实验报告

电子工艺实训

姓	名	熊恪峥
学	号	22920202204622
日	期	2022年6月27日 至2022年7月8日
年	级	2020级
系	别	计算机科学系

电子工艺实训

姓名：熊恪峥　　学号：22920202204622　　总分：

目录

一、PCB制作部分

1、PCB设计过程及遇到的主要问题

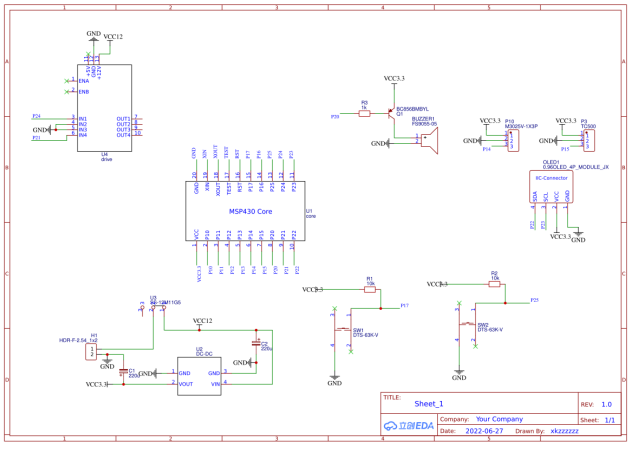
PCB制作过程和遇到的问题如表 1。

表格 1: PCB制作过程和遇到的问题

制作过程	问题
原理图绘制	1. 在绘制原理图时连线有虚接的现象。 2. 没有连接到正确的端口。
封装	1. 某些封装没有注意尺寸问题，使得原件安装困难。 2. 有些封装没有明显标注正负极性，导致安装的时候翻查原理图。
布线	1. 没有注意线宽导致线过细，之后进行了修改 2. 在保证无锐角、无重合的时候遇到了困难，之后重新调整了原件布置顺序 3. 开关的引脚尺寸不合适
布局	1. 有些部分没有为安装预留足够的空隙 2. 没有留够助焊区域使得焊接困难

PCB制作过程中的原理图如图 1。

图 1: 原理图



封装如图 2，PCB的正反面如图 5。实物图如图 4。

图 2: 封装

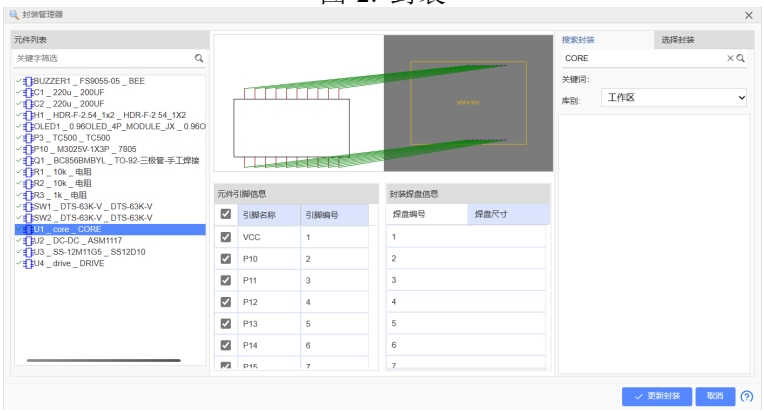
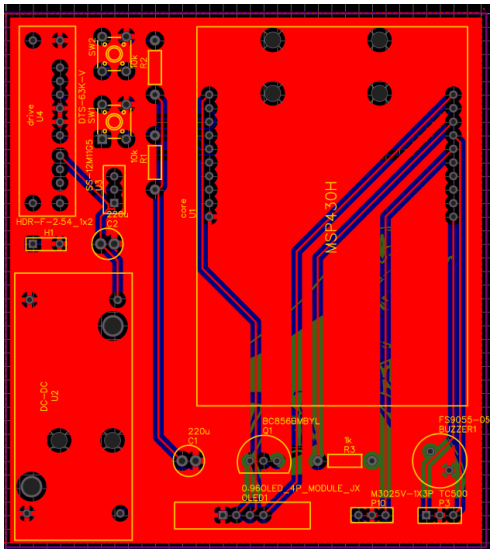
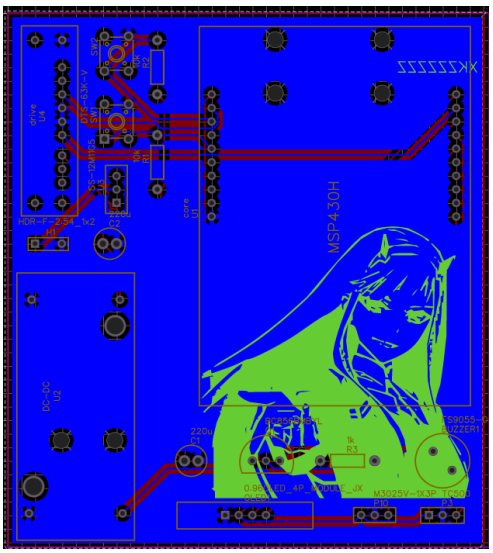


