顾名思义，就是把模拟信号转换成数字信号。模拟量可以是电压、电流等电信号，也可以是压力、温度、湿度、位移、声音等非电信号。但在 A/D 转换前，输入到 A/D 转换器的输入信号必须经各种传感器把物理量转换成电压信号。A/D 转换后，输出的数字信号可以有8位、10位、12位、14位和16位等。

D/A 转换就是数模转换，就是将离散的数字量转换为连接变化的模拟量。与数模转换相对应的就是模数转换，模数转换是数模转换的逆过程。

Arduino 有6个模拟接口，编号从0到5，6个接口也可以是接口函数重用。除模拟接口功能外，六个接口还可用作数字接口，编号为14-19。

7.3 实验原理

通过函数 `analogRead(indexOfAnalog)` 函数来读取 AD 接口的采集值。Arduino 328p 采用10位 A/D 采样，因此模拟测量值范围为 0-1023，数值只是 AD 的值，需要将其转换为实际电压值。我们将使用如下公式来计算实际电压：

$$V_R = \frac{\text{Value}}{2^{10} - 1} \times V_{DD}$$

式中 V_R 为实际电压， Value 为 A/D 采用值， V_{DD} 为 A/D 参考电压值。

7.4 实验用品

Arduino UNO 开发板、USB 数据线、面包板、10k 电位器、LED、220Ω 电阻、若干跳线。

7.5 实验步骤

- (1) 按照如图7.1所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 打开串口监视器，不断旋转电位器的转柄，查看打印出的模拟接口采集值和转换后的电压值。

7.6 接线图

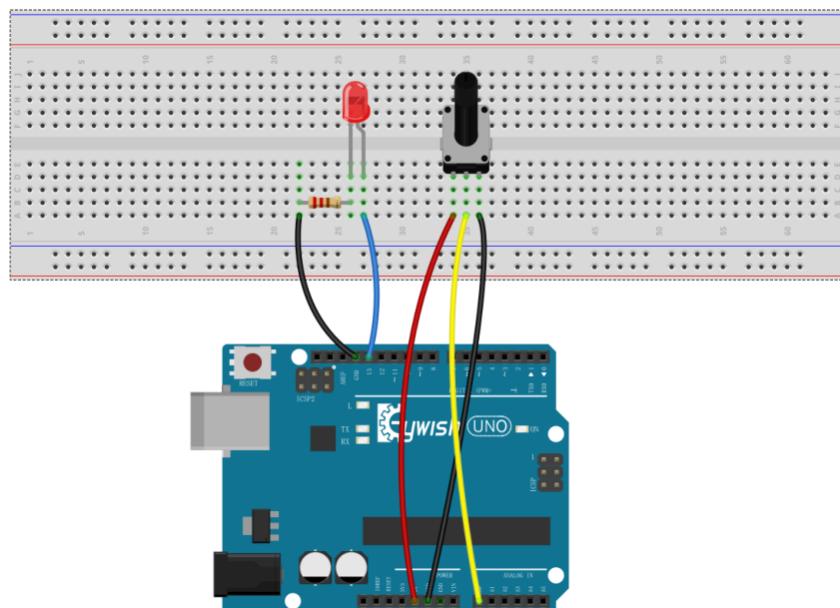


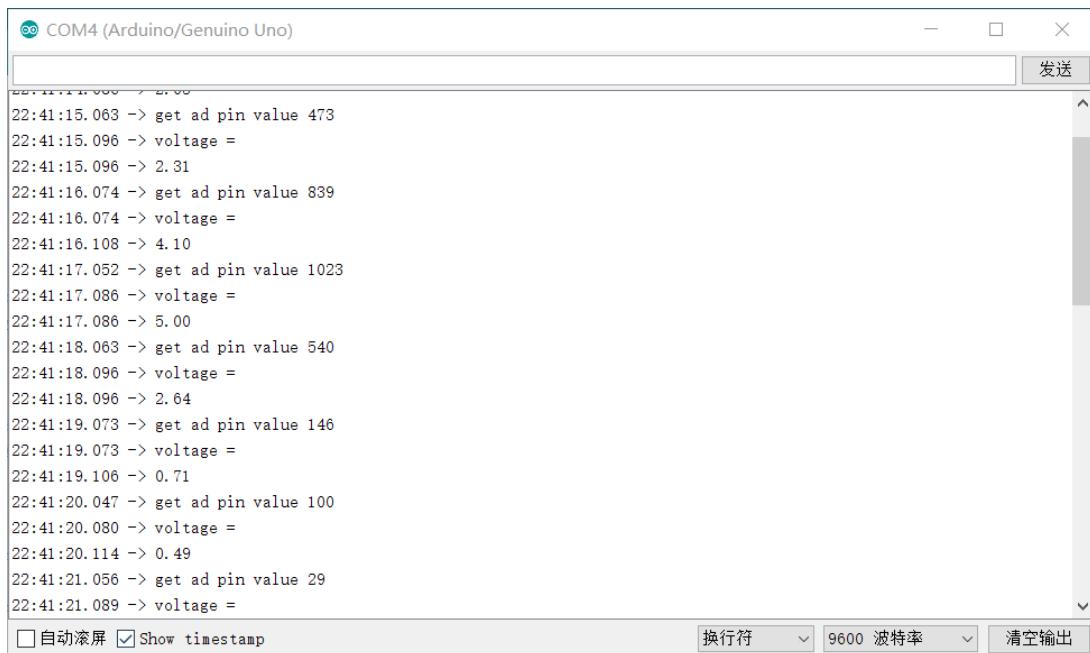
图 7.1: 实验接线图

7.7 实验程序

```
1 // ad.ino
2 // Date: 2020/04/22
3
4 int ADPIN = A0 ;
5 int LEDPIN = 13 ;
6 int value = 0 ;
7 float voltage = 0.0 ;
8
9 void setup()
10 {
11     pinMode(ADPIN, INPUT); // define ADPIN to be input
12     pinMode(LEDPIN, OUTPUT); // define LEDPIN to be output
13     Serial.begin(9600); // Serial Baud rate is 115200
```

```
14 }
15
16 void loop()
17 {
18     digitalWrite(LEDPIN, HIGH);           // light on led
19     value = analogRead(ADPIN);          // read analog pin raw data
20     voltage = ((float)value) / 1023 ;
21     voltage = voltage * 5;             // convert analog raw data to real voltage = (
22     //analog/1023)*5
23     Serial.print("get ad pin value ");
24     Serial.print(value);
25     Serial.println("\nvoltage = ");
26     Serial.println(voltage);
27     delay(1000);
28 }
```

7.8 实验结果



实验八 蜂鸣器模块实验

8.1 实验目的

Arduino 可用于创建大量的交互式工作，最常见的是声光显示。我们之前已经在实验中使用过 LED，现在我们开动蜂鸣器来播放两个频率的声音。只要频率与乐谱相匹配，我们就能听到美妙的音乐。

8.2 蜂鸣器

蜂鸣器 (Buzzer) 是产生声音的信号装置，是一种一体化结构的电子讯响器，它采用直流电压供电，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。蜂鸣器的典型应用包括警笛，报警装置，火灾警报器，防空警报器，防盗器，定时器。

按照结构区分，蜂鸣器分为压电式蜂鸣器和电磁式蜂鸣器。压电式为压电陶瓷片发音，电流比较小一些，电磁式蜂鸣器为线圈通电震动发音，体积比较小。

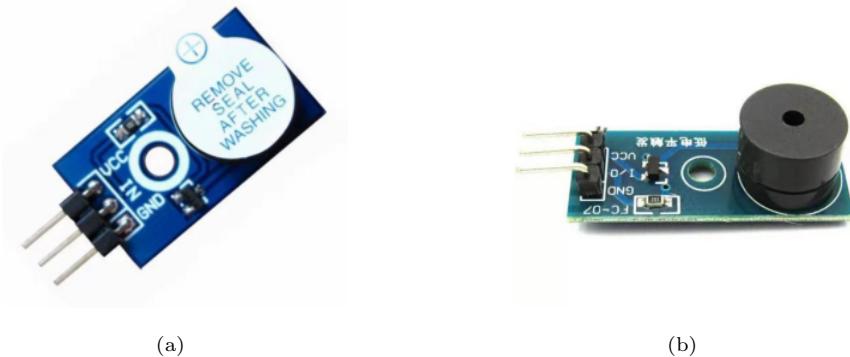


图 8.1: (a) 有源蜂鸣器模块; (b) 无源蜂鸣器模块

按照驱动方式区分，蜂鸣器分为有源蜂鸣器和无源蜂鸣器。这里的有源和无源不是指电源，而是振荡源。有源蜂鸣器内部带了振荡源，只要通电就会发出响声，但是它发出声音的频率是固定的。而无源蜂鸣器内部是不带振荡源的，必须用 500 Hz - 4.5 kHz 之间的脉冲频率信号来驱动它才会响。有源蜂鸣器往往比无源蜂鸣器贵一些，因为里边多了振荡电路，驱动发音也简单，靠电平就可以驱动，而无源蜂鸣器价格比较便宜，但是无源蜂鸣器声音频率可以控制，而音阶与频率又有确定的对

应关系，因此就可以用无源蜂鸣器做出来“do re mi fa sol la si”的效果，可以用它制作出简单的音乐曲目，比如生日歌、两只老虎等等。

蜂鸣器模块有三个管脚，其中标注‘-’的管脚接地 (GND)，中间的管脚接 5V，(图8.1中左侧) 标注‘S’的管脚接信号 (数字 I/O)。

8.3 实验用品

Arduino UNO 开发板、USB 数据线、面包板、有源蜂鸣器模块、无源蜂鸣器模块、若干杜邦线。

8.4 实验步骤

- (1) 按照如图8.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 分别对有源蜂鸣器与无源蜂鸣器进行实现，比较他们发出的声音。

8.5 接线图

在本实验中，有源蜂鸣器与无源蜂鸣器的接线图一样，均如图8.2所示。

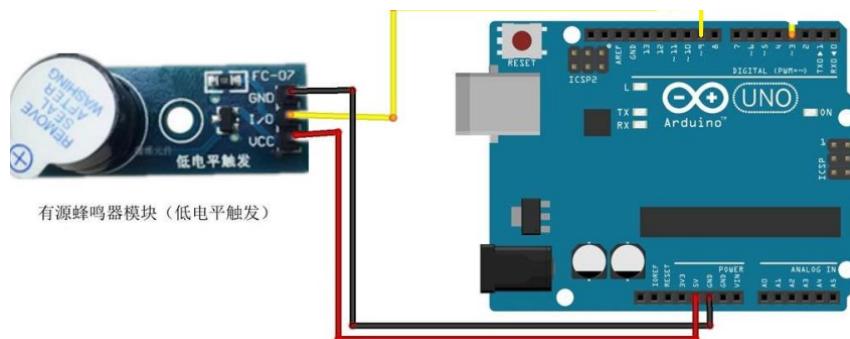


图 8.2: 实验接线图

8.6 实验程序

8.6.1 有源蜂鸣器实验程序

```

1 // ActiveBuzzer.ino
2 // Date: 2020/04/22
3
4 int buzzer = 9;
5
6 void setup()

```

```
7  {
8      pinMode(buzzer, OUTPUT);
9  }
10
11 void loop()
12 {
13     digitalWrite(buzzer, HIGH); // sound production
14     delay(1000);
15     digitalWrite(buzzer, LOW);
16     delay(1000);
17 }
```

8.6.2 无源蜂鸣器实验程序

```
1 // Buzzer.ino
2 // Date: 2020/04/22
3
4 int buzzer = 9;
5
6 void frequence_1(void)    // 1k HZ
7 {
8     for(int i=0; i<800; i++)
9     {
10         digitalWrite(buzzer, HIGH); // sound production
11         delay(0.5);
12         digitalWrite(buzzer, LOW);
13         delay(0.5);
14     }
15 }
16
17 void frequence_2(void)    // 250 HZ
18 {
19     for(int i=0; i<800; i++)
20     {
21         digitalWrite(buzzer, HIGH); // sound production
22         delay(2);
23         digitalWrite(buzzer, LOW);
24         delay(2);
25     }
26 }
27
28 void setup()
29 {
30     pinMode(buzzer, OUTPUT);
31 }
32
33 void loop()
34 {
```

```
35     frequence_1();  
36     delay(1000);  
37     frequence_2();  
38     delay(1000);  
39 }
```

8.7 实验结果

有源蜂鸣器只能发出一种频率的声音，而无源蜂鸣器可以发出多种频率的声音。

第二部分

传感器模块实验

传感器分类

- 电阻式传感器：水深传感器、雨滴传感器、土壤湿度传感器、倾斜开关；
- 电容式传感器：触摸开关、声音传感器；
- 光电式传感器：灰度传感器、光敏电阻、光电传感器；
- 磁敏式传感器：霍尔开关、小磁簧开关；
- 热电式传感器：LM35 温度传感器、热敏电阻；
- 化学式传感器：MQ-2 气体传感器；
- 辐射与波式传感器：火焰传感器、红外循迹模块、红外避障模块、手指侦测心跳模块；
- 复合传感器：温湿度传感器。

实验九 四位七段数码管模块显示实验

9.1 实验目的

在本实验中，我们希望通过如下两个小项目来学习四位七段数码管的结构和工作原理。

- (1) 在四位七段数码管模块上显示“1234”这四个字符；
- (2) 基于四位七段数码管模块实现计时功能（例如从十二点开始）。

9.2 七段数码管

9.2.1 数码管的结构

七段数码管是一类价格便宜使用简单，通过对其不同的管脚输入相对的电流，使其发亮，从而显示出数字能够显示时间、日期、温度等所有可用数字表示的参数的器件。在电器领域，特别是家电领域应用极为广泛，如显示屏、空调、热水器、冰箱等等。绝大多数热水器用的都是数码管，其他家电也用液晶屏与荧光屏。

数码管可分为七段数码管和八段数码管，区别在于八段数码管比七段数码管多一个用于显示小数点的发光二极管单元 DP (decimal point)，其基本单元是发光二极管。如图9.1所示，七段数码管由七段条形发光二极管和一个圆点发光二极管组成。我们可以通过控制七段条形发光二极管的亮与灭来显示 0-F (0-9 与 A-F) 这16个字符及其他特殊字符，在八段数码管中，除了上述功能外，我们还可以通过控制 DP 发光二极管的亮灭可以控制是否显示小数点。

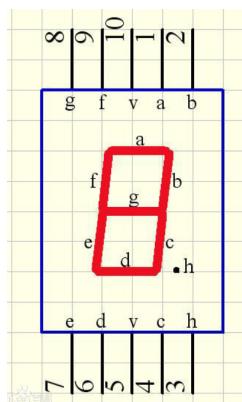


图 9.1: 一个数码管的八段编号及对应引脚示意图

9.2.2 四位七段数码管模块

如图9.2所示，四位七段数码管模块由一个12管脚的四位七段共阳极数码管和一个控制芯片TM1637构成。该模块采用沉金工艺，外观更加美观，同时采用防插反接口，操作更加安全，插口一边有大写字母I表示该模块采用IIC协议通信，另一边是数码管的图标标志。此产品可以应用在时间显示，跑表显示以及其他需要显示数字的设备上。该模块的规格参数由表9.1给出，同时，该模块有四个引脚，其意义由表9.2给出。



图 9.2: 四位七段数码管模块示意图

工作电压	+5V
尺寸大小	45mm×25mm
重量大小	8g
通信协议	IIC

表 9.1: 四位七段数码管模块的规格参数

引脚	意义
GND	接地引脚
VCC	电源引脚，接电源正极，+5V
DIO	数据IO口，可以接任意的数字引脚
CLK	时钟引脚，可以接任意的数字引脚

表 9.2: 四位七段数码管模块四个引脚的意义

9.2.3 四位数码管的动态扫描原理

数码管的动态扫描利用了人的视觉暂留原理，它快速循环显示各个数码管的字符，从而形成连续的字符串。下面我们以在四位七段数码管中显示“1234”为例对此过程进行详细说明：

- (1) 第1位数码管显示“1”，第2、3、4位数码管无显示内容；
- (2) 经过时间t后，第2位数码管显示“2”，第1、3、4位数码管无显示内容；
- (3) 经过时间t后，第3位数码管显示“3”，第1、2、4位数码管无显示内容；
- (4) 经过时间t后，第4位数码管显示“4”，第1、2、3位数码管无显示内容；
- (5) 经过时间t后，返回步骤(1)，如此循环往复。

在上述步骤中，扫描显示时间间隔 t 是一个很关键的参数， t 太长将会导致数码管闪烁，我们一般取 t 为 5ms 或更短。

9.3 实验用品

Arduino UNO 开发板、USB 数据线、四位七段数码管模块、若干杜邦线。

9.4 实验步骤

- (1) 按照如图9.3所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 观察数码管上的显示结果。

9.5 接线图

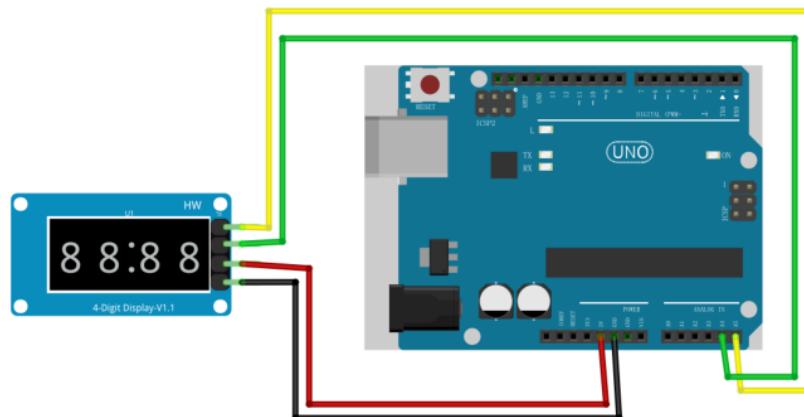


图 9.3: 实验接线图

9.6 实验程序

9.6.1 显示“1234”

```
1 // _4x7Segment_Display.ino
2 // Date: 2020/05/03
3
4 #include "TM1637.h"
5 #define CLK A5
6 #define DIO A4
7
```

```
8 int8_t Disp[] = {1, 2, 3, 4};  
9  
10 TM1637 _4Segment_Display(CLK, DIO);  
11  
12 void setup()  
13 {  
14     _4Segment_Display.set();  
15     _4Segment_Display.init();  
16     _4Segment_Display.point(POINT_OFF);  
17 }  
18 void loop()  
19 {  
20     _4Segment_Display.display(Disp);  
21     while(1);  
22 }
```

