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

1.1 PCB设计过程及遇到的主要问题

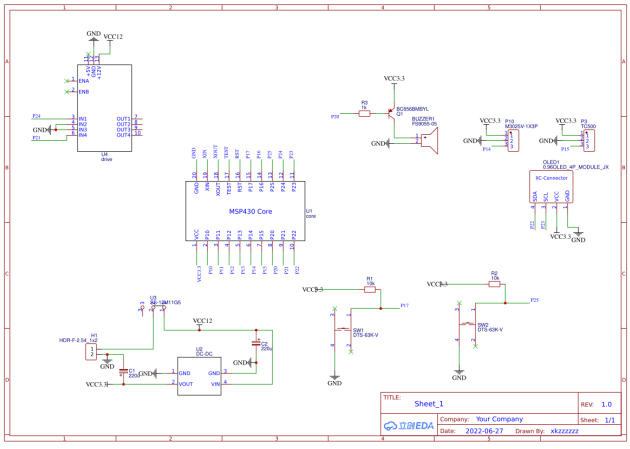
PCB制作过程和遇到的问题如表 1。

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 3。实物图如图 4。

图 2: 封装

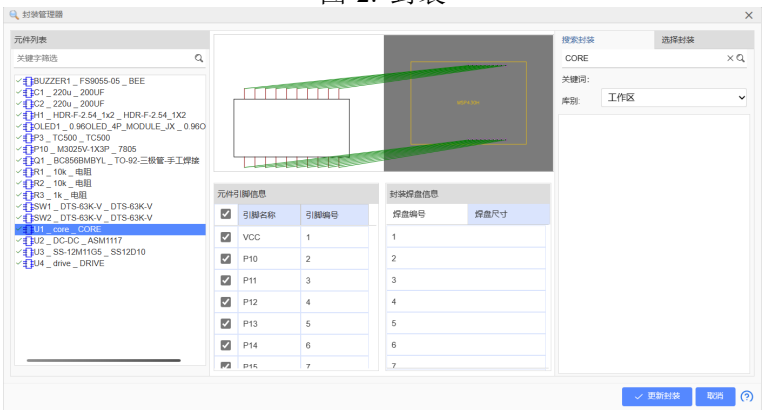
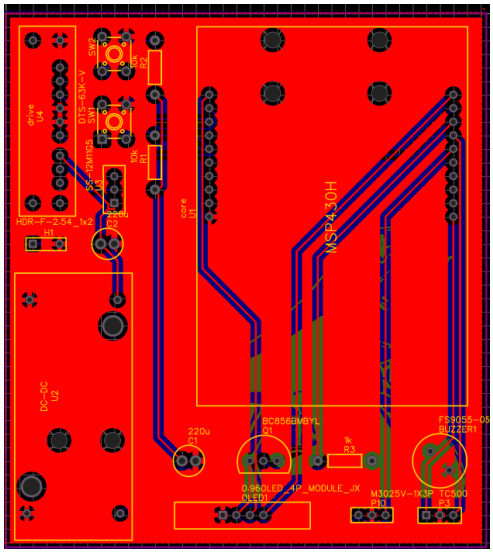
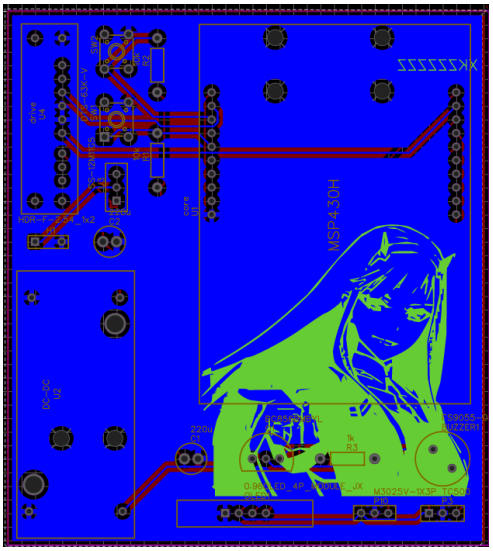