图 3: PCB



(a) 顶部



(b) 底部

图 4: 实物图

2、注意事项

1. 需要插接导线或者其它线缆的接口元器件一般放到电路板的外侧，并且接线的一面要朝外。例如，P1主控板插针一般放到电路板的外侧，使单片机接上后多余的部分露在外侧，减少占用PCB板空间。
2. 元器件就近原则。元器件就近放置，可以缩短PCB导线的距离，如果是去耦电容或者滤波电容，越靠近元器件，效果越好。例如，H5和SW1就近放置，可以缩短PCB导线距离。
3. 整齐排列。一个IC芯片的辅助电容电阻电路，围绕此IC把电阻电容整齐的排列，可以更美观。例如，R6、R7和C10、C8整齐排列，使整个电路板美观。
4. 在排列放置元件时，要考虑到实际安装的要求，避免把针脚画反导致需要用线搭接。

3、主要挑战

- 封装：元件封装需要耐心和细心，为了取得良好的效果，我首先在库中寻找符合规格的封装，对封装间隔和引脚对不上的元件，仔细一个个测量修改。
- 布局：许多元件需要额外留空间才能放置，如主控板插针需要布局在PCB板外侧，突出的地方可以留在外部减少空间， LCD屏幕的位置需要预留出左右两侧较多的空间防止元件与屏幕干涉。

4、个人总结和经验感受

通过两天半的PCB电路板训练，从PCB原理图、封装、原理图库、PCB库、PCB布局、布线、覆铜，通过立创平台提交文件和订单，收到PCB板并运用到小车上，我学习了电路板设计制作和使用的流程。对PCB的经验总结，就是要仔细地对元器件进行封装和检查，在设计时考虑布局的合理性。立创平台是功能强大的SaaS平台，使用非常方便，既有丰富的元件库，而且操作简单便捷，无需额外安装。这两天半学到的技能对日后参加比赛、制作电子器件有很大的帮助。

二、MSP430单片机部分

1、作业1

读取 MSP430G2553 LaunchPad 上S2的按键状态，并用该按键控制LED1。按键按下时让LED 1亮起，按键松开时让LED 1熄灭。

代码 1: 作业1

```
#include <msp430.h>
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer

    P1DIR |= BIT0;
    P1DIR &= ~BIT3;
    P1OUT|=BIT3;
    P1REN |= BIT3;

    while(1)
    {
        if(!(BIT3 & P1IN))
            P1OUT |= BIT0;
        else
            P1OUT &= ~BIT0;
```

```
    }
    return 0;
}
```

2.1.1 现象和讨论

MSP430G2553 LaunchPad 上 S2按键按下时让 LED 1亮起，按键松开时让 LED 1熄灭。按键S2对应P1.3端口，设置P1.3为输入，LED1对应P1.0端口，设置P1.0为输出，使得按下S2时， P1.0输出为0，让灯亮，松开时，P1.0输出为1，让灯灭。