9.6.2 计时功能实现

```
1 // TimerDisplay.ino  
2 // Date: 2020/05/03  
3  
4 #include "TimerOne.h"  
5 #include "TM1637.h"  
6 #define ON 1  
7 #define OFF 0  
8 #define CLK A5  
9 #define DIO A4  
10  
11 int8_t TimeDisp[] = {0x00,0x00,0x00,0x00};  
12 unsigned char ClockPoint = 1;  
13 unsigned char Update;  
14 unsigned char halfsecond = 0;  
15 unsigned char second;  
16 unsigned char minute = 0;  
17 unsigned char hour = 12;  
18  
19 TM1637 tm1637(CLK,DIO);  
20  
21 void setup()  
22 {  
23     tm1637.set();  
24     tm1637.init();  
25     Timer1.initialize(500000);  
26     Timer1.attachInterrupt(TimingISR);  
27 }  
28  
29 void loop()  
30 {
```

```
31     if(Update == ON)
32     {
33         TimeUpdate();
34         tm1637.display(TimeDisp);
35     }
36 }
37
38 void TimingISR()
39 {
40     halfsecond++;
41     Update = ON;
42     if(halfsecond == 2){
43         second++;
44         if(second == 60)
45         {
46             minute++;
47             if(minute == 60)
48             {
49                 hour++;
50                 if(hour == 24)
51                     hour = 0;
52                 minute = 0;
53             }
54             second = 0;
55         }
56         halfsecond = 0;
57     }
58 // Serial.println(second);
59 ClockPoint = (~ClockPoint) & 0x01;
60 }
61
62 void TimeUpdate(void)
63 {
64     if(ClockPoint) tm1637.point(POINT_ON);
65     else tm1637.point(POINT_OFF);
66     TimeDisp[0] = hour / 10;
67     TimeDisp[1] = hour % 10;
68     TimeDisp[2] = minute / 10;
69     TimeDisp[3] = minute % 10;
70     Update = OFF;
71 }
```

9.7 实验结果

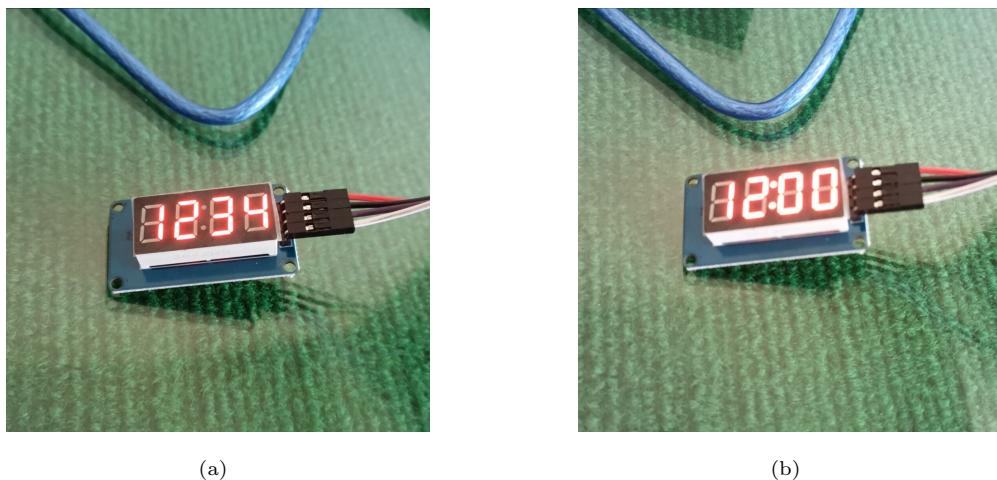


图 9.4: 四位七段数码管模块实验结果示意图: (a) 显示数字; (b) 计时

实验十 MAX7219 驱动单个 8×8 点阵模块

10.1 实验目的

基于 Arduino 实现使用 MAX7219 驱动单个 8×8 点阵模块。

10.2 MAX 7219

MAX7219/MAX7221 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。

一个方便的四线串行接口可以联接通用的微处理器。每个数据可以寻址在更新时不需要改写所有的显示。MAX7219 同样允许用户对每一个数据选择编码或者不编码。整个设备包含一个 $150\mu\text{A}$ 的低功耗关闭模式，模拟和数字亮度控制，一个扫描限制寄存器允许用户显示 1-8 位数据，还有一个让所有 LED 发光的检测模式。

总之，此点阵模块中，MAX7219 集成电路就是来帮助单片机输出显示的。如果用普通的方法来驱动一个 8×8 的点阵，需要用到8个单片机 IO，2个就是16个 IO，如果驱动几个点阵，单片机的 IO 就不够了，如果用 MAX7219 帮忙的话，用单片机的3个 IO 口就可以驱动 1 个/10 个/20 个点阵。只点阵显示时无闪烁！支持级联！

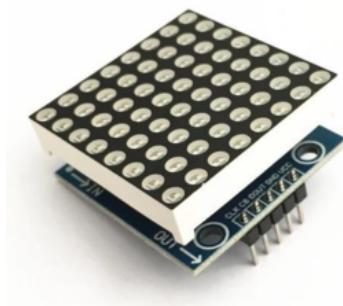


图 10.1: MAX7219 模块示意图

10.3 实验用品

Arduino UNO 开发板、USB 数据线、MAX7219 显示驱动器、八位七段数字 LED 模块、若干杜邦线。

10.4 接线图

VCC 接 5V、GND 接 GND、CLK 接 5、CS 接 6、DIN 接 7。

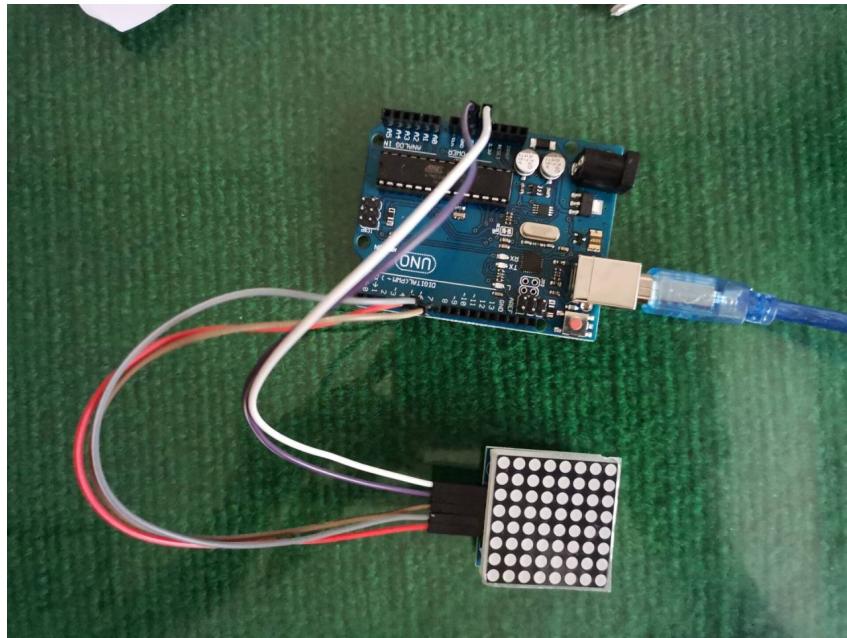


图 10.2: 实验接线图

10.5 实验程序

```
1 // MaxMatrix.ino
2 // Date: 2020/05/03
3
4 #include "MAX7219_MaxMatrix.h"
5
6 //define Max7219 pins
7 #define PIN_DIN    7 //max 7219
8 #define PIN_CS     6
9 #define PIN_CLK    5
10 // init Max7219 LED Matrix, 1 module
11 MaxMatrix ledmatrix(PIN_DIN,PIN_CS,PIN_CLK, 1);
12
13 void setup(){
14     Serial.begin(9600);
15     ledmatrix.init();
```

```

16     ledmatrix.setIntensity(1);
17     ledmatrix.clearMatrix();
18     Serial.println("zero:0\nnone:1\nntwo:2\nthree:3\nfour:4\nfive:5\nsix:6
19                               \nseven:7\neight:8\nnine:9\nsmile:10\nhappyOpen:11
20                               \nhappyClosed:12\nheart:13\nbigSurprise:14\nsmallSurprise:15
21                               \ntongueOut:16\nvamp1:17\nvamp2:18\nlineMouth:19\nconfused:20
22                               \ndiagonal:21\nsad:22\nsadOpen:23\nsadClosed:24\nokMouth:25
23                               \nxMouth:26\ninterrogation:27\nthunder:28\nculito:29
24                               \nangry:30");
25 }
26
27 void loop() {
28     while(Serial.available())
29     {
30         int val = Serial.parseInt();
31         if (val >= 0 && val <= 30)
32         {
33             ledmatrix.writeFull(ledmatrix.getMouthShape(val));
34         }
35         delay(2000);
36     }
37 }
```

10.6 实验结果

打开串口监视器，可以看到其中打印出的如下信息：

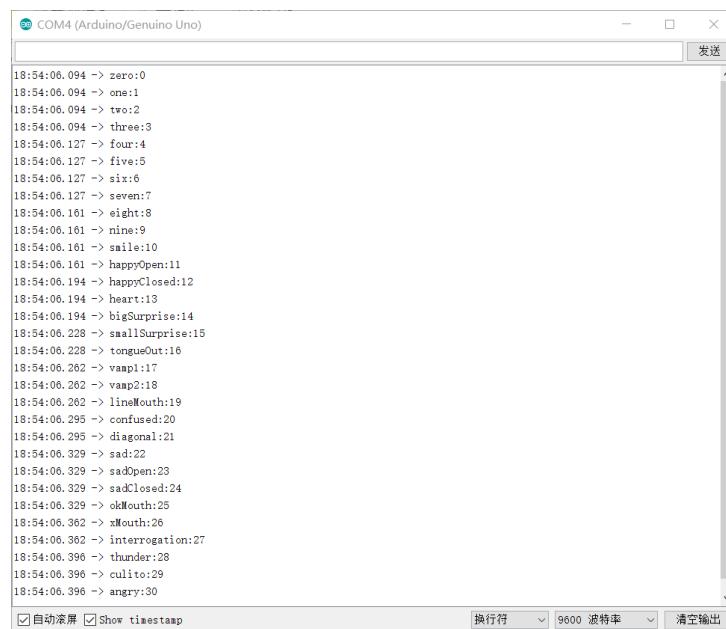
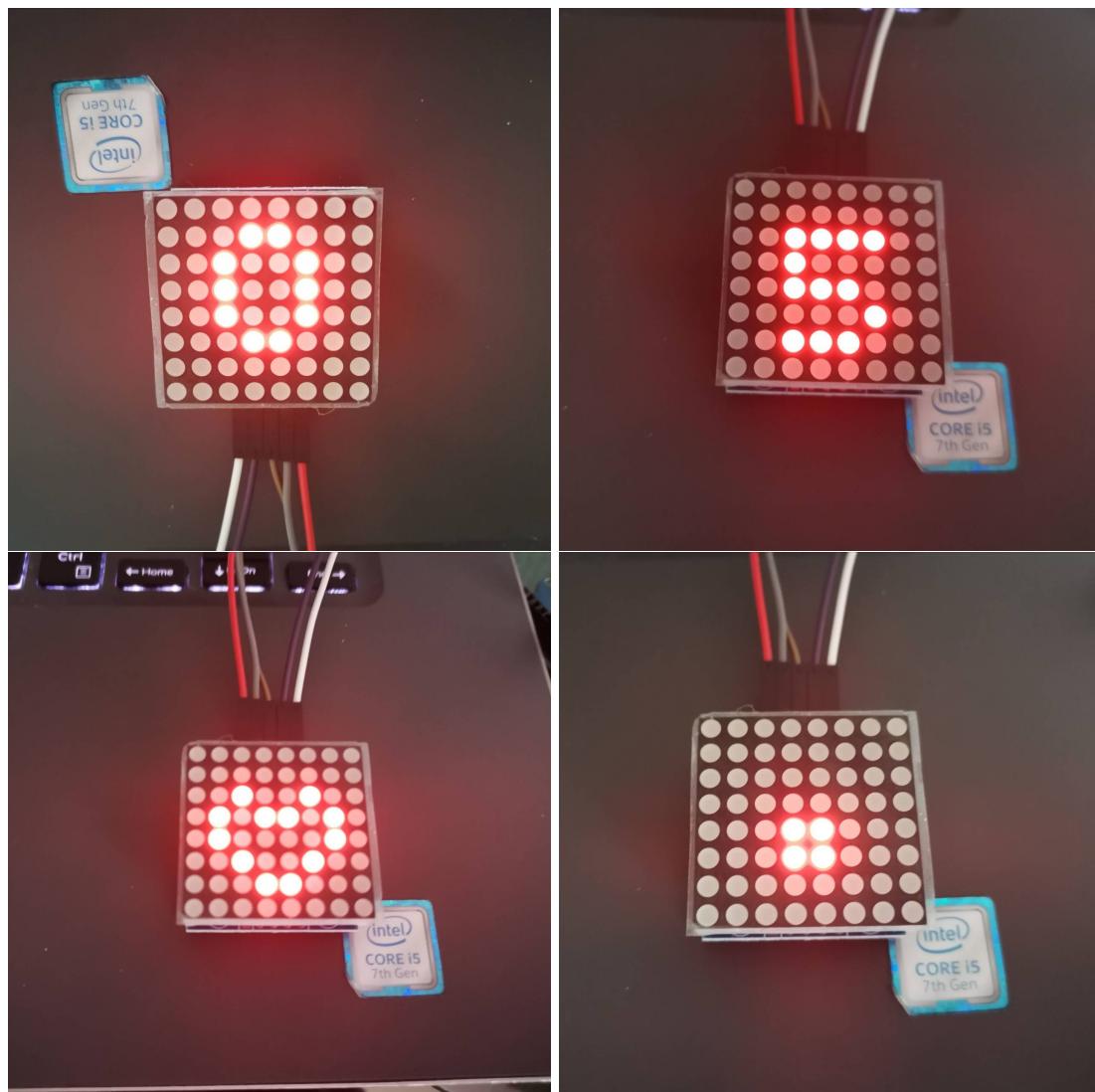


图 10.3: 串口监视器中打印出的信息

我们在上面一栏中每输入其中一个数字，就可以在显示器上看到对应的图像，如下图所示：



实验十一 串行 LCD 动态显示屏实验

11.1 实验目的

通过 Arduino 控制 LCD1602 液晶显示屏显示字符。

11.2 LCD 显示模块

LCD1602 是一种专门用于显示字母、数字和符号的字符 LCD 模块。它被广泛应用于工业，比如电子钟、温度显示器。市场上的字符液晶显示器大多是基于 HD44780 字符的 LCD 芯片，其控制原理完全相同。“1602” 表示每行 16 个字符，总共 2 行。带了转接板的 LCD1602 显示屏，使用了 IIC 接口，这节省了许多的 I/O 口。通过 LiquidCrystal 类库提供的 API，我们可以很方便的使用 1602LCD 显示英文字母与一些符号。在使用 1602LCD 前，我们需要将其连接到 Arduino 上。通过模块上的电位器，我们可以调节 LCD 显示器的对比度。通过设置跳线还可以设置地址: 0x20-0x27，从而使 Arduino 能控制多块 LCD 1602。



图 11.1: LCD1602 模块示意图

11.3 实验用品

Arduino UNO 开发板、USB 数据线、LCD1602 显示模块、若干杜邦线。

11.4 实验步骤

- (1) 按照如图11.2所示的方式完成接线;
- (2) 扫描 I2C 地址;
- (3) 烧录显示字符程序，观察显示情况。

11.5 接线图

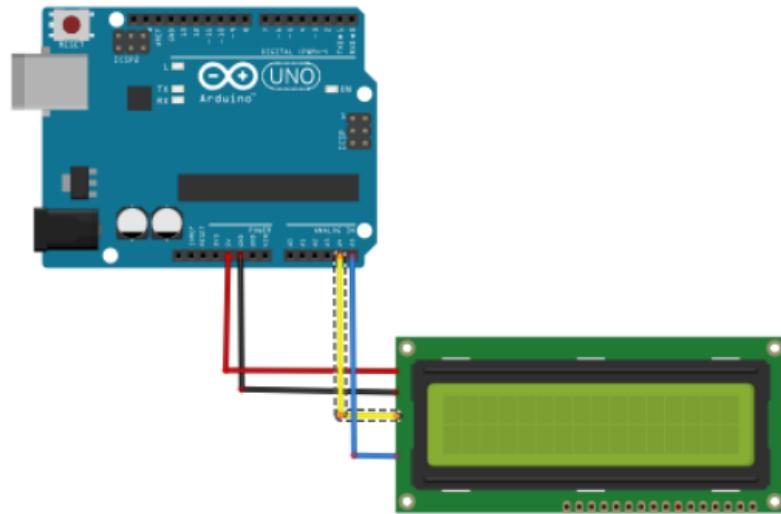


图 11.2: 实验接线图

11.6 实验程序

```
1 // Serial_LCD1602_Display.ino
2 // Date: 2020/05/03
3
4 #include <Wire.h>
5 #include "LiquidCrystal_I2C.h"
6
7 // set the LCD address to 0x27 for a 16 chars and 2 line display
8 LiquidCrystal_I2C lcd(0x27, 16, 2);
9
10 void setup()
11 {
12     lcd.init();           // initialize the lcd
13     // Print a message to the LCD.
14     lcd.backlight();
15     lcd.setCursor(2, 0);    // go to start of 2nd line
16     lcd.print("Hello, world!");
17     lcd.setCursor(4, 1);    // go to start of 2nd line
18     lcd.print("keywish");
19 }
20
21 void loop(){}
```

11.7 实验结果

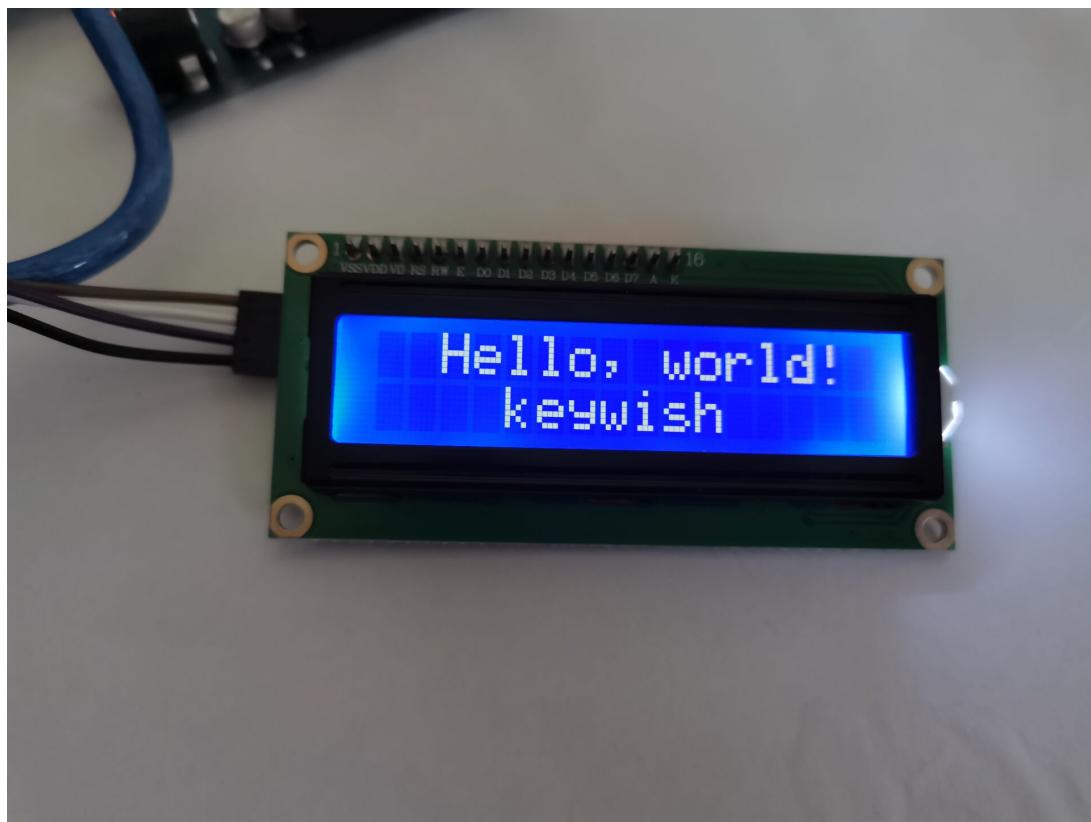


图 11.3: 显示结果

实验十二 游戏操纵五向按键模块实验

12.1 实验目的

通过操纵杆控制 LED，在操纵杆上下左右移动以及按下操纵杆时，分别确保相应的 LED 指示灯亮起。

12.2 游戏操纵五向按键模块

操纵杆结构非常简单，它包含一个触摸按钮 (z 轴) 和两个电位器 (x 轴和 y 轴)。操纵杆根据两个触点控制运动，其中一个触点向左和向右，另一个向上和向下。操纵杆移动决定了触点的位置，就像地球的纬度和经度一样，不同的位置对应不同的电压，然后控制器可以通过 AD 传感器读取不同的电压值，从而识别特定的远程位置。

模拟信号从上方操纵杆的 V_{R_x}, V_{R_y} (x 和 y 轴) 引脚输入， V_{R_x} 的值从 0 到 1023 表示从左到右的位置，而 V_{R_y} 的值从 0 到 1023 分别表示从上到下的位置，如果没有按下按钮，则两个值都是 512，这是中间值。

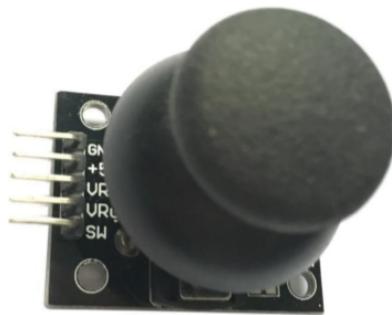


图 12.1: 游戏操纵五向按键模块示意图

12.3 实验用品

Arduino UNO 开发板、USB 数据线、面包板、游戏操纵五向按键模块、LED 灯 $\times 5$ 、 220Ω 电阻 $\times 5$ 、若干杜邦线。

12.4 实验步骤

- (1) 按照如图12.2所示的方式完成接线;
- (2) 打开 Arduino IDE，输入程序;
- (3) 检查无误后上传程序;
- (4) 操作遥杆，观察 LED 指示灯的变化。

12.5 接线图

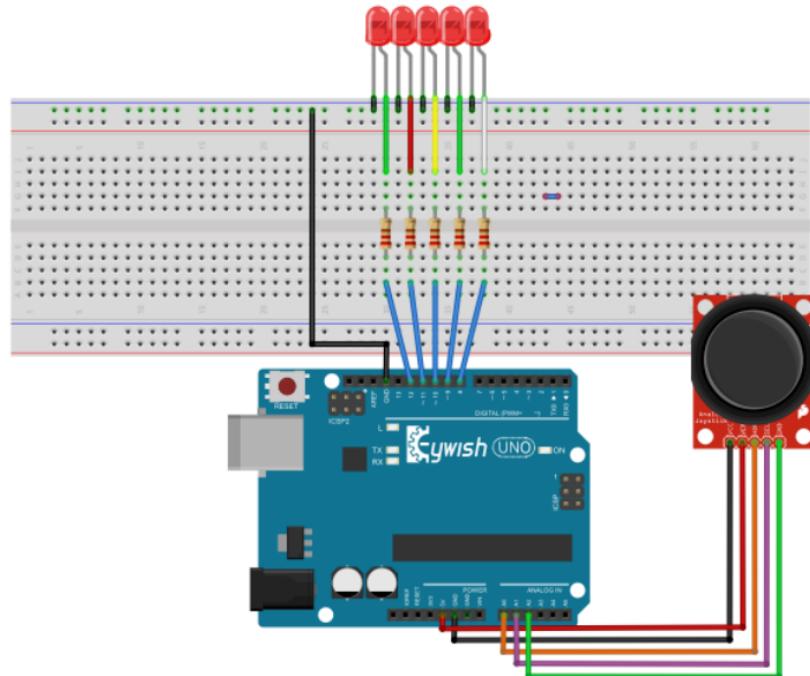


图 12.2: 实验接线图

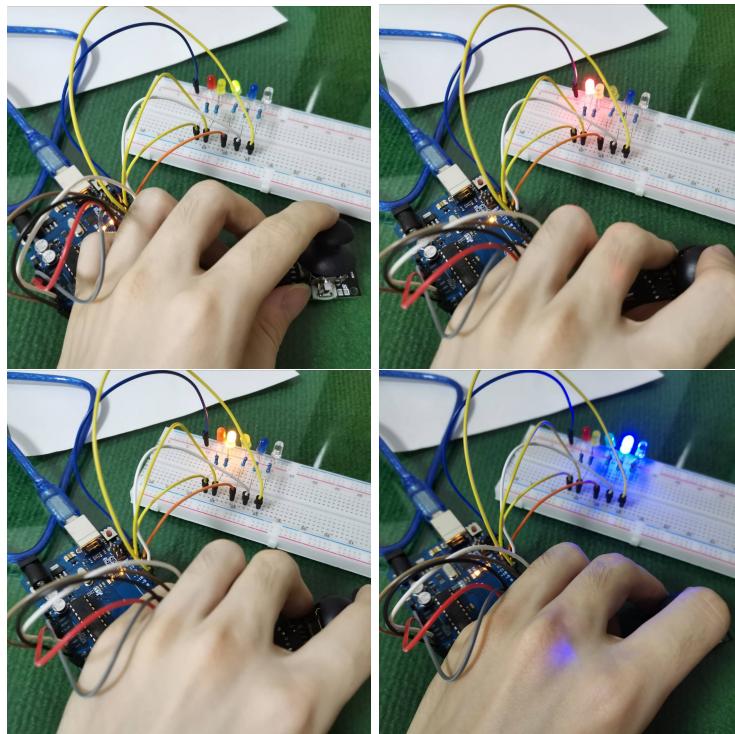
12.6 实验程序

```
1 // Joystick.ino
2 // Date: 2020/05/03
3
4 #define JOYSTICK_X      A0
5 #define JOYSTICK_Y      A1
6 #define JOYSTICK_SW     A2
7
8 #define LED_ENTER       8  //enter
9 #define LED_LEFT        9  //left
10 #define LED_UP          10 //up
11 #define LED_RIGHT       11 //right
```

```
12 #define LED_DOWN    12 //down
13
14 void setup()
15 {
16     pinMode(JOYSTICK_X, INPUT);
17     pinMode(JOYSTICK_Y, INPUT);
18     pinMode(JOYSTICK_SW, INPUT_PULLUP);
19     pinMode(LED_ENTER, OUTPUT);
20     pinMode(LED_LEFT, OUTPUT);
21     pinMode(LED_UP, OUTPUT);
22     pinMode(LED_RIGHT, OUTPUT);
23     pinMode(LED_DOWN, OUTPUT);
24     Serial.begin(9600);
25 }
26
27 void loop()
28 {
29     int value_x = analogRead(JOYSTICK_X);
30     int value_y = analogRead(JOYSTICK_Y);
31     int value_sw = digitalRead(JOYSTICK_SW);
32
33     if(value_x==0)
34     {
35         digitalWrite(LED_RIGHT, LOW);
36         digitalWrite(LED_LEFT, HIGH);
37         Serial.print("left");
38     }
39     else if(value_x==1023)
40     {
41         digitalWrite(LED_LEFT, LOW);
42         digitalWrite(LED_RIGHT, HIGH);
43         Serial.print("right");
44     }
45     else
46     {
47         digitalWrite(LED_LEFT, LOW);
48         digitalWrite(LED_RIGHT, LOW);
49     }
50
51     if(value_y==0)
52     {
53         digitalWrite(LED_DOWN, LOW);
54         digitalWrite(LED_UP, HIGH);
55         Serial.print("up");
56     }
57     else if(value_y==1023)
58     {
59         digitalWrite(LED_UP, LOW);
```

```
61     digitalWrite(LED_DOWN, HIGH);
62     Serial.print("down");
63 }
64 else
65 {
66     digitalWrite(LED_UP, LOW);
67     digitalWrite(LED_DOWN, LOW);
68 }
69
70 if(value_sw==0)
71 {
72     digitalWrite(LED_ENTER, HIGH);
73     Serial.print("enter");
74 }
75 else
76 {
77     digitalWrite(LED_ENTER, LOW);
78 }
79
80 delay(100);
81 }
```

12.7 实验结果



实验十三 魔术光环模块实验

13.1 实验目的

使用 Arduino 控制魔术光环模块上 LED 灯的亮灭。

13.2 魔术光环模块

魔术光环模块是由两部分组成，分别是水银开关和 LED 灯。当模块上的水银开关闭合时，其上面的 LED 灯会被点亮；反之则 LED 灯熄灭。

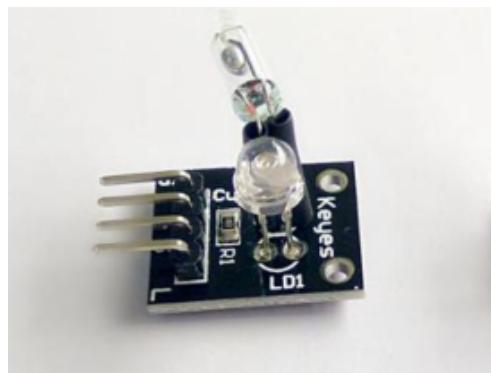


图 13.1: 魔术光环模块示意图

13.3 实验用品

Arduino UNO 开发板、USB 数据线、魔术光环模块、若干杜邦线。

13.4 实验步骤

- (1) 按照如图13.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 晃动模块，观察 LED 灯的变化

13.5 接线图

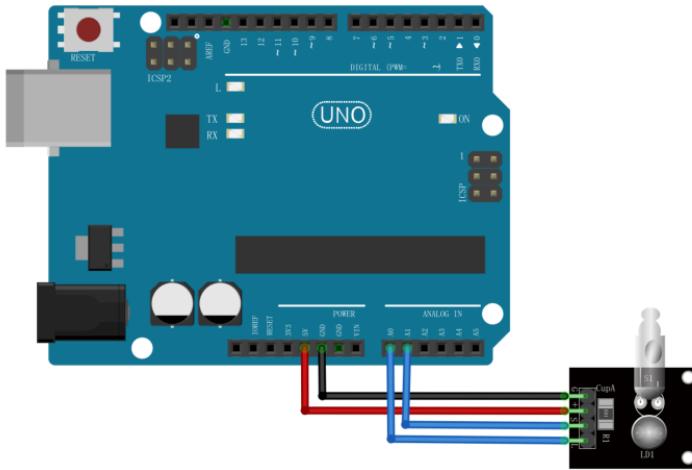


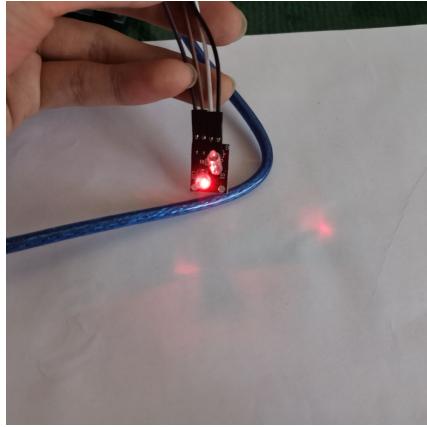
图 13.2: 实验接线图

13.6 实验程序

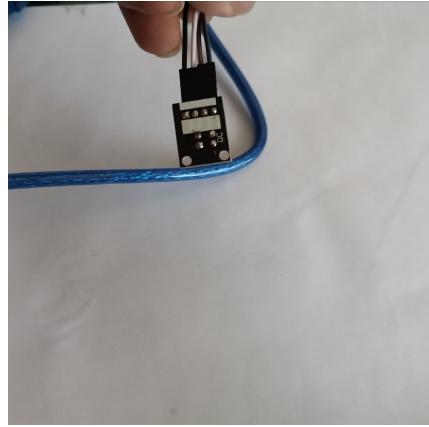
```
1 // MagicLED.ino
2 // Date: 2020/05/03
3
4 const int LED_PIN=A0;
5 const int SWITCH_PIN=A1;
6
7 void setup()
8 {
9     pinMode(LED_PIN, OUTPUT);
10    pinMode(SWITCH_PIN, INPUT);
11    Serial.begin(9600);
12 }
13
14 void loop()
15 {
16     int value = digitalRead(SWITCH_PIN);
17     Serial.println(value);
18     if(value==0)
19     {
20         digitalWrite(LED_PIN, HIGH);
21     }
22     else
23     {
24         digitalWrite(LED_PIN, LOW);
25     }
26 }
```

13.7 实验结果

如图13.3所示，当程序烧录成功后，正置模块，可以观察到 LED 灯亮起，倒置模块，可以观察到 LED 灯熄灭。



(a)



(b)

图 13.3: 魔术光环模块实验结果示意图: (a) 将模块正置时, LED 灯亮起; (b) 将模块倒置时, LED 灯熄灭

实验十四 灰度传感器模块实验

14.1 实验目的

- (1) 学习灰度传感器的工作原理；
- (2) 了解如何使用 Arduino 获取灰度传感器模块的值；
- (3) 了解灰度传感器的应用场景。

14.2 灰度传感器

灰度传感器是模拟传感器，有一只发光二极管和一只光敏电阻，安装在同一面上。灰度传感器利用不同颜色的检测面对光的反射程度不同（反射回的光的强度不同），导致光敏电阻检测到不同检测面返回的光时，其阻值也不同的原理进行颜色深浅检测。在有效的检测距离内，发光二极管发出白光，照射在检测面上，检测面反射部分光线，光敏电阻检测此光线的强度并将其转换为机器人可以识别的信号。它输出的是连续的模拟信号。



图 14.1: 灰度传感器模块示意图

14.3 主要工作参数

工作电压	3.5-5.5V
工作电流	<20mA
工作温度	-10-70°C
探测分辨率	10%
PCB 尺寸	24mm × 21mm

表 14.1: 灰度传感器模块的主要工作参数

14.4 实验原理

通过灰度传感器检测反光面的颜色，利用 Arduino 将获取的颜色值通过控制 LCD 显示屏显示出来。

14.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、灰度传感器模块、LCD 显示屏模块、若干杜邦线。

14.6 实验步骤

- (1) 按照如图14.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将不同颜色的平面对准灰度传感器，观察 LCD 上显示的灰度范围。

14.7 接线图

- 灰度传感器模块：OUT 接 AO-S，GND 接 AO-G，VCC 接 AO-V；
- LCD 显示模块：GND 接 A4-G，VCC 接 A4-V，SDA 接 A4-S，SCL 接 A5-S。

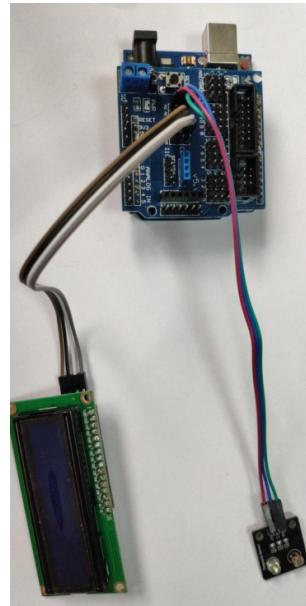


图 14.2: 实验接线图

14.8 实验程序

```
1 // GrayLevel.ino
2 // Date: 2020/05/03
3
4 #include <Wire.h>
5 #include "LiquidCrystal_I2C.h"
6
7 int analogPin = A0;
8 LiquidCrystal_I2C lcd(0x27, 16, 2);
9
10 void setup()
11 {
12     lcd.init();
13     lcd.backlight();
14     pinMode(analogPin, INPUT);
15     Serial.begin(9600);
16 }
17
18 void loop() {
19     float data = analogRead(analogPin);
20     Serial.println(data);
21     lcd.setCursor(0, 0);
22     lcd.print("The range is: ");
23     lcd.setCursor(0, 1);
24     lcd.print(data, 2);
25     delay(200);
26 }
```

14.9 实验结果



图 14.3: 灰度传感器模块实验结果示意图: (a) 正常情况下的示数; (b) 用未亮起的 iPad 屏幕遮挡在灰度传感器前的示数。

实验十五 触摸开关模块实验

15.1 实验目的

- (1) 学习触摸开关模块的工作原理；
- (2) 基于 Arduino 实现利用触摸开关来控制 LED 灯。

15.2 触摸开关模块

触摸开关模块是一个基于触摸检测 IC(TTP223B) 的电容式点动型触摸开关模块。常态下，模块输出低电平，模式为低功耗模式；当用手指触摸相应位置时，模块会输出高电平，模式切换为快速模式；当持续 12 秒没有触摸时，模式又切换为低功耗模式。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。

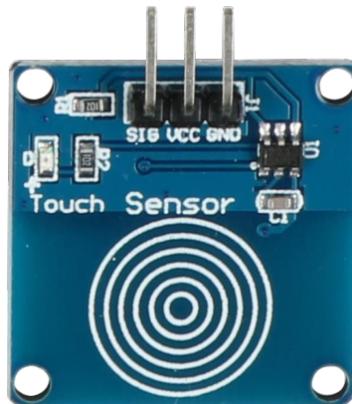


图 15.1: 触摸开关模块示意图

15.3 主要工作参数

工作电压	3.3-5V
工作电流	<20mA
重量	2g
响应时间	220ms

表 15.1: 触摸开关模块的主要工作参数

15.4 实验原理

用手指触摸触摸开关模块，当 Arduino 检测到触摸开关模块输出高电平时，它会控制 LED 灯亮，否则 LED 灯灭。

15.5 实验用品

Arduino UNO 开发板、USB 数据线、触摸开关模块、LED 模块、若干杜邦线。

15.6 实验步骤

- (1) 按照如图15.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 分别用手指触摸和离开触摸开关，观察 LED 灯模块的变化。