在实现上，由于该程序比较简单，没有复杂的需求，对资源管理的要求不高，所以直接轮询而不是使用中断。但这种方式可能浪费CPU资源，在更复杂的程序中应该使用中断。

2、作业2

在上一节Blink程序的基础上，将MCLK分别设置为1MHz和8MHz 并观察LED1闪烁的频率有何变化。

代码 2: 作业2

```
#include <msp430g2553.h>

void set_1mhz()
{
  BCSCTL1 = CALBC1_1MHZ; // Set range

  DCOCTL = CALDCO_1MHZ;

  BCSCTL2 &= ~(DIVS_3); // SMCLK = DCO = 1MHz
}

void set_8mhz()
{
  BCSCTL1 = CALBC1_8MHZ; // Set range

  DCOCTL = CALBC1_8MHZ;

  BCSCTL2 &= ~(DIVS_3); // SMCLK = DCO = 1MHz
}

int main()
{
  WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
  P1DIR |= 0x01;            // configure P1.0 as output

  set_1mhz();
  //set_8mhz();

  volatile unsigned int i; // volatile to prevent optimization

  while (1)
  {
    P1OUT ^= 0x01; // toggle P1.0
```

```
35      for (i = 10000; i > 0; i--)
36        __delay_cycles(10); // delay
37    }
38    return 0;
39  }
```

2.2.1 现象和讨论

通过对比将主时钟设为1MHz和8MHz的情况下，能明显感觉到8MHz下LED1闪烁频率比1MHz快。通过改变主时钟频率来改变运行速度从而改变时钟的频率。

3、作业3

通过GPIO中断的方式，用两个按键分别控制两盏不同的LED灯。每按下一次按键，相应的LED灯改变一次亮灭状态。提示：板子上只有一个按键S2，可以用杜邦线一端连接在GND或者VCC，另外一端触碰下自己选择的IO口，模拟按键按下的状态。

代码 3: 作业3

```
1
2   #include <msp430.h>
3   #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
4   #pragma vector=PORT1_VECTOR
5   __interrupt void Port_1(void)
6   #elif defined(__GNUC__)
7   void __attribute__((interrupt(PORT1_VECTOR))) Port_1(void)
8   #else
9   #error Compiler not supported!
10  #endif
11  {
12      if(!(P1IN & BIT4))
13      {
14          P1OUT ^= BIT0;
15          P1IFG &= ~BIT4;
16      }
17      if(!(P1IN & BIT5))
18      {
19          P1OUT ^= BIT6;
20          P1IFG &= ~BIT5;
21      }
22
23  }
24
25  int main(void)
26  {
27      WDTCTL = WDTPW + WDTHOLD;   // stop watchdog timer
28      P1DIR |= BIT0;
29      P1OUT &= ~BIT0;
30
31      P1DIR |= BIT6;
32      P1OUT &= ~BIT6;
33
34      P1DIR &= ~BIT4;
35      P1OUT != BIT4;
```

```
    P1REN |= BIT4;

    P1DIR &= ~BIT5;
    P1OUT != BIT5;
    P1REN |= BIT5;

    P1IES |= BIT4;
    P1IFG &= ~BIT4;
    P1IE |= BIT4;

    P1IES |= BIT5;
    P1IFG &= ~BIT5;
    P1IE |= BIT5;

    __bis_SR_register(GIE);
    return 0;
}
```