15.7 接线图

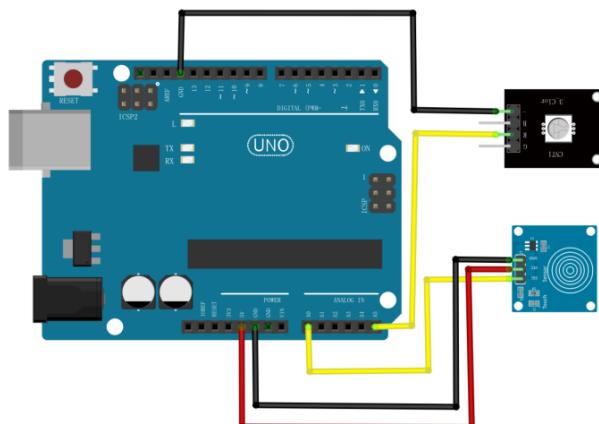


图 15.2: 实验接线图

15.8 实验程序

```
1 // FigureSensor.ino
2 // Date: 2020/05/03
3
4 int Led_pin=A5;
5 int Sensor_pin=A0;
6
7 void setup()
8 {
9     pinMode(Led_pin, OUTPUT);
10    pinMode(Sensor_pin, INPUT);
11    Serial.begin(9600);
12 }
13
14 void loop()
15 {
16     int value=digitalRead(Sensor_pin);
17     Serial.println(value);
18     if(value==1)
19     {
20         digitalWrite(Led_pin, HIGH);
21     }
22     else
23         digitalWrite(Led_pin, LOW);
24 }
25 }
```

15.9 实验结果

如图15.3所示，当程序烧录成功后，触摸开关时，可以观察到 LED 灯亮起；未触摸开关时，可以观察到 LED 灯熄灭。

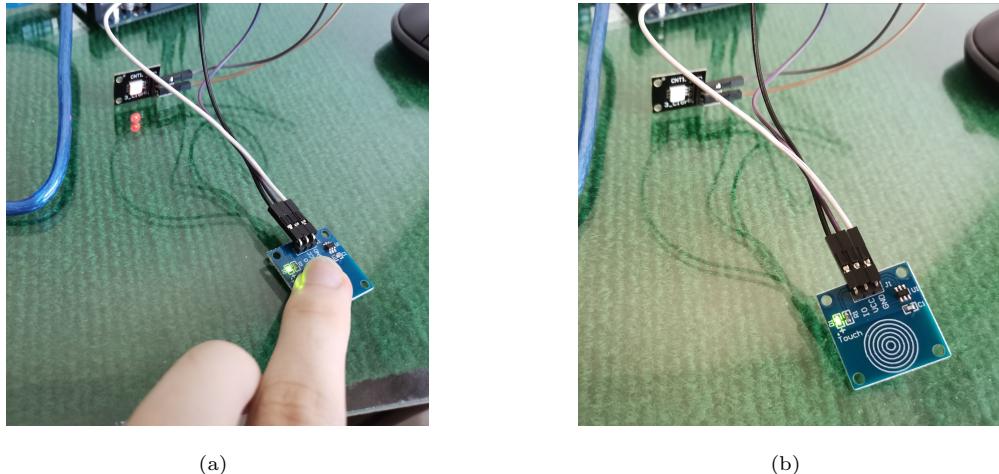


图 15.3: 触摸开关模块实验结果示意图: (a) 未触摸开关时, LED 灯不亮; (b) 触摸开关时, LED 灯亮起

实验十六 光敏电阻模块实验

16.1 实验目的

- (1) 学习光敏电阻的原理；
- (2) 基于 Arduino 实现利用光敏电阻来控制 LED 灯的亮度。

16.2 光敏电阻

光敏电阻 (photoresistor or light-dependent resistor, 后者缩写为 ldr) 或光导管 (photoconductor)，常用的制作材料为硫化镉，另外还有硒、硫化铝、硫化铅和硫化铋等材料。这些制作材料具有在特定波长的光照射下，其阻值迅速减小的特性。这是由于光照产生的载流子都参与导电，在外加电场的作用下作漂移运动，电子奔向电源的正极，空穴奔向电源的负极，从而使光敏电阻器的阻值迅速下降。

通常来说，当入射光强度上升时，光敏电阻的阻值会降低；当入射光强度减弱时，其阻值会增加。光敏电阻器一般用于光的测量、光的控制和光电转换 (将光的变化转换为电的变化)。



图 16.1: 光敏电阻模块示意图

16.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	33×14mm
输出形式	模拟量输出 / 数字量输出

表 16.1: 光敏电阻模块的主要工作参数

16.4 实验原理

- (1) 光敏电阻模块对环境光线最敏感，一般用来检测周围环境的光线的亮度，触发单片机或继电器模块等；
- (2) 模块在环境光线亮度达不到设定阈值时，DO (数字端口) 端输出高电平，当外界环境光线亮度超过设定阈值时，DO 端输出低电平；
- (3) DO 输出端可以与单片机直接相连，通过单片机来检测高低电平，由此来检测环境的光线亮度改变；
- (4) DO 输出端可以直接驱动继电器模块，由此可以组成一个光控开关；
- (5) 模拟量输出 AO 可以和 AD 模块相连，通过 AD 转换，可以获得环境光强更精准的数值。

16.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、光敏电阻、LED 模块、若干杜邦线。

16.6 实验步骤

- (1) 按照如图16.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 调节光敏电阻模块的电位器（蓝色的突起），使其检测阈值在合适的范围内。然后改变环境光线的强度，观察 LED 灯模块的变化。

16.7 接线图

- 声音传感器模块：A0 接 A0-S，G 接 A0-G，+ 接 A0-V；
- LED 模块：- 接 11-G，G 接 11-S。

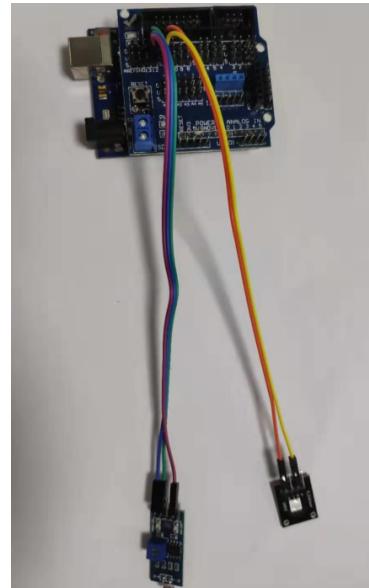


图 16.2: 实验接线图

16.8 实验程序

```
1 // photosensitive.ino
2 // Date: 2020/05/02
3
4 int AD_pin = A0 ;
5 int LED_pin = 11 ;
6 int value = 0 ;
7 float voltage = 0.0 ;
8
9 void setup()
10 {
11     pinMode(LED_pin, OUTPUT);
12     pinMode(AD_pin, INPUT);
13     Serial.begin(9600);
14
15 void loop()
16 {
17     int value = analogRead(AD_pin);
18     int voltage = ((float)value) / 1023 ;
19     value = (int)(voltage*256) ;           //convert voltage to value
20     Serial.println(value);
21     analogWrite(LED_pin, value);
22
23 }
```

16.9 实验结果

如图16.3所示，当程序烧录成功后，可以观察到：环境光线强度较小时，LED 灯亮度较大；而环境光线强度较大时，LED 亮度较小。

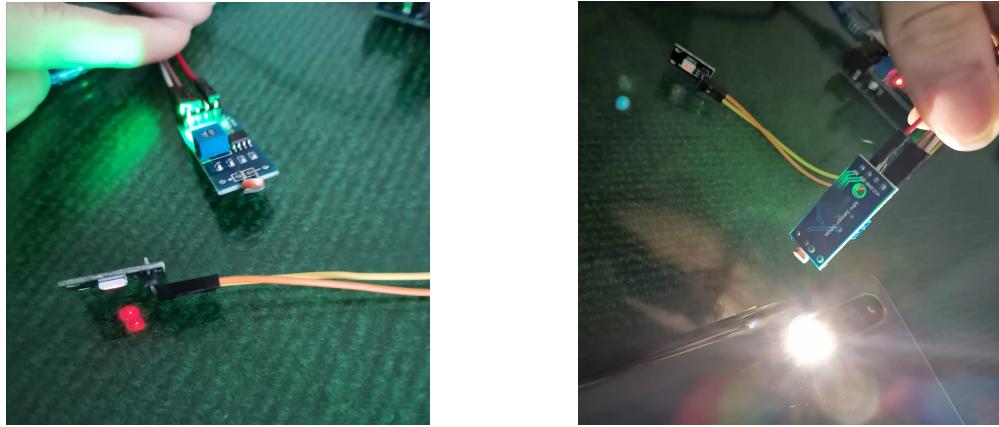


图 16.3: 光敏电阻模块实验结果示意图: (a) 环境光线强度较小时, LED 灯亮度较大; (b) 环境光线强度较大时, LED 亮度较小。

实验十七 火焰传感器模块实验

17.1 实验目的

- (1) 学习火焰传感器的原理；
- (2) 基于 Arduino 进行相关情景模拟。

17.2 火焰传感器

火焰传感器 (flame sensor) 可以用来探测火源或其它波长在760至1100纳米范围内的光源。火焰的热辐射具有离散光谱的气体辐射和连续光谱的固体辐射。不同燃烧物的火焰辐射强度、波长分布有所差异，但总体来说，其对应火焰温度的近红外波长域及紫外光域有很大的辐射强度，火焰传感器就是根据这种特性制作而成。火焰传感器利用红外线对火焰非常敏感的特点，使用特制的红外线接收管检测火焰，然后把火焰的亮度转化为高低变化的电平信号，输入到中央处理器中，中央处理器根据信号的变化做出相应的程序处理。

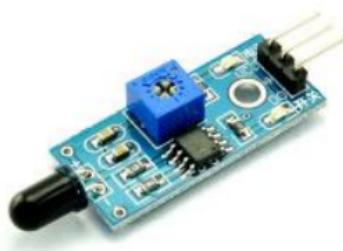


图 17.1: 火焰传感器模块示意图

17.3 主要工作参数

工作电压	3.3-5V
检测范围	波长 760-1100 纳米范围内的光源
探测角度	60°
输出形式	数字开关量输出 (0和1)

表 17.1: 火焰传感器模块的主要工作参数

17.4 实验原理

本实验所用火焰传感器模块输出的是数字信号，当没有检测到火焰的时候，输出高电平，当检测到火焰时，输出低电平。

17.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、火焰传感器模块、有源蜂鸣器、LED 模块、若干杜邦线。

17.6 实验步骤

- (1) 按照如图17.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 调节火焰传感器模块的电位器（蓝色的突起），使其检测阈值在合适的范围内，然后用打火机等在火焰传感器附近打火，观察蜂鸣器与 LED 灯的变化。

17.7 接线图

- 火焰传感器模块：DO 接 A0-S，GND 接 A0-G，VCC 接 A0-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。

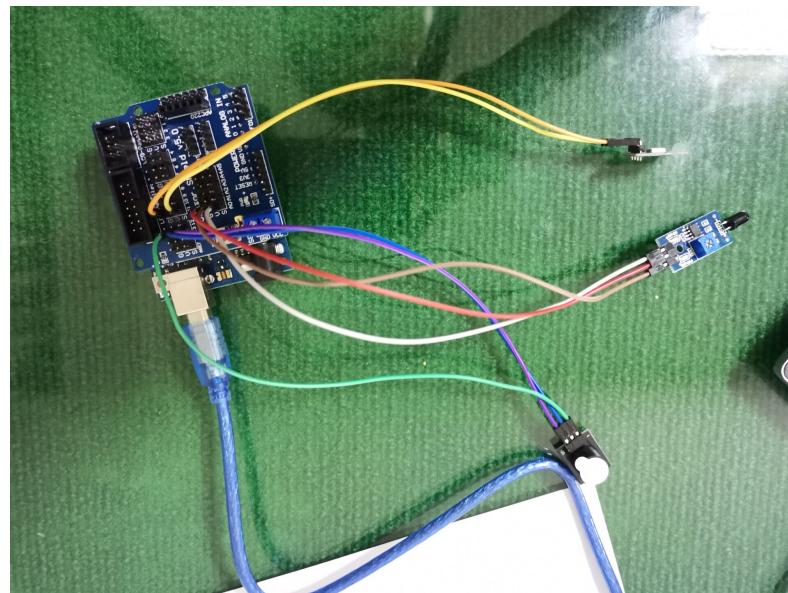


图 17.2: 实验接线图

17.8 实验程序

为提高报警的准确性，尽量减少偶发因素带来的误报，我们在此设置一个平滑值（在下面的程序中，它是变量 smooth_value，我们在本实验中将这个值设为 5）。因此，当输出的信号连续五次高于阈值时，我们认为有火焰，此时蜂鸣器发声且 LED 灯亮起；若小于阈值，蜂鸣器保持静音且 LED 熄灭。

```
1 // Flame_sensor.ino
2 // Date: 2020/04/30
3
4 int fire_pin = A0;
5 int buzzer_pin = 13;
6 int LED_pin = 11;
7
8 int count = 0;
9 int smooth_value = 5;
10
11 void setup()
12 {
13     pinMode(buzzer_pin, OUTPUT);
14     pinMode(LED_pin, OUTPUT);
15     pinMode(fire_pin, INPUT);
16     Serial.begin(9600);
17     digitalWrite(buzzer_pin, LOW);
18 }
19
20 void loop()
21 {
22     int value = digitalRead(fire_pin);
23     Serial.println(value);
24
25     if (value == 0)
26     {
27         count++;
28     }
29     else
30     {
31         count = 0;
32     }
33
34     if (count >= smooth_value)
35     {
36         digitalWrite(buzzer_pin, HIGH);
37         digitalWrite(LED_pin, HIGH);
38     }
39     else
40     {
41         digitalWrite(buzzer_pin, LOW);
```

```

42         digitalWrite(LED_pin, LOW);
43     }
44
45     // delay(500);
46 }
```

17.9 实验结果

如图17.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们在火焰传感器模块的红外线接收管前用打火机打着火焰时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 A0 引脚的读数值，模块在未检测到火焰时时会输出 1，检测到火焰时会输出 0，如图17.4所示。

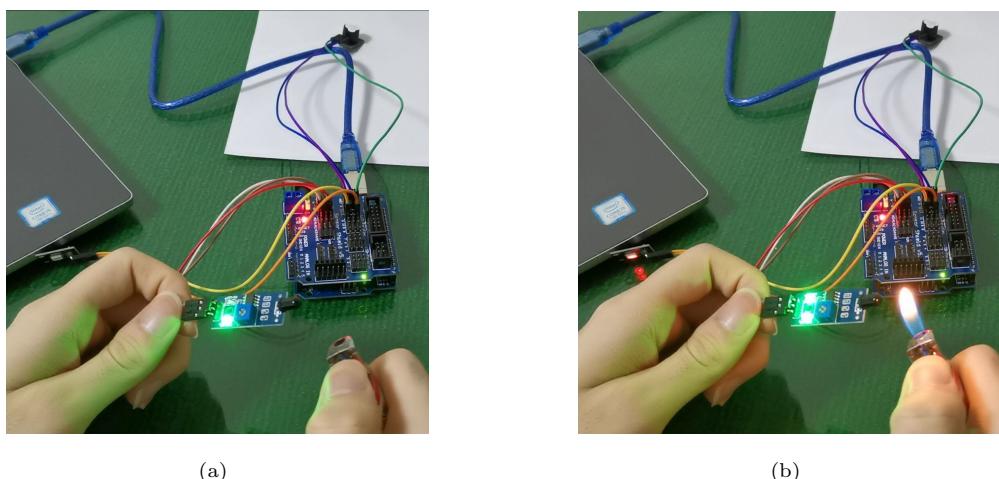


图 17.3: 火焰传感器模块实验结果示意图：(a) 模块未检测到火焰时，LED 灯不亮且蜂鸣器不发出声音；(b) 模块检测到火焰时，LED 亮起且蜂鸣器发出响声。



图 17.4: 串口监视器中不断打印出的 A0 引脚的读数值

实验十八 LM35 温度传感器实验

18.1 实验目的

基于 Arduino 与 LM35 温度传感器搭建测温系统。

18.2 LM35 温度传感器

温度传感器是指能感受温度并转换成可用输出信号的传感器。温度传感器是温度测量仪表的核心部分，品种繁多。温度传感器对于环境温度的测量非常准确，广泛应用于农业、工业、车间、库房等领域。

LM35 是由 National Semiconductor 所生产的温度传感器，其输出电压为摄氏温标。它是一种得到广泛使用的温度传感器。由于它采用内部补偿，所以输出可以从 0°C 开始，LM35 有多种不同封装型式。在常温下，LM35 不需要额外的校准处理即可达到 $\pm 0.25^\circ\text{C}$ 的准确率。LM35 的输出电压与摄氏温度成线性比例。因此，LM35 远远优于绝对标度线性温度传感器。LM35 系列传感器重现性高，输出阻抗低。线性输出和内部校准精度使读出或控制电路接口易于使用。

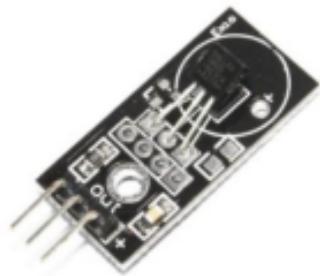


图 18.1: LM35 温度传感器示意图

18.3 主要工作参数

工作电压	3.3-5V
低功耗	<60 μ A
温度测量范围	0-100°C
温度测量精度	0.5°C

表 18.1: LM35 温度传感器模块的主要工作参数

18.4 实验原理

温度传感器电路将测量到的温度信号转换成电压信号输出到信号放大电路，与温度值对应的电压信号经放大后输出至 A/D 转换电路，把电压信号转换成数字量输送至 Arduino 系统，Arduino 系统根据显示需要对数字量进行处理，再送至温度显示系统进行显示。

18.5 实验用品

Arduino UNO 开发板、USB 数据线、面包板、LM35 温度传感器、若干跳线。

18.6 实验步骤

- (1) 按照如图18.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 打开串口监视器，查看输出信息。

18.7 接线图

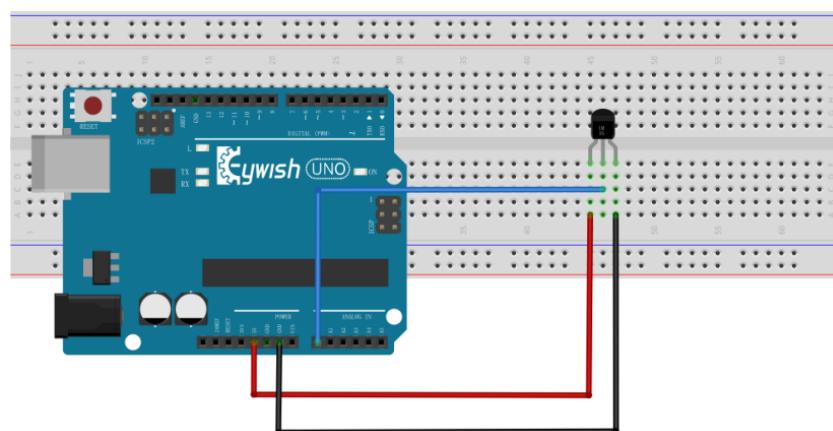


图 18.2: 实验接线图

18.8 实验程序

```
1 // LM35.ino
2 // datee: 2020/05/03
3
4 int Temp_Pin = A0;
5
6 void setup()
7 {
8     Serial.begin(115200);
9 }
10
11 void loop()
12 {
13     int value = analogRead(Temp_Pin);
14     float voltage = ((float)value) / 1023 ;
15     voltage = voltage * 5 ;
16     int date = voltage * 100;
17     Serial.print("Current Temp : ");
18     Serial.print(date);
19     Serial.println("C");
20     delay(500);
21 }
```

18.9 实验结果

```
Current Temp : 28C
Current Temp : 28C
Current Temp : 29C
Current Temp : 28C
Current Temp : 30C
Current Temp : 30C
Current Temp : 30C
Current Temp : 30C
Current Temp : 31C
Current Temp : 31C
Current Temp : 29C
Current Temp : 30C
Current Temp : 30C
```

图 18.3: LM35 温度传感器模块实验结果

实验十九 温湿度传感器模块实验

19.1 实验目的

- (1) 学习温湿度传感器的工作原理；
- (2) 基于 Arduino 获取温湿度传感器模块的值并将该值通过 LCD1602 显示屏显示出来。

19.2 DHT11 温湿度传感器

DHT11 包括一个电阻式测湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。通过单片机等微处理器简单的电路连接就能够实时的采集本地湿度和温度。DHT11 与单片机之间能采用简单的单总线进行通信，仅仅需要一个 I/O 口。传感器内部湿度和温度数据 40Bit 的数据一次性传给单片机，数据采用校验和方式进行校验，有效的保证数据传输的准确性。DHT11 功耗很低，5V 电源电压下，工作平均最大电流 0.5mA，非常适用于对精度和实时性要求不高的温湿度测量场合。

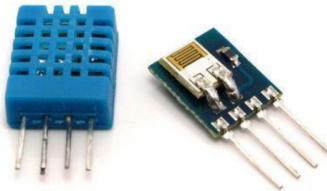


图 19.1: DHT11 温湿度传感器模块示意图

该模块共有四个引脚。其中 VDD 连接电源正极 (3-5V)；GND 接地；DATA 为数据端，可直接接单片机的 I/O 口，为提高稳定性，需要在数据端和电源正之间接一只 5.1K 的上拉电阻；NC 为空脚，此管脚悬空不用。

19.3 主要工作参数

工作电压	3.5-5.5V
工作电流	0.5mA
湿度测量范围	20-90%RH
温度测量范围	0-50°C
湿度分辨率	1%RH
温度分辨率	1°C
采样时间	1s

表 19.1: DHT11 的主要工作参数

19.4 实验步骤

- (1) 按照如图19.2所示的方式完成接线;
- (2) 打开 Arduino IDE，输入程序;
- (3) 检查无误后上传程序;
- (4) 观察 LCD 显示屏上的显示结果。

19.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、DHT 温湿度传感器模块、LCD1602 显示模块、若干杜邦线。

19.6 接线图

- 土壤湿度传感器模块: S 接 A4-S, - 接 A4-G, + 接 A4-V;
- LCD 显示模块: GND 接 A4-G, VCC 接 A4-V, SDA 接 A4-S, SCL 接 A5-S。

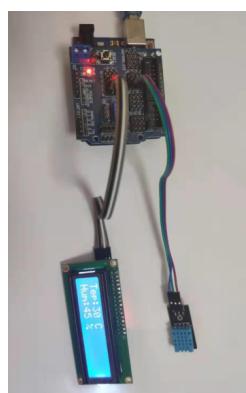


图 19.2: 实验接线图

19.7 实验程序

```
1 // Temperature_Humidity_Sensor.ino
2 // Date: 2020/05/02
3
4 #include <Wire.h>
5 #include "dht11.h"
6 #include "LiquidCrystal_I2C.h"
7
8 #define DHT11PIN 8
9 dht11 DHT11;
10 LiquidCrystal_I2C lcd(0x27, 16, 2);
11
12 void setup() {
13     pinMode(DHT11PIN, OUTPUT);
14     lcd.init();
15     lcd.backlight();
16     Serial.begin(9600);
17 }
18
19 void loop() {
20     int chk = DHT11.read(DHT11PIN);
21     lcd.setCursor(0, 0);
22     lcd.print("Temp: ");
23     lcd.print((float)DHT11.temperature, 2);
24     lcd.print("C");
25     lcd.setCursor(0, 1);
26     lcd.print("Hum: ");
27     lcd.print((float)DHT11.humidity, 2);
28     lcd.print("%");
29     delay(200);
30 }
```

19.8 实验结果



图 19.3: LCD 显示屏上打印出的当前环境的温湿度信息

实验二十 红外循迹模块实验

20.1 实验目的

- (1) 学习红外循迹模块的工作原理；
- (2) 基于 Arduino 通过获取红外循迹模块的值来控制 LED 灯或者有源蜂鸣器。

20.2 红外循迹模块

红外循迹模块本质上是一个红外线收发装置，它具有一对红外线发射与接收管，发射管发射出一定频率的红外线，当其遇到反射面时会反射回来被接收管接收，经过比较器电路处理之后，指示灯会亮起，同时信号输出接口输出数字信号（一个低电平信号）。

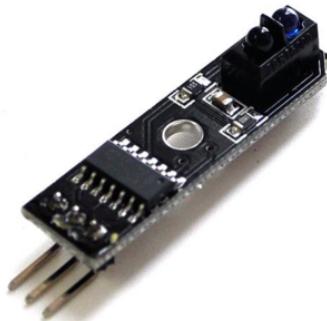


图 20.1: 红外循迹模块示意图

20.3 主要工作参数

工作电压	3.3-5V
检测距离	0-3cm
模拟量电压输出	0-5V
数字开关量输出	0和1

表 20.1: 红外寻迹模块的主要工作参数

20.4 实验原理

利用红外循迹的工作特性，当红外循迹模块检测到黑线时，LED 亮起且蜂鸣器响；如果没有检测到黑线，则 LED 熄灭且蜂鸣器不响。

20.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、红外循迹模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

20.6 实验步骤

- (1) 按照如图20.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 分别将红外循迹模块对准白色色块和黑色色块，观察 LED 灯与蜂鸣器的变化。

20.7 接线图

- 红外循迹模块：OUT 接 12-S，GND 接 12-G，VCC 接 12-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。

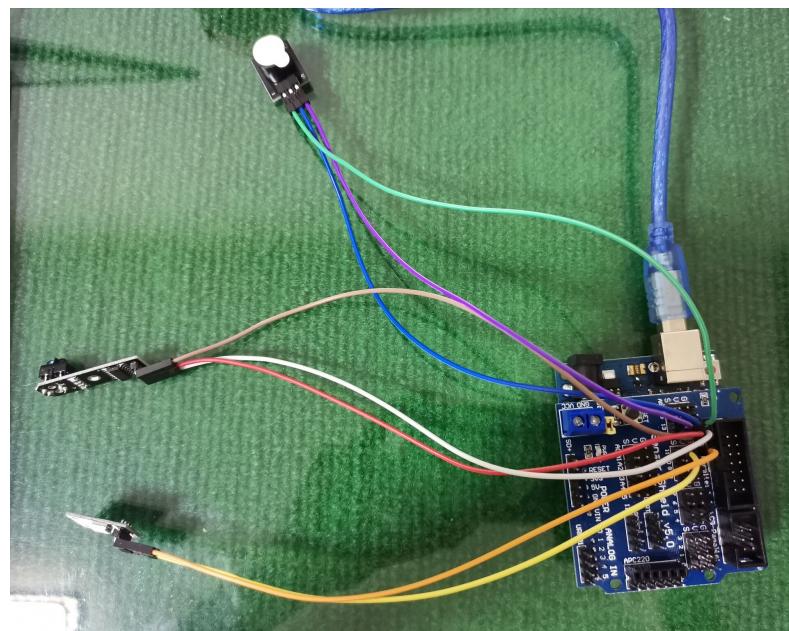


图 20.2: 实验接线图

20.8 实验程序

```
1 // InfraredTracking.ino
2 // Date: 2020/04/28
3
4 int Led_pin = 11;
5 int Buzzer_pin = 13;
6 int Sensor_pin = 12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value = digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==1)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 // delay(500);
31 }
```

20.9 实验结果

如图20.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们用手指挡在红外循迹模块的发射管与接收管之前时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 12 引脚的读数值，模块前无遮挡时会输出 0，而当用手指挡在模块前时会输出 1，如图20.4所示。

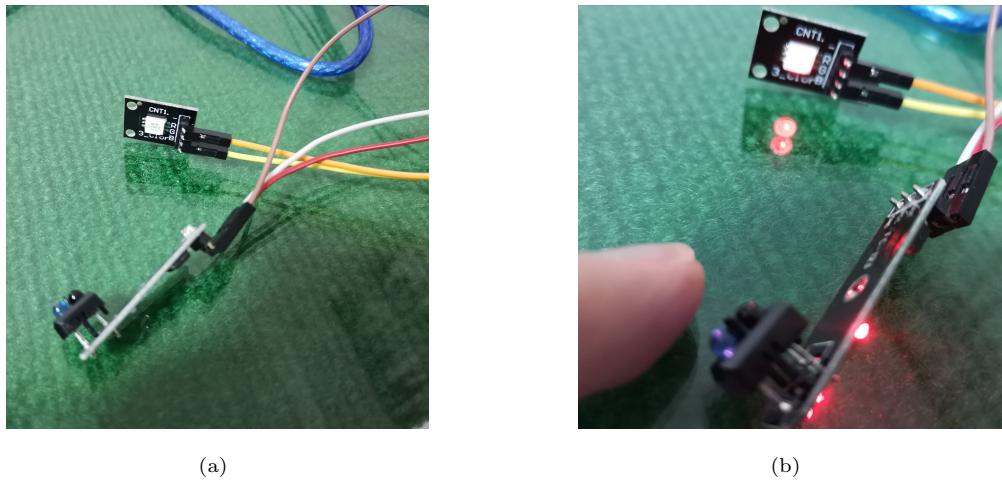


图 20.3: 红外循迹模块实验结果示意图: (a) 模块前无遮挡时, LED 灯不亮且蜂鸣器不发出声音; (b) 用手挡在模块前时, LED 亮起且蜂鸣器发出响声。

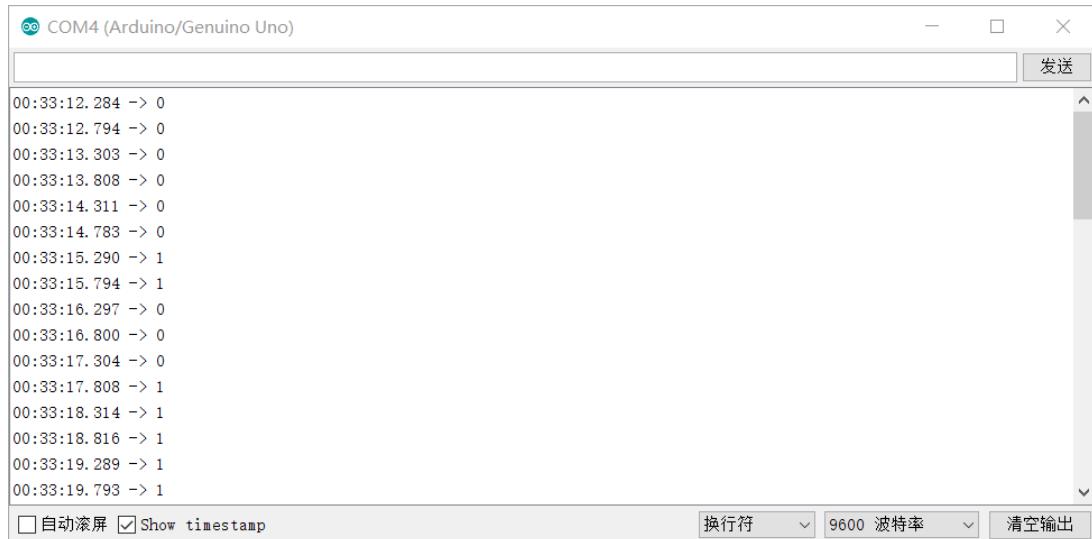


图 20.4: 串口监视器中不断打印出的 12 引脚的读数值

实验二十一 红外避障模块实验

21.1 实验目的

- (1) 学习红外避障模块的工作原理；
- (2) 基于 Arduino 获取红外避障模块的值，进而对 LED 灯与蜂鸣器进行控制。

21.2 红外避障模块

红外避障模块具有一对红外线发射管与接收管，发射管发射出一定频率的红外线，当检测方向遇到障碍物（反射面）时，红外线反射回来被接收管接收，经过比较器电路处理之后，绿色指示灯会亮起，同时信号输出接口输出数字信号（一个低电平信号），可通过电位器旋钮调节检测距离，有效距离范围 2-30cm，工作电压为 3.3V-5V。该传感器的探测距离可以通过电位器调节，由于使用的是红外线，所以其抗干扰能力很强，它在距离适中的时候测量精度很高。除此之外，该模块便于装配、使用方便，可以广泛应用于机器人避障、避障小车、流水线计数及黑白线循迹等众多场合。



图 21.1: 红外避障模块示意图

21.3 主要工作参数

工作电压	3.3-5V
工作电流	10mA 以内
检测距离	2-30cm (可通过电位器旋钮调节)
检测角度	35°

表 21.1: 红外避障模块的主要工作参数

21.4 实验原理

利用红外避障模块的工作特性，当红外避障模块检测到障碍物时，LED 亮起且蜂鸣器响；如果没有检测到障碍物，则 LED 熄灭且蜂鸣器不响。

21.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、红外避障模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

21.6 实验步骤

- (1) 按照如图21.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将障碍物（例如手指）置于红外避障模块前，观察 LED 灯与蜂鸣器的变化。

21.7 接线图

- 红外避障模块：OUT 接 12-S，GND 接 12-G，VCC 接 12-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。



图 21.2: 实验接线图

21.8 实验程序

```
1 // InfraredObstacleAvoidance.ino
2 // Date: 2020/04/28
3
4 int Led_pin = 11;
5 int Buzzer_pin = 13;
6 int Sensor_pin = 12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value = digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 // delay(500);
31 }
```

21.9 实验结果

如图21.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们用手指挡在红外避障模块的发射管与接收管之前时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 12 引脚的读数值，模块前无遮挡时会输出 0，而当用手指挡在模块前时会输出 1，如图21.4所示。

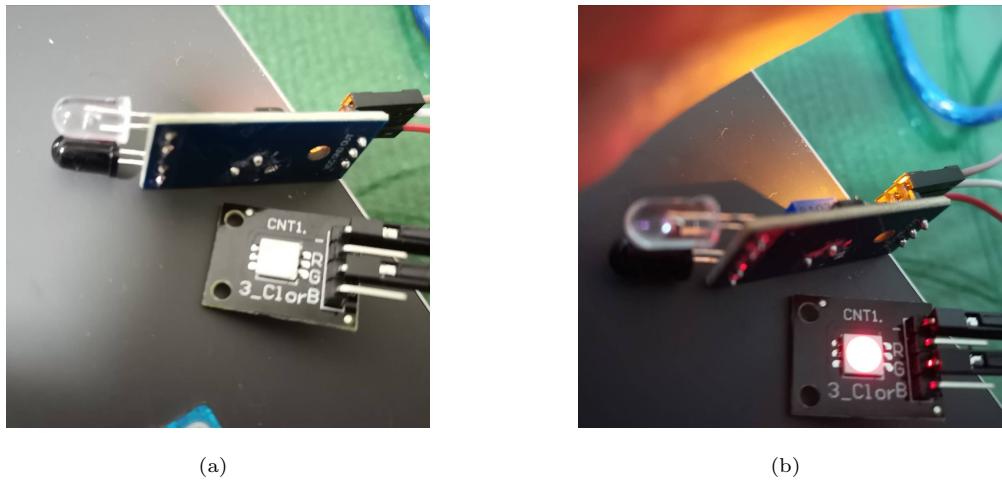


图 21.3: 红外避障模块实验结果示意图: (a) 模块前无遮挡时, LED 灯不亮且蜂鸣器不发出声音; (b) 用手挡在模块前时, LED 亮起且蜂鸣器发出响声。

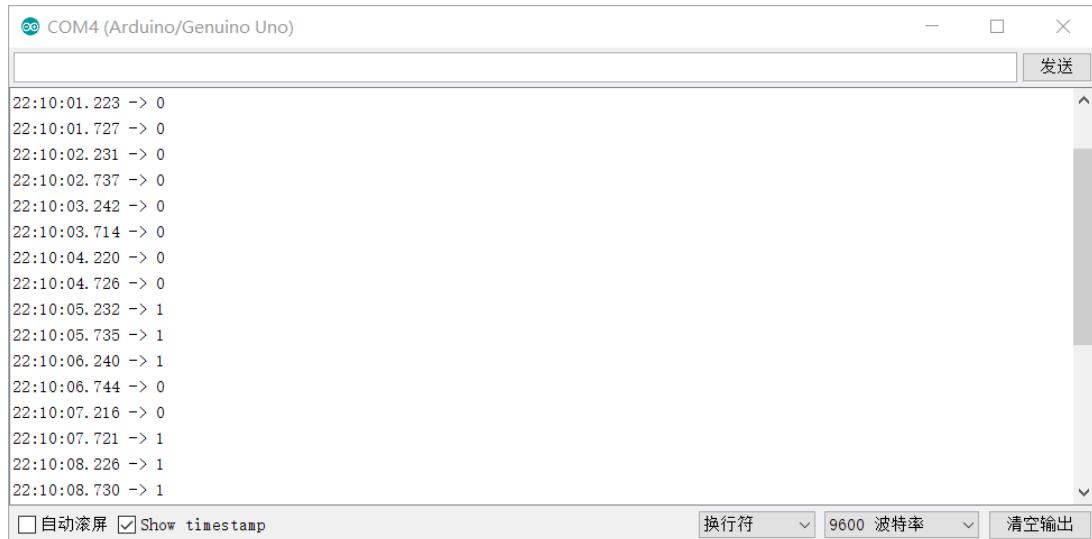


图 21.4: 串口监视器中不断打印出的 12 引脚的读数值

实验二十二 热敏电阻模块实验

22.1 实验目的

- (1) 学习热敏电阻的工作原理；
- (2) 使用热敏电阻模块来控制 LED 灯与蜂鸣器。

22.2 热敏电阻

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在温度越高时电阻值越大，负温度系数热敏电阻器在温度越高时电阻值越低，它们同属于半导体器件。



图 22.1: 热敏电阻模块示意图

22.3 主要工作参数

工作电压	3.3-5V
比较器输出	<15mA
输出形式	数字量输出 (0和1)
尺寸	3.2×1.4cm

表 22.1: 热敏电阻模块的主要工作参数

22.4 实验原理

Arduino 主板读取热敏电阻的输出数字值，当读取的模拟值达到阀值0 (低电平) 的时候，LED 灯亮起，同时有源蜂鸣器响；反之则 LED 熄灭，有源蜂鸣器不响。

22.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、热敏电阻模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

22.6 实验步骤

- (1) 按照如图22.2所示的方式完成接线;
- (2) 打开 Arduino IDE，输入程序;
- (3) 检查无误后上传程序;
- (4) 想办法改变热敏电阻的温度 (如在热敏电阻周围点着打火机)，观察 LED 灯与蜂鸣器的变化。