4、作业4

编程实现：利用定时器编写呼吸灯。所谓呼吸灯是指LED在一个周期内先逐渐变亮，再逐渐变暗。

代码 4: 作业4

```

#include <msp430g2553.h>

int IncDec_PWM = 1;

int main(void)
{

  /*** Watchdog timer and clock Set-Up ***/
  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
  DCOCTL = 0;               // Select lowest DCOx and MODx
  BCSCTL1 = CALBC1_1MHZ;    // Set range
  DCOCTL = CALDCO_1MHZ;     // Set DCO step + modulation

  P1DIR |= BIT6;
  P1SEL |= BIT6;

  /*** Timer0_A Set-Up ***/
  TA0CCR0 |= 1000;           // PWM period
  TA0CCR1 |= 1;              // TA0CCR1 PWM duty cycle
  TA0CCTL1 |= OUTMOD_7;      // TA0CCR1 output mode = reset/set
  TA0CTL |= TASSEL_2 + MC_1; // SMCLK, Up Mode (Counts to TA0CCR0)

  /*** Timer1_A Set-Up ***/
  TA1CCR0 |= 2000;           // Counter value
  TA1CCTL0 |= CCIE;          // Enable Timer1_A interrupts
  TA1CTL |= TASSEL_2 + MC_1; // SMCLK, Up Mode (Counts to TA1CCR0)

  _BIS_SR(LPM0_bits + GIE); // Enter Low power mode 0 with interrupts enabled

  return 0;
}
```

```
33
34 #pragma vector = TIMER1_A0_VECTOR // Timer1 A0 interrupt service routine
35 __interrupt void Timer1_A0(void)
36 {
37
38   TA0CCR1 += IncDec_PWM * 2;
39   if (TA0CCR1 > 998 || TA0CCR1 < 2)
40     IncDec_PWM = -IncDec_PWM;
41 }
```

5、作业5

编写接收程序和发送程序，当开发板串口接收到PC机发来的字符“1”时，点亮LED1，并向PC机发送“LED_ON”；当当开发板串口接收到PC机发来的字符“0”时，熄灭LED1，并向PC机发送“LED_OFF”；当收到其它字符时，翻转LED1状态，并向PC机发送“LED_ON”或者“LED_OFF”，表明LED1当前的状态。

代码 5: 作业5

```
1
2  #include <msp430g2553.h>
3  #include <stdint.h>
4  #include <stdbool.h>
5
6  void set_1mhz()
7  {
8    BCSCTL1 = CALBC1_1MHZ; // Set range
9
10   DCOCTL = CALDCO_1MHZ;
11 }
12
13 void uart_init()
14 {
15   UCA0CTL1 |= UCSSEL_2;
16   UCA0BR0 = 104;
17   UCA0BR1 = 0;
18   UCA0MCTL = UCBRS0;
19   UCA0CTL1 &= ~UCSWRST;
20   IE2 |= UCA0RXIE;
21 }
22
23 void uart_puts(char *c)
24 {
25   while (*c)
26   {
27     while (!(IFG2 & UCA0TXIFG))
28     {
29       __nop();
30     }
31     UCA0TXBUF = *c++;
32   }
33 }
34
35 void __attribute__((interrupt(USCIABORX_VECTOR))) uart_rx_isr()
36 {
```

```
  volatile int led_on = !!UCA0RXBUF;

  if (led_on)
  {
    P1OUT |= BIT0;
    uart_puts("LED ON\n");
  }
  else
  {
    P1OUT &= ~BIT0;
    uart_puts("LED OFF\n");
  }
}

int main()
{
  WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

  P1DIR |= BIT0; // configure P1.0 as output
  P1OUT &= ~BIT0;

  DCOCTL = 0;
  set_1mhz();

  P1SEL = BIT1 | BIT2;
  P1SEL2 = BIT1 | BIT2;

  uart_init();

  __bis_SR_register(LPM0_bits | GIE);
  return 0;
}
```