22.7 接线图

- 热敏电阻模块: DO 接 12-S, GND 接 12-G, VCC 接 12-V;
- 有源蜂鸣器模块: - 接 13-G, S 接 13-S, 中间引脚接 13-V;
- LED 模块: - 接 11-G, G 接 11-S。

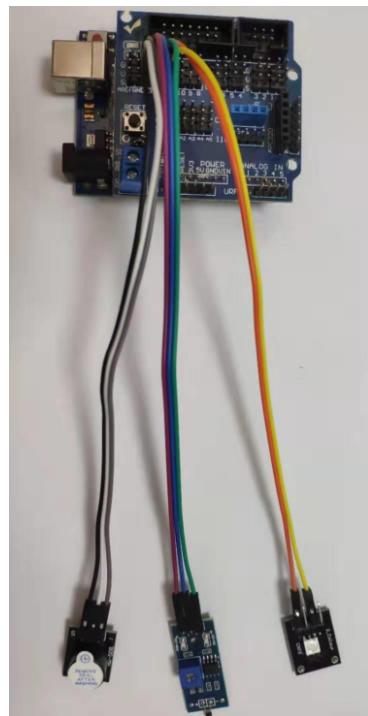


图 22.2: 实验接线图

22.8 实验程序

```
1 // Thermistor.ino
2 // Date: 2020/05/03
3
4 int Led_pin = 11;
5 int Buzzer_pin = 13;
6 int Sensor_pin = 12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value=digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 }
```

22.9 实验结果

如图22.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们用手握住模块的检测器时，起检测到的温度升高，当该温度大于设定的阈值时，LED 灯被点亮，蜂鸣器响起。

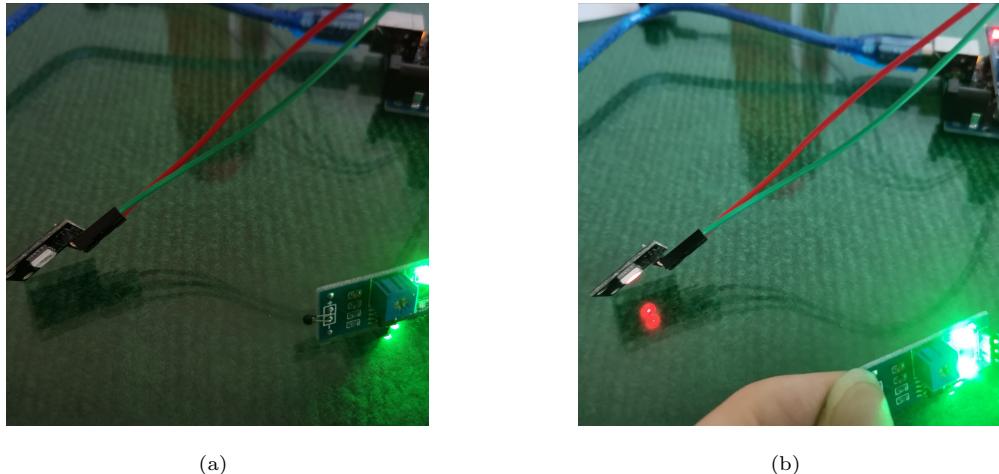


图 22.3: 热敏电阻模块实验结果示意图: (a) 常温下, LED 灯不亮, 蜂鸣器不响; (b) 用手握住模块的检测器, LED 灯亮起, 蜂鸣器报警。

实验二十三 水深传感器模块实验

23.1 实验目的

- (1) 学习水深传感器的工作原理；
- (2) 获取水深传感器测得的水位深度值并将其打印出来。

23.2 水深传感器

水深传感器专为水位检测而设计，可广泛用于感应降雨，水位，甚至液体泄漏。模块主要由三部分组成：电子连接器， $1M\Omega$ 电阻器和几条裸导线。水深传感器可通过一系列裸露的平行线缝合来判断水位，以测量水滴/水的大小。该模块可轻松将水量转换为模拟信号，输出模拟值可直接用于程序，实现水位报警功能。



图 23.1: 水深传感器模块示意图

23.3 主要工作参数

工作电压	5V
工作电流	<16mA
工作温度	10-30°C
检测宽度	40×16mm
输出电压	0-4.2V

表 23.1: 水深传感器模块的主要工作参数

23.4 实验用品

Arduino UNO 开发板、USB 数据线、水深传感器模块、若干杜邦线。

23.5 实验步骤

- (1) 按照如图23.2所示的方式完成接线;
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 打开串口监视器，并将水深传感器放入不同深度的水中，观察其读数变化。

23.6 接线图

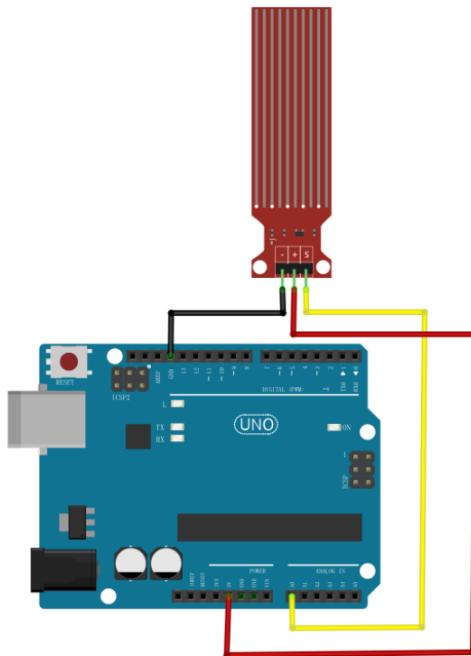


图 23.2: 实验接线图

23.7 实验程序

```
1 // Water_Level_Sensor.ino
2 // Date: 2020/05/02
3
4 int analog_pin=A0;
5
6 void setup() {
7     pinMode(analog_pin, INPUT);
8     Serial.begin(9600);
9 }
10
11 void loop() {
12     double temp = (long)analogRead(analog_pin);
13     double data = (temp/650)*4;
14     Serial.print("the depth is:");
15     Serial.print(data);
16     Serial.println("cm");
17     delay(1000);
18 }
```

23.8 实验结果

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的水位值，如图23.3所示。

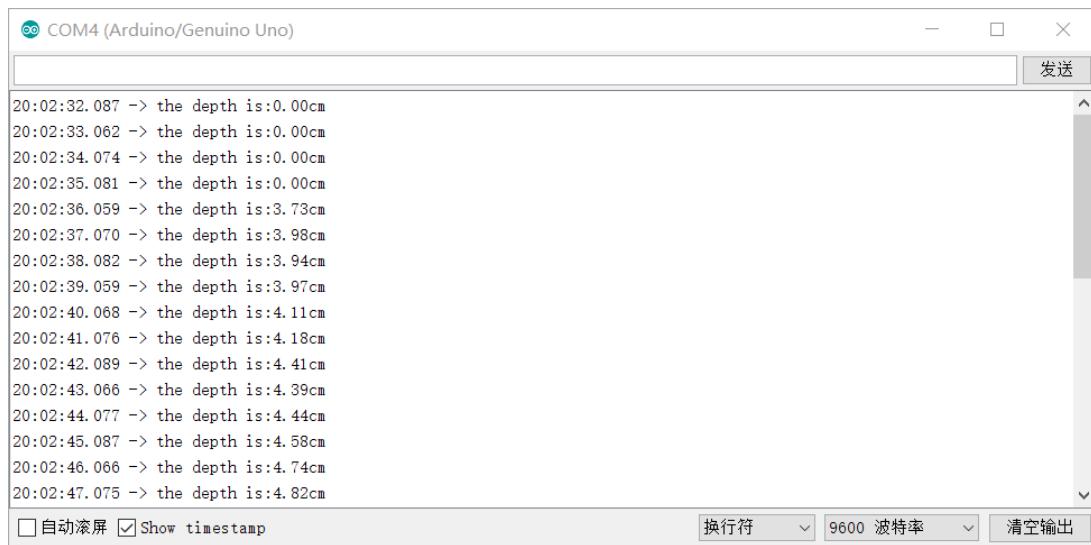


图 23.3: 串口监视器中不断打印出的水位值

实验二十四 雨滴传感器模块实验

24.1 实验目的

- (1) 学习雨滴传感器的工作原理；
- (2) 用雨滴传感器检测是否有水滴及水滴的多少。

24.2 雨滴传感器

雨滴传感器是一种传感装置，主要用于检测是否下雨及雨量的大小，并广泛应用于汽车自动刮水系统、智能灯光系统和智能天窗系统等。它具有抗氧化，导电性和更出色的耐久性能，当传感器连接到 5V 电源时，电源指示灯亮，感应板上没有水滴，DO 输出处于高电平，开关指示灯熄灭；当滴下液滴时，DO 输出处于低电平，开关指示灯将亮起；如果我们刷掉水滴，输出将返回高水平状态。AO 模拟输出可以连接到微控制器上的 AD 接口，以检测上面雨滴的大小。DO 数字输出也可以连接微控制器，以检测是否有雨。

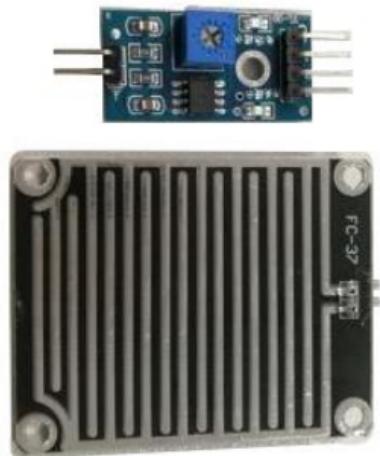


图 24.1: 雨滴传感器模块示意图

24.3 主要工作参数

工作电压	3.5-5V
输出形式	数字量输出 / 模拟量输出
尺寸	5.0×4.0cm

表 24.1: 雨滴传感器模块的主要工作参数

24.4 实验原理

当我们把雨滴滴到传感器上时，传感器会通过电位器上的模拟接口将数据传输到 Arduino 板。在此过程之后，当前的降雨大小将显示在 LCD1602 上。当没有下雨或我们刷掉雨滴之后，LCD1602 将显示当前的雨量为零。

24.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、雨滴传感器模块、LCD1602 模块、若干杜邦线。

24.6 实验步骤

- (1) 按照如图24.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将不同大小和数量的水滴滴在雨滴传感器上，观察 LCD1602 显示屏上的读数变化。

24.7 接线图

- 雨滴传感器模块：OUT 接 AO-S，GND 接 AO-G，VCC 接 AO-V；
- LCD 显示模块：GND 接 A4-G，VCC 接 A4-V，SDA 接 A4-S，SCL 接 A5-S。



图 24.2: 实验接线图

24.8 实验程序

```
1 // Raindrop_Sensor.ino
2 // Date: 2020/05/03
3
4 #include <Wire.h>
5 #include "LiquidCrystal_I2C.h"
6
7 int analogPin = A0;
8 LiquidCrystal_I2C lcd(0x27, 16, 2);
9
10 void setup()
11 {
12     lcd.init();
13     lcd.backlight();
14     pinMode(analogPin, INPUT);
15     Serial.begin(9600);
16 }
17
18 void loop() {
19     int data = analogRead(analogPin);
20     int j = 1023 - data;
21     lcd.setCursor(0, 0);
22     lcd.print("The rainfall is: ");
23     // set the cursor to column 0, line 1
24     // (note: line 1 is the second row, since counting begins with 0)
25     lcd.setCursor(0, 1);
26     // print the number of seconds since reset:
27     lcd.print((float)j, 2);
28     lcd.print("mm");
29     delay(200);
30 }
```

24.9 实验结果

在雨滴传感器模块上没有水分时，LCD 显示屏上显示的降雨量为 0；在我们往模块上加水的时候可以观察到，LCD 显示屏上的示数随着我们加入水量的增加而增加，如图 24.3 所示。

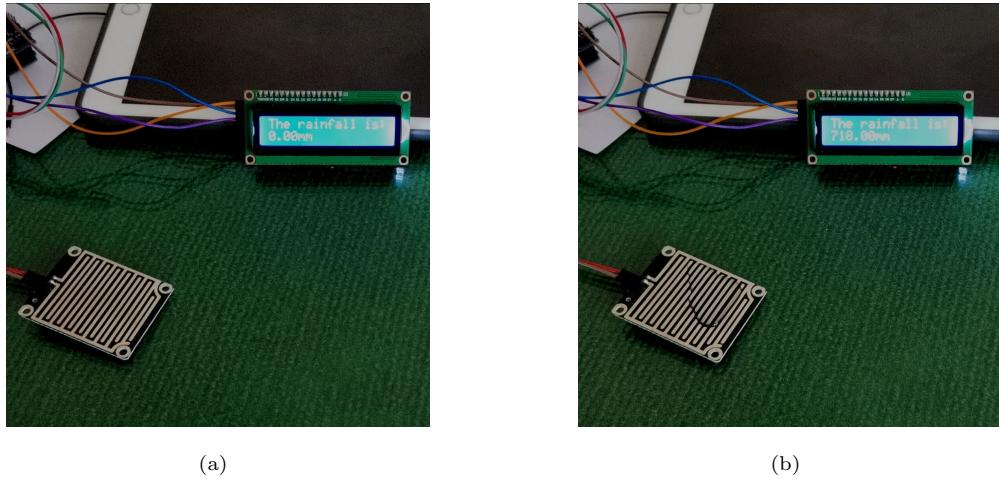


图 24.3: 雨滴传感器模块实验结果示意图: (a) 滴水前; (b) 滴水后。

实验二十五 土壤湿度传感器模块实验

25.1 实验目的

- (1) 学习土壤湿度传感器的工作原理；
- (2) 用土壤湿度传感器检测给定土壤的湿度，并利用 LCD1602 将其显示出来。

25.2 土壤湿度传感器

土壤湿度传感器模块有两个铜条作为探头。将它们插入土壤时，它们可以检测到水分。土壤越湿润则导电性越好，它们之间的电阻越低；而土壤越干燥则导电性就越差，其之间的阻值也就越高。土壤湿度传感器是模拟传感器，因此我们通过模拟输入获得电压值；我们也可以通过数字接口获取高低电平。因为土壤的湿度可以分为几个等级，当我们利用土壤湿度传感器做一个自动浇花系统的时将会非常方便。



图 25.1: 雨滴传感器模块示意图

25.3 主要工作参数

工作电压	3.3-5V
工作电流	<20mA
PCB 尺寸	32×14mm
输出形式	模拟量输出 / 数字量输出

表 25.1: 土壤湿度传感器模块的主要工作参数

25.4 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、土壤湿度传感器模块、LCD1602 模块、若干杜邦线。

25.5 实验步骤

- (1) 按照如图25.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将传感器的探头插入土壤，观察 LCD1602 显示屏上的读数变化。

25.6 接线图

- 土壤湿度传感器模块：AO 接 AO-S，GND 接 AO-G，VCC 接 AO-V；
- LCD 显示模块：GND 接 A4-G，VCC 接 A4-V，SDA 接 A4-S，SCL 接 A5-S。

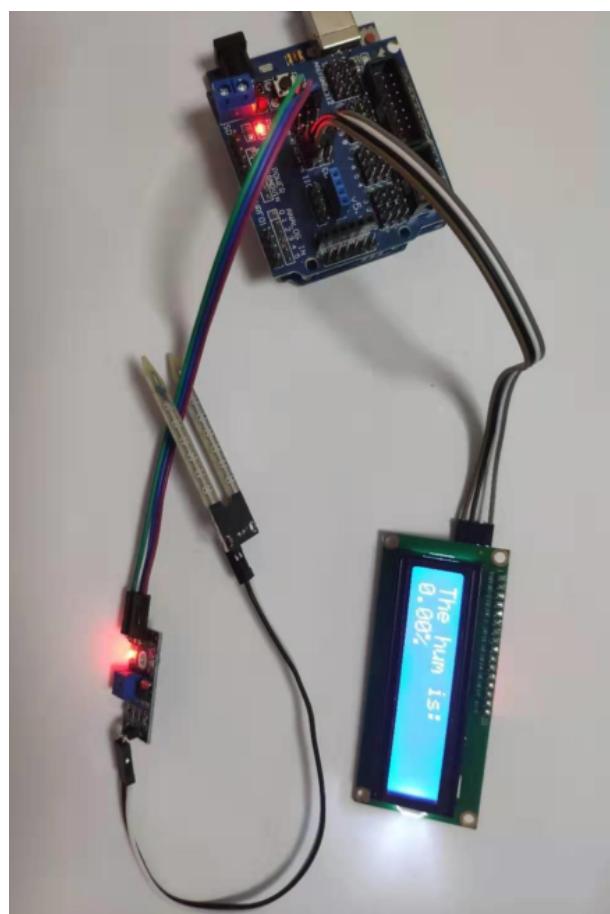


图 25.2: 实验接线图

25.7 实验程序

```
1 // SoilMoistureSensor.ino
2 // Date: 2020/05/02
3
4 #include <Wire.h>
5 #include "LiquidCrystal_I2C.h"
6
7 int analogPin = A0;
8 LiquidCrystal_I2C lcd(0x27, 16, 2);
9
10 void setup()
11 {
12     lcd.init();
13     lcd.backlight();
14     pinMode(analogPin, INPUT);
15     Serial.begin(9600);
16 }
17
18 void loop() {
19     float data=analogRead(analogPin);
20     Serial.println(data);
21     float i = data / 1023;
22     float j = (1-i) * 100;
23     lcd.setCursor(0, 0);
24     lcd.print("The hum is: ");
25     lcd.setCursor(0, 1);
26     lcd.print((float)j, 2);
27     lcd.print("%");
28     delay(200);
29 }
```

25.8 实验结果

将模块插入土壤中，观察一次示数，然后向土壤中加入少量水，再次观察示数。结果如图25.3所示。

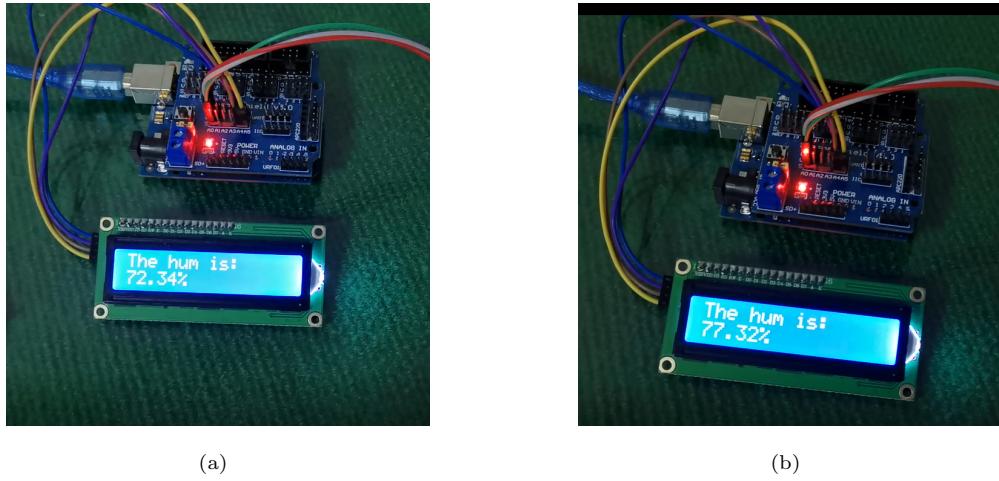


图 25.3: 土壤湿度传感器模块实验结果示意图: (a) 加水前; (b) 加水后。

实验二十六 霍尔开关模块实验

26.1 实验目的

- (1) 学习霍尔开关的工作原理；
- (2) 使用霍尔开关控制 LED 灯与蜂鸣器。

26.2 霍尔开关

当一块通有电流的金属或半导体薄片垂直地放在磁场中时，薄片的两端就会产生电位差，这种现象就称为霍尔效应。其两端具有的电位差值称为霍尔电势 U ，表达式为

$$U = \frac{KIB}{d}$$

式中， K 为霍尔系数， I 为薄片中通过的电流， B 为外加磁场的磁感应强度， d 是薄片的厚度。

由此可见，霍尔效应的灵敏度高低与外加磁场的磁感应强度成正比的关系。霍尔开关就属于这种有源磁电转换器件，它是在霍尔效应原理的基础上，利用集成封装和组装工艺制作而成，它可方便的把磁输入信号转换成实际应用中的电信号，同时又具备工业场合实际应用易操作和可靠性的要求。

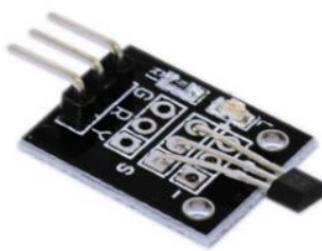


图 26.1: 霍尔开关模块示意图

霍尔开关的输入端是以磁感应强度 B 来表征的，当 B 值达到一定的程度时，霍尔开关内部的触发器翻转，霍尔开关的输出电平状态也随之翻转。输出端一般采用晶体管输出，和接近开关类似有 NPN、PNP、常开型、常闭型、锁存型（双极性）、双信号输出之分。

霍尔开关具有无触点、低功耗、长使用寿命、响应频率高等特点，内部采用环氧树脂封灌成一体化，所以能在各类恶劣环境下可靠的工作。霍尔开关可应用于接近开关，压力开关，里程表等，作为一种新型的电器配件。

26.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	32×14mm
输出形式	数字开关量输出 (0和1)

表 26.1: 霍尔开关模块的主要工作参数

26.4 实验原理

霍尔开关在通电后，当没有磁通量通过霍尔开关的时，开关处于断开状态；当有磁铁等靠近霍尔开关时会使得霍尔开关导通，从而使 LED 灯亮起且蜂鸣器响起。

26.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、霍尔开关模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

26.6 实验步骤

- (1) 按照如图26.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将磁铁靠近霍尔开关，观察 LED 灯和蜂鸣器的变化。

26.7 接线图

- 霍尔开关模块: S 接 12-S, - 接 12-G, + 接 12-V;
- 有源蜂鸣器模块: - 接 13-G, S 接 13-S, 中间引脚接 13-V;
- LED 模块: - 接 11-G, G 接 11-S。

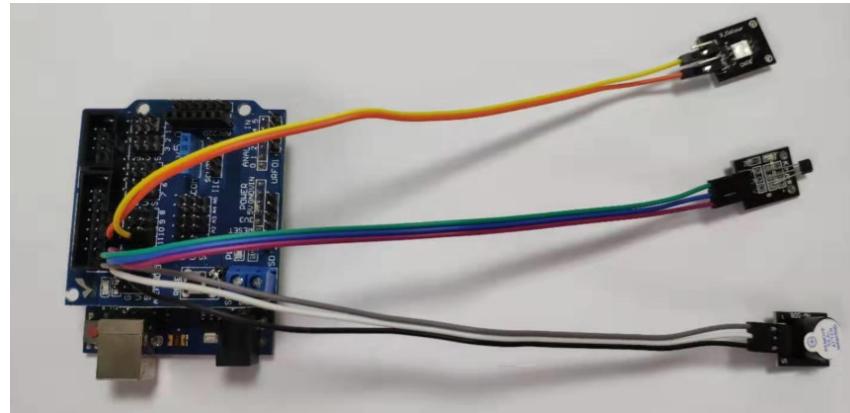


图 26.2: 实验接线图

26.8 实验程序

```
1 // HallSwitch.ino
2 // Date: 2020/05/02
3
4 int Led_pin=11;
5 int Buzzer_pin=13;
6 int Sensor_pin=12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value=digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 }
```

26.9 实验结果

如图26.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们把霍尔开关模块靠近磁铁时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 12 引脚的读数值，模块周围无磁铁时会输出 1，有磁铁时会输出 0，如图26.4所示。

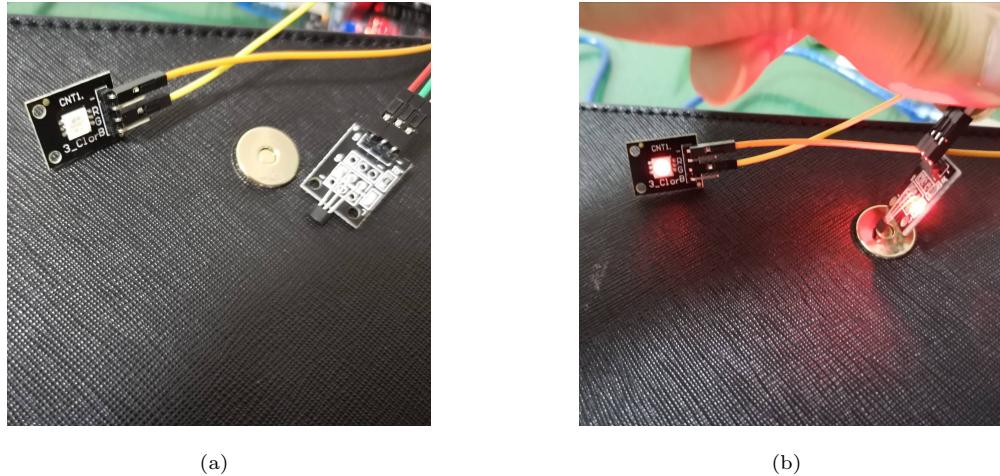


图 26.3: 霍尔开关模块实验结果示意图：(a) 模块周围无磁铁时，LED 灯不亮且蜂鸣器不发出声音；(b) 模块周围有磁铁时，LED 亮起且蜂鸣器发出响声。



图 26.4: 串口监视器中不断打印出的 12 引脚的读数值

实验二十七 倾斜开关模块实验

27.1 实验目的

- (1) 学习倾斜开关的工作原理；
- (2) 基于 Arduino 控制倾斜开关模块来实现防盗报警的功能。

27.2 倾斜开关

倾斜开关模块也称为珠形开关或钢球开关，它实际上是一种振动开关。它的工作原理是滚珠通过与销接触或不接触来控制电路的连接或断开。简单地说，就像打开或关闭灯一样，如果开关接触内部的金属板，则灯将亮；当开关离开时，灯就会熄灭。与金属端子接触或在开关中用小珠子改变光的行进路径将能够产生传导效应。

倾斜传感器模块采用 MEC 原装倾斜开关 SW-460D，灵敏度高，它一般用来检测物体角度的变化。模块在无倾斜或者倾斜角度达不到设定阈值时，DO 口输出高电平，当传感器倾斜角度超过设定阈值时，模块 D0 输出低电平；数字量输出 D0 可以与单片机直接相连，通过单片机来检测高低电平，由此来检测物体角度的变化。倾斜开关可以应用于胎压监控系统 (TPMS)、脚踏车灯、数位相框旋转、萤幕旋转、视讯镜头翻转、防盗系统等。



图 27.1: 倾斜开关模块示意图

27.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	32×14mm
输出形式	数字开关量输出 (0和1)

表 27.1: 倾斜开关模块的主要工作参数

27.4 实验原理

在本实验中，我们利用倾斜开关模块的工作特性，当电路板处于静止状态时，倾斜开关模块闭合；当有外力触碰到了电路板致使其倾斜开关里面的滚珠和导线断开接触从而断开电路时会触发报警，LED 亮起且蜂鸣器发出响声；当断开模块恢复到静止状态时，LED 熄灭且蜂鸣器无响声。

27.5 实验用品

Arduino UNO 开发板、USB 数据线、面包板、倾斜开关模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

27.6 实验步骤

- (1) 按照如图27.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 触碰倾斜开关模块，观察 LED 灯和蜂鸣器的变化。

27.7 接线图

- 倾斜开关模块: DO 接 12-S, GND 接 12-G, VCC 接 12-V;
- 有源蜂鸣器模块: - 接 13-G, S 接 13-S, 中间引脚接 13-V;
- LED 模块: - 接 11-G, G 接 11-S。

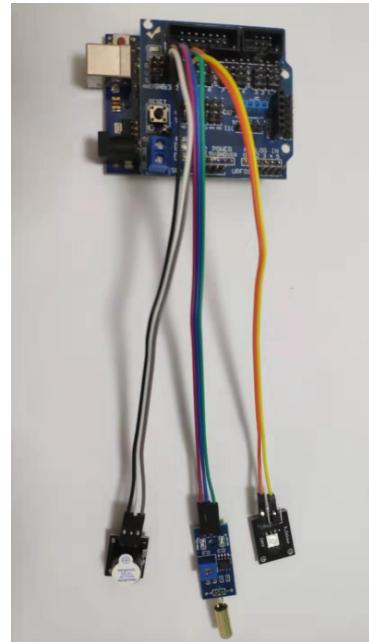


图 27.2: 实验接线图

27.8 实验程序

```
1 // Tilt_Switch_Module.ino
2 // Date: 2020/05/02
3
4 int Led_pin=11;
5 int Buzzer_pin=13;
6 int Button_pin=12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Button_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value=digitalRead(Button_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24         // delay(1000);
25     }
}
```

```

26     else
27     {
28         digitalWrite(Led_pin, LOW);
29         digitalWrite(Buzzer_pin, LOW);
30     }
31 }
```

27.9 实验结果

如图27.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们用手将倾斜开关拿起而使其倾斜时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 12 引脚的读数值，倾斜开关平放时会输出 1，倾斜时会输出 0，如图27.4所示。

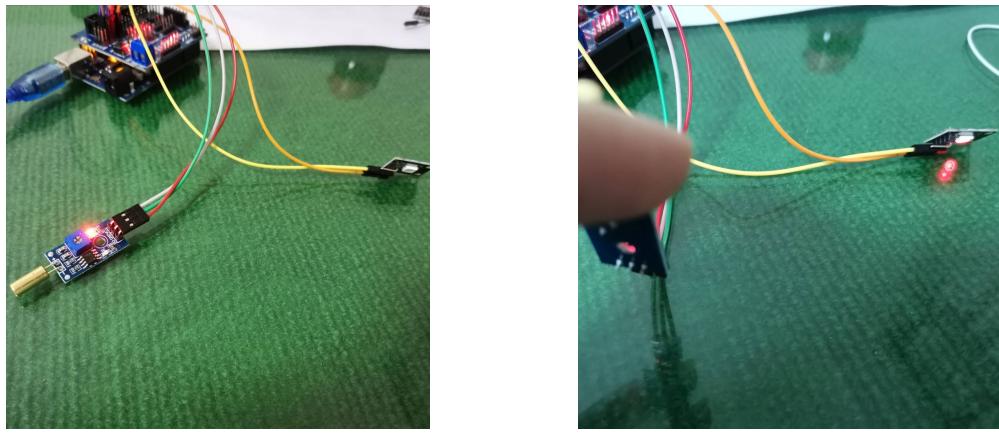


图 27.3: 倾斜开关模块实验结果示意图：(a) 倾斜开关平放时，LED 灯不亮且蜂鸣器不发出声音；(b) 用手将倾斜开关拿起而使其倾斜时，LED 亮起且蜂鸣器发出响声。

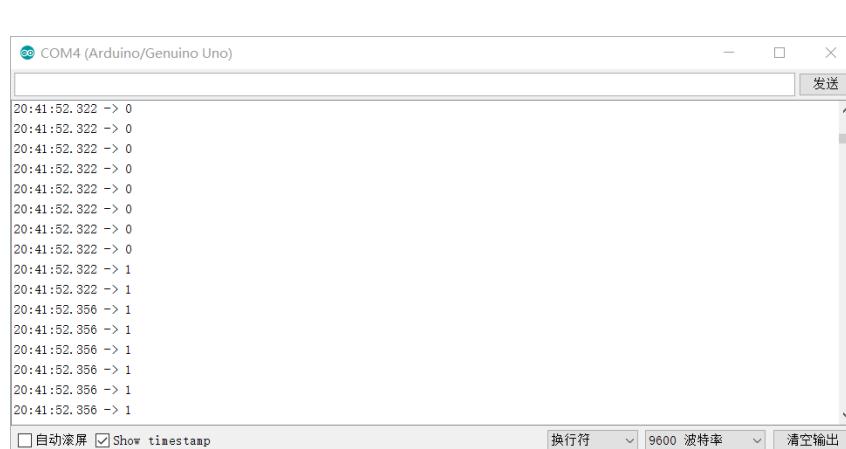


图 27.4: 串口监视器中不断打印出的 12 引脚的读数值

实验二十八 磁簧开关模块实验

28.1 磁簧开关

磁簧开关 (Reed Switch)，也叫干簧管，它是一个通过所施加的磁场来操作的电开关。磁簧开关的工作原理非常简单，两片端点处重叠的可磁化的簧片（通常是由铁和镍这两种金属所组成的）密封于一玻璃管中，两簧片呈交迭状且间隔有一小段空隙（仅约几个微米），这两片簧片上的触点上镀有层很硬的金属，通常都是铑和钌，这层硬金属大大提升了切换次数及产品寿命。玻璃管中装填有高纯度的惰性气体（如氮气），部份干簧开关为了提升其高压性能，更会把内部做成真空状态。

簧片的作用相当与一个磁通导体。在尚未操作时，两片簧片并未接触；在通过永久磁铁或电磁线圈产生的磁场时，外加的磁场使两片簧片端点位置附近产生不同的极性，当磁力超过簧片本身的弹力时，这两片簧片会吸合导通电路；当磁场减弱或消失后，干簧片由于本身的弹性而释放，触面就会分开从而打开电路。



图 28.1: 磁簧开关模块示意图

28.2 主要工作参数

工作电压	3.3-5V
PCB 尺寸	32×14mm
输出形式	数字开关量输出 (0和1)

表 28.1: 磁簧开关模块的主要工作参数

28.3 实验原理

磁簧开关在通电后，当没有磁通量通过磁簧开关的时，开关处于断开状态；当有磁铁等靠近磁簧开关时会使得磁簧开关导通，从而使 LED 灯亮起且蜂鸣器响起。

28.4 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、磁簧开关模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

28.5 实验步骤

- (1) 按照如图28.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将磁铁靠近磁簧开关，观察 LED 灯和蜂鸣器的变化。

28.6 接线图

- 磁簧开关模块：S 接 12-S，- 接 12-G，+ 接 12-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。

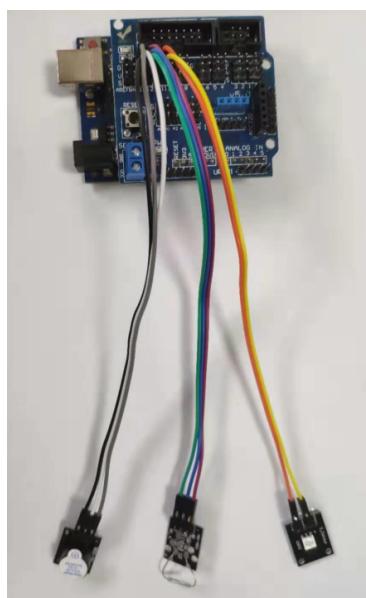


图 28.2: 实验接线图

28.7 实验程序

```
1 // Reed_Switch_Module.ino
2 // Date: 2020/05/02
3
4 int Led_pin=11;
5 int Buzzer_pin=13;
6 int Sensor_pin=12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value=digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 }
```

28.8 实验结果

如图28.3所示，当程序烧录成功后，LED 灯为熄灭状态且蜂鸣器不发出响声；当我们把磁簧开关模块靠近磁铁时，可以观察到 LED 灯亮起且蜂鸣器发出响声。

打开串口监视器，将波特率设置为9600 (和程序中一致)，即可看到不断打印出的 12 引脚的读数值，模块周围无磁铁时会输出 1，有磁铁时会输出 0，如图28.4所示。

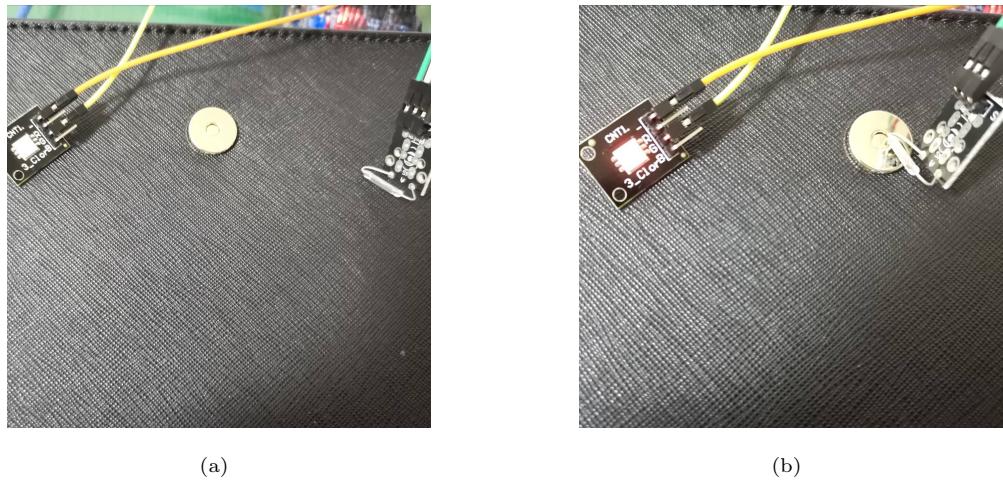


图 28.3: 磁簧开关模块实验结果示意图: (a) 模块周围无磁铁时, LED 灯不亮且蜂鸣器不发出声音; (b) 模块周围有磁铁时, LED 亮起且蜂鸣器发出响声。