6、作业6

按如下格式在OLED屏上显示自己的姓名、学号和系别信息。

代码 6: 作业6

```

#include "msp430g2553.h"
#include"I2C_OLED.H"
#include"zimo.h"

#define LEN(arr) (sizeof(arr)/sizeof(arr[0]))

int name_indexes[]={1,2,7,8,9};
int major_indexes[]={5,6,10,11,12,13,14};
int sid[]={2,2,9,2,0,2,0,2,2,0,4,6,2,2};

int main(void)
{
   system_clock();
   I2C_OLED_Init();
   while(1)
   {
```

```
18              OLED_All(0);
19  //          delay_ms(500);
20
21
22              volatile int i=0,cnt=0;
23              for(i=0;i<LEN(name_indexes);i++)
24              {
25                  OLED_P16x16Ch((cnt++)*16,0,name_indexes[i]);
26              }
27
28              i=cnt=0;
29              for(i=0;i<LEN(major_indexes);i++)
30              {
31                  OLED_P16x16Ch((cnt++)*16,3,major_indexes[i]);
32              }
33
34              OLED_P16x16Ch(0,6,3);
35              OLED_P16x16Ch(16,6,4);
36              OLED_P6x8Str(32,6,"22920202204622");
37
38              for(;;);
39      }
40  }
```

三、基于MSP430的智能小车行驶

1、电路与程序设计

3.1.1 需求分析

图 5: 小车赛道

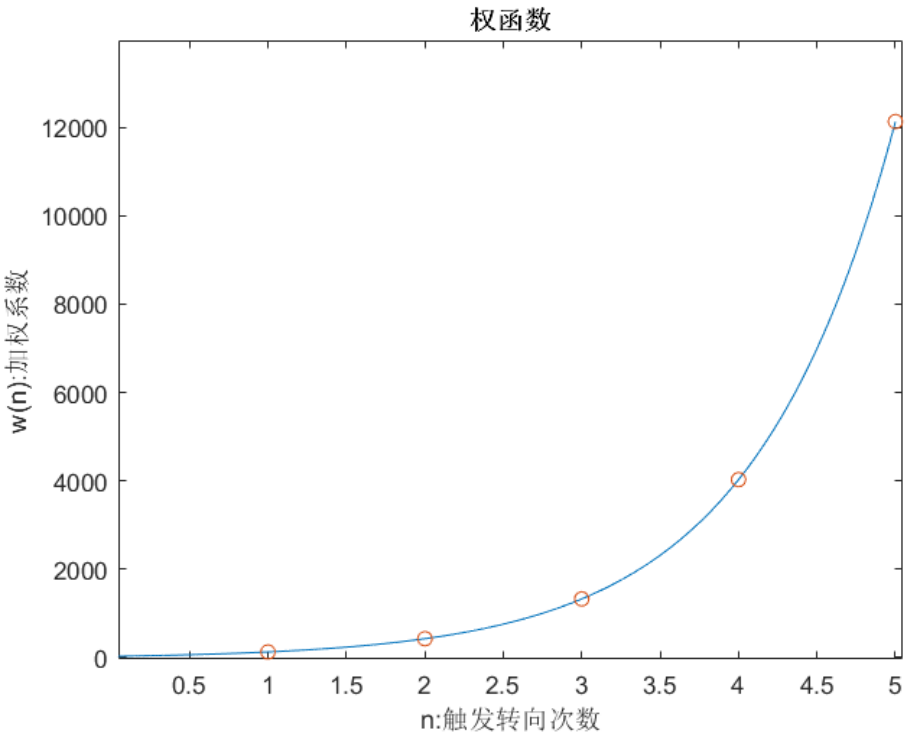
(a) 赛道示意图

(b) 轨迹示意图

表格 2: 加权函数

权函数类型	表达式
余弦型函数	$y(k;n)=\cos(nk\pi)$
多项式型函数	$y(k;n)=a_nk^n+a_{n-1}k^{n-1}+\cdots+a_1k+a_0$
指数型函数	$y(k;a,m,n)=m\cdot a^k+n$

图 6: 函数图像



3.1.2 设计思路

3.1.3 非对称的转向参数

3.1.4 递增的转向速度加权

$$y(k;a,m,n)=m\cdot a^k+n \tag{1}$$

$$y(k)=a\cdot y(k-1)+b \tag{2}$$

3.1.5 方案总结

2、测试方案与测试结果

3、本人所做的工作

4、经验总结与个人感受

四、人工智能入门

五、实验改进建议

六、实训总结