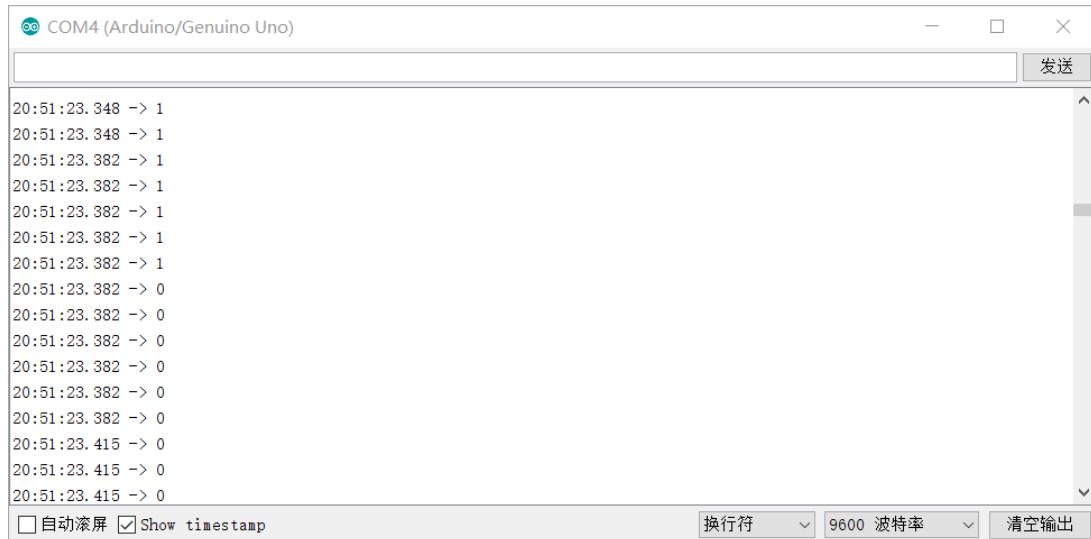


图 28.4: 串口监视器中不断打印出的 12 引脚的读数值

实验二十九 声音传感器模块实验

29.1 实验目的

- (1) 学习声音传感器的工作原理；
- (2) 利用声音传感器和 LED 灯模块实现声控灯的功能。

29.2 声音传感器

声音传感器模块的作用相当于一个话筒（麦克风），它可以用来接收声波，显示声音的振动图象，但不能对噪声的强度进行测量。该传感器内置一个对声音敏感的电容式驻极体话筒。声波使话筒内的驻极体薄膜振动，导致电容发生变化，从而产生与之对应的微小电压变化，这一电压随后被转化成 0-5V 的电压，经过 A/D 转换被数据采集器接受，并传送给主控芯片。



图 29.1: 声音传感器模块示意图

29.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	35×15mm
输出形式	模拟量输出 / 数字量输出

表 29.1: 声音传感器模块的主要工作参数

29.4 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、声音传感器模块、LED 模块、若干杜邦线。

29.5 实验步骤

- (1) 按照如图29.2所示的方式完成接线;
- (2) 打开 Arduino IDE, 输入程序;
- (3) 检查无误后上传程序;
- (4) 调节声音传感器模块的电位器 (蓝色的突起), 使其检测阈值在合适的范围内。然后不断发出音量逐渐变大的声音, 观察 LED 灯的变化。

29.6 接线图

- 声音传感器模块: A0 接 A0-S, G 接 A0-G, + 接 A0-V;
- LED 模块: - 接 11-G, G 接 11-S。

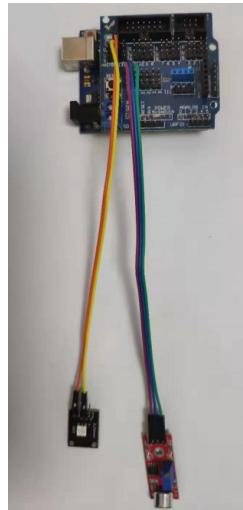


图 29.2: 实验接线图

29.7 实验程序

```
1 // VoiceOperatedSwitch.ino
2 // Date: 2020/05/02
3
4 int Voice_pin = A0;      // define analog 0 pin for voice-sensor pin
5 int LED_pin = 11;
6
7 void setup()
8 {
9     pinMode(LED_pin, OUTPUT);
10    pinMode(Voice_pin, INPUT);
11    digitalWrite(LED_pin, LOW);
```

```
12     Serial.begin(9600);
13 }
14
15 void loop()
16 {
17     int value = analogRead(Voice_pin);
18     Serial.println(value);
19     if(value >48)
20     {
21         digitalWrite(LED_pin, HIGH);
22         delay(500);
23     }
24     else
25     {
26         digitalWrite(LED_pin, LOW);
27     }
28 }
```

29.8 实验结果

如图29.3所示，当程序烧录成功后，周围环境声音音量较小（未超过设定阈值）时，LED 灯不亮；周围环境声音音量较大（超过设定阈值）时，LED 灯亮起。

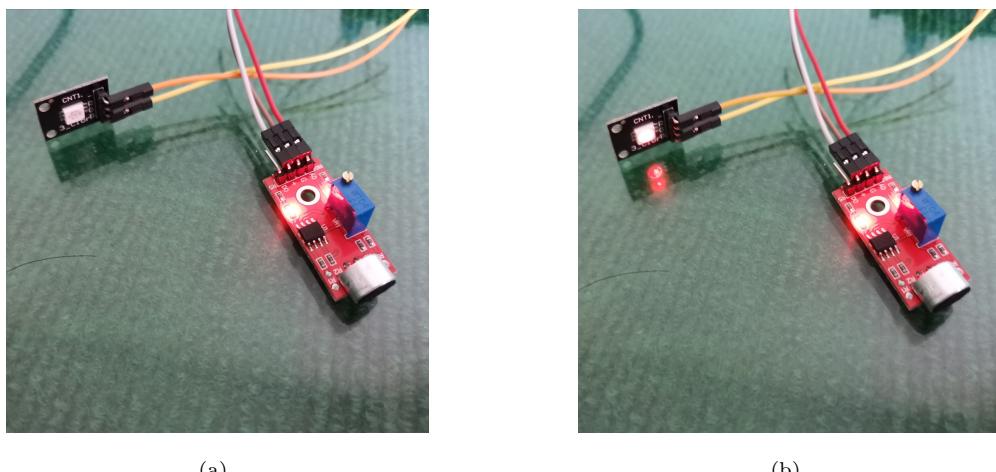


图 29.3：声音传感器模块实验结果示意图：(a) 周围环境声音音量较小（未超过设定阈值）时，LED 灯不亮；(b) 周围环境声音音量较大（超过设定阈值）时，LED 灯亮起

实验三十 手指侦测心跳模块实验

30.1 实验目的

- (1) 学习手指侦测心跳模块的工作原理；
- (2) 基于 Arduino 实现测试心跳功能。

30.2 手指侦测心跳模块

手指侦测心跳模块有一个红外发射管和一个红外接收管，人脉搏跳动的时候，血液的透光性不一样，这会导致接收器端接收到的信号强弱不一样。我们可以把变化的信号在进行滤波、放大、整形等系列处理后输出，这就是手指检测心跳模块设计的思路。

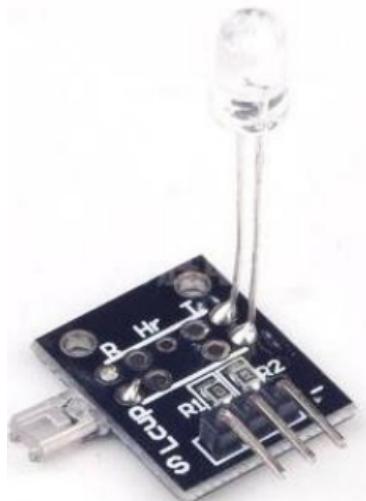


图 30.1: 手指侦测心跳模块示意图

30.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	32×14mm
输出形式	模拟量输出

表 30.1: 手指侦测心跳模块的主要工作参数

30.4 实验原理

我们将手指放在红外线发射器和红外光敏三极管之间后，随着心脏跳动，红外光敏三极管的输出也在不断变化，我们通过这种变化来检测心跳。为防止外界红外线对数据造成影响，最好将手指测心跳模块放在一个封闭的盒子里面，然后进行心跳检测。

30.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、手指侦测心跳模块、LCD 显示模块、若干杜邦线。

30.6 实验步骤

- (1) 按照如图30.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将手指放在红外线发射器和红外光敏三极管之间后，观察 LCD 显示屏上的数据。

30.7 接线图

- 手指侦测心跳模块：S 接 A0-S，- 接 A0-G，+ 接 A0-V；
- LCD 显示模块：GND 接 A4-G，VCC 接 A4-V，SDA 接 A4-S，SCL 接 A5-S。

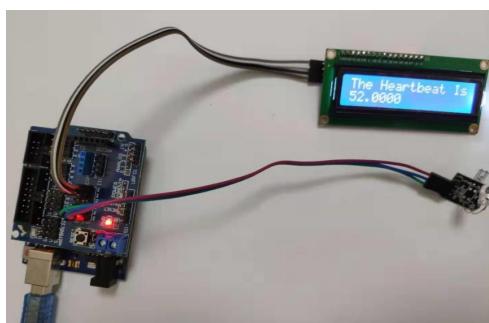


图 30.2: 实验接线图

30.8 实验程序

```
1 // HeartbeatDetector.ino
2 // Date: 2020/05/02
3
4 #include <Wire.h>
5 #include "LiquidCrystal_I2C.h"
6
7 int analog_pin = A0;
8 LiquidCrystal_I2C lcd(0x27, 16, 2);
9
10 void setup()
11 {
12     lcd.init();
13     lcd.backlight();
14     pinMode(analog_pin, INPUT);
15     Serial.begin(9600);
16 }
17
18 void loop() {
19     float data = analogRead(analog_pin);
20     Serial.println(data);
21     lcd.setCursor(0, 0);    // print at row 0, col 0 of LCD
22     lcd.print("The range is: ");
23     lcd.setCursor(0, 1);    // print at row 1, col 0 of LCD
24     lcd.print(data, 2);
25     delay(500);
26 }
```

30.9 实验结果

将手指放在红外线发射器和红外光敏三极管之间后，观察 LCD 显示屏上的数据，如图 30.3 所示。然而本实验中由于干扰因素较多（如其他来源的红外线、模块本身的精度等），导致误差极大。



图 30.3: LCD 显示屏上打印出的心跳速率信息

实验三十一 光电传感器模块实验

31.1 实验目的

- (1) 学习光电传感器模块的工作原理；
- (2) 基于 Arduino 实现防盗报警功能。

31.2 光电传感器

光电传感器模块又称为穿透型光电感应器、光遮断器、光电断续器、光电遮断器，它将发光组件与受光组件面对面排列并设置于同一封装内，利用检测物体通过时会遮光的原理来实现检测功能。光遮断模块默认输出引脚输出低电平，当模块中间凹进去的地方有东西遮住时，输出引脚输出高电平。

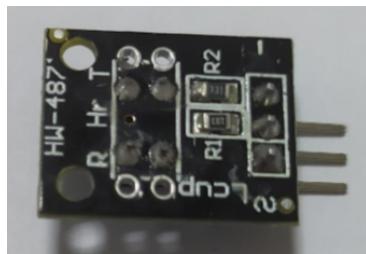


图 31.1: 光电传感器模块示意图

31.3 主要工作参数

工作电压	3.3-5V
PCB 尺寸	32×14mm
输出形式	数字量输出

表 31.1: 光电传感器模块的主要工作参数

31.4 实验原理

当光电传感器的光槽没有被遮断时，输出低电平；当光槽被遮断时，输出高电平，此时触发报警，LED 灯被点亮，蜂鸣器响起。

31.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、光电传感器模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

31.6 实验步骤

- (1) 按照如图31.2所示的方式完成接线;
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 遮挡光电传感器的光槽，观察 LED 灯与蜂鸣器的变化。

31.7 接线图

- 光电传感器模块：S 接 12-S，- 接 12-G，+ 接 12-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。

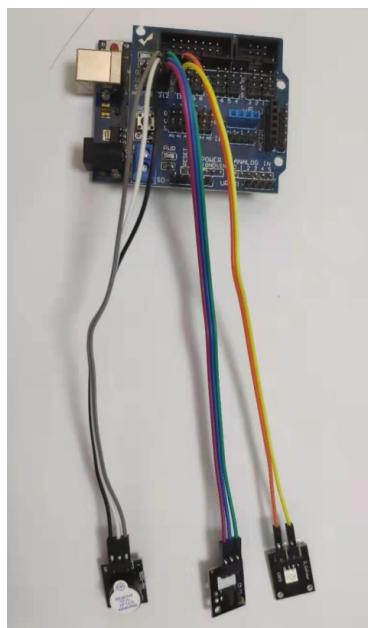


图 31.2: 实验接线图

31.8 实验程序

```
1 // LightSwitch.ino
2 // Date: 2020/05/02
3
4 int Led_pin = 11;
5 int Buzzer_pin = 13;
6 int LightSwitch_pin = 12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(LightSwitch_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value = digitalRead(LightSwitch_pin);
19     Serial.println(value);
20     if(value==1)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24         // delay(1000);
25     }
26     else
27     {
28         digitalWrite(Led_pin, LOW);
29         digitalWrite(Buzzer_pin, LOW);
30     }
31 }
```

31.9 实验结果

如图31.3所示，当程序烧录成功后，光电传感器的光槽未被遮挡时，LED 灯不亮，蜂鸣器不响；当它的光槽被遮住时，LED 灯被点亮，蜂鸣器响起。

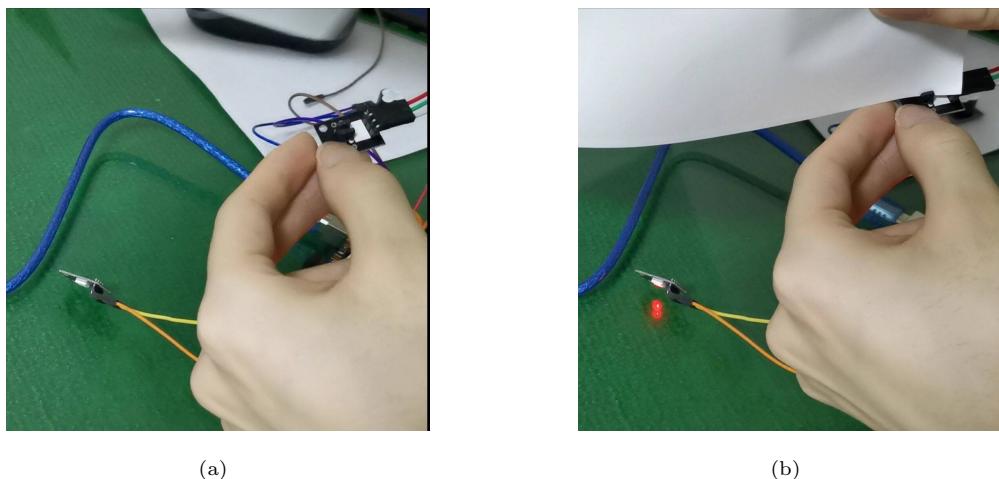


图 31.3: 光电传感器模块实验结果示意图: (a) 光电传感器的光槽未被遮住时, LED 灯不亮, 蜂鸣器不响; (b) 光电传感器的光槽被遮住时, LED 灯被点亮, 蜂鸣器响起。

实验三十二 烟雾气体传感器模块实验

32.1 实验目的

- (1) 学习 MQ-2 烟雾传感器模块的工作原理；
- (2) 基于 Arduino 实现烟雾报警功能。

32.2 MQ-2 烟雾气体传感器

MQ-2 气体传感器所使用的气敏材料是在清洁空气中电导率较低的二氧化锡 (SnO_2)。当传感器所处环境中存在可燃气体时，传感器的电导率随空气中可燃气体浓度的增加而增大。使用简单的电路即可将电导率的变化转换为与该气体浓度相对应的输出信号。MQ-2 气体传感器对液化气、丙烷、氢气的灵敏度高，对天然气和其它可燃蒸汽的检测也很理想。这种传感器可检测多种可燃性气体，是一款适合多种应用的低成本传感器。



图 32.1: MQ-2 烟雾气体传感器模块示意图

32.3 主要工作参数

输入电压	5V
功耗 (电流)	150mA
DO 输出	TTL数字量 0 和 1 (0.1 和 5V)
AO 输出	0.1-0.3V

表 32.1: MQ-2 烟雾传感器模块的主要工作参数

32.4 实验原理

MQ-2 烟雾传感器的灵敏度很高，当测量浓度大于设定浓度（达到阀值）时，模块会输出低电平信号，此时触发报警，LED 被点亮，同时蜂鸣器响起。

32.5 实验用品

Arduino UNO 开发板、Sensor V5.0 扩展板、USB 数据线、MQ-2 烟雾传感器模块、LED 模块、有源蜂鸣器模块、若干杜邦线。

32.6 实验步骤

- (1) 按照如图32.2所示的方式完成接线；
- (2) 打开 Arduino IDE，输入程序；
- (3) 检查无误后上传程序；
- (4) 将 MQ-2 烟雾传感器模块倒置，用打火机在其下方打火（注意保持适当距离，防止烧坏模块），观察 LED 灯和蜂鸣器的变化。

32.7 接线图

- MQ-2 烟雾传感器模块：DO 接 12-S，GND 接 12-G，VCC 接 12-V；
- 有源蜂鸣器模块：- 接 13-G，S 接 13-S，中间引脚接 13-V；
- LED 模块：- 接 11-G，G 接 11-S。

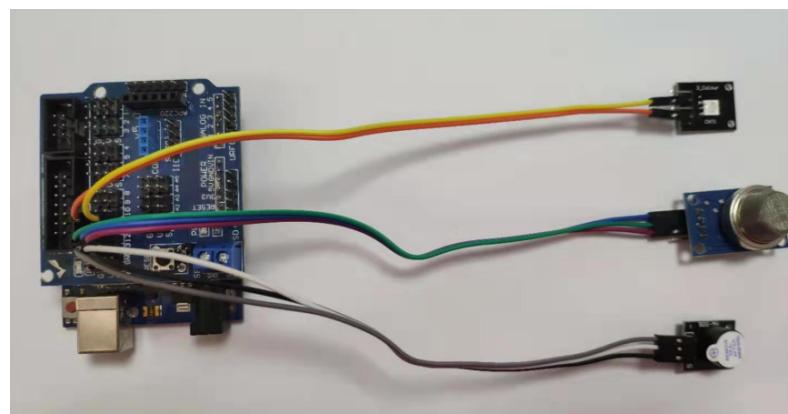


图 32.2: 实验接线图

32.8 实验程序

```
1 // MQ2_Sensor_Module.ino
2 // Date: 2020/05/02
3
4 int Led_pin = 11;
5 int Buzzer_pin = 13;
6 int Sensor_pin = 12;
7
8 void setup()
9 {
10     pinMode(Led_pin, OUTPUT);
11     pinMode(Buzzer_pin, OUTPUT);
12     pinMode(Sensor_pin, INPUT);
13     Serial.begin(9600);
14 }
15
16 void loop()
17 {
18     int value = digitalRead(Sensor_pin);
19     Serial.println(value);
20     if(value==0)
21     {
22         digitalWrite(Led_pin, HIGH);
23         digitalWrite(Buzzer_pin, HIGH);
24     }
25     else
26     {
27         digitalWrite(Led_pin, LOW);
28         digitalWrite(Buzzer_pin, LOW);
29     }
30 }
```

32.9 实验结果

将 MQ-2 烟雾传感器模块倒置，用打火机在其下方打火（注意保持适当距离，防止烧坏模块），即可看到 LED 灯亮起，蜂鸣器报警。