图 3: PCB



(a) 顶部



(b) 底部

图 4: 实物图

1.2 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。

1.3 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。

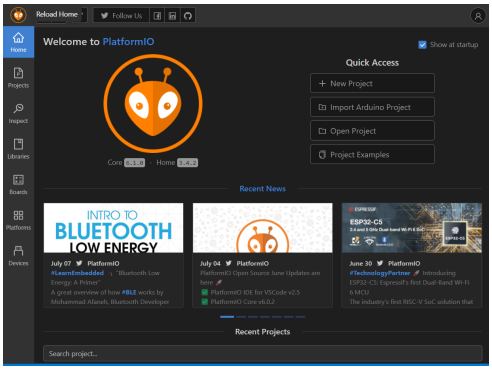
1.4 个人总结和经验感受

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。

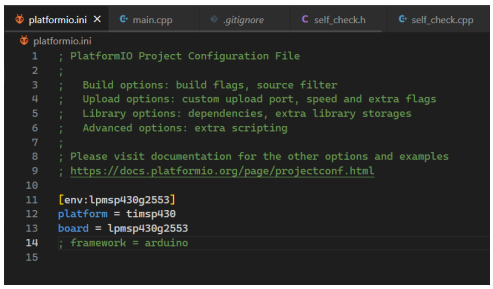
二、MSP430单片机部分

CCS作为一款IDE在2020年代的视角下看是不合格的。它没有良好的代码智能提示和重构功能，也不能很好地支持源代码版本控制工具，因此我使用VSCode+PlatformIO这一新平台完成了代码的编写这款插件通过配置相应的选项可以完美地兼容MSP430G2553平台。如图 5

图 5: PlatformIO



(a) PlatformIO启动页



(b) 相关配置文件

2.1 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

```c
#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。

2.2 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

```c

#include <msp430g2553.h>

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

  DCOCTL = CALDCO_1MHZ;

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

```

```c
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
    for (i = 10000; i > 0; i--)
      __delay_cycles(10); // delay
  }
  return 0;
}
```

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，能明显发现到8MHz下LED1闪烁频率比1MHz快。可见，主时钟频率越大、每个时钟周期的时间就越短，相对来说运行就越快，但功耗也越多。

2.3 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

```c

#include <msp430.h>
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
#elif defined(__GNUC__)
void __attribute__((interrupt(PORT1_VECTOR))) Port_1(void)
#else
#error Compiler not supported!
#endif
{
    if(!(P1IN & BIT4))
    {
```

```
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;
36      P1REN |= BIT4;
37
38      P1DIR &= ~BIT5;
39      P1OUT != BIT5;
40      P1REN |= BIT5;
41
42      P1IES |= BIT4;
43      P1IFG &= ~BIT4;
44      P1IE |= BIT4;
45
46      P1IES |= BIT5;
47      P1IFG &= ~BIT5;
48      P1IE |= BIT5;
49
50      __bis_SR_register(GIE);
51      return 0;
52  }
```

2.3.1 现象和讨论

将C1接到单片机的GND端，R1对应开关S1接单片机P1.4端口，R2对应开关S2接单片机P1.5端口用来触发中断。当对应开关（第一列前两行）按下时，对应端口由输出高电平变为输出低电平，对应LED灯就会改变一次亮灭状态，

2.4 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

```
1
2   #include <msp430g2553.h>
```

```
3
4  int IncDec_PWM = 1;
5
6  int main(void)
7  {
8
9    /*** Watchdog timer and clock Set-Up ***/
10   WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
11   DCOCTL = 0;               // Select lowest DCOx and MODx
12   BCSCTL1 = CALBC1_1MHZ;    // Set range
13   DCOCTL = CALDCO_1MHZ;     // Set DCO step + modulation
14
15   P1DIR |= BIT6;
16   P1SEL |= BIT6;
17
18   /*** Timer0_A Set-Up ***/
19   TA0CCR0 |= 1000;           // PWM period
20   TA0CCR1 |= 1;              // TA0CCR1 PWM duty cycle
21   TA0CCTL1 |= OUTMOD_7;      // TA0CCR1 output mode = reset/set
22   TA0CTL |= TASSEL_2 + MC_1; // SMCLK, Up Mode (Counts to TA0CCR0)
23
24   /*** Timer1_A Set-Up ***/
25   TA1CCR0 |= 2000;           // Counter value
26   TA1CCTL0 |= CCIE;          // Enable Timer1_A interrupts
27   TA1CTL |= TASSEL_2 + MC_1; // SMCLK, Up Mode (Counts to TA1CCR0)
28
29   _BIS_SR(LPM0_bits + GIE); // Enter Low power mode 0 with interrupts enabled
30
31   return 0;
32 }
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 }
```

2.4.1 现象和讨论

通过实验现象可以发现LED在一个周期内先逐渐变亮再逐渐变暗。通过不断设置CCR1的值不断变大，调节PWM占空比，使得LED灯逐渐变亮，然后当达到最亮后，通过设置CCR1的值不断减小，，调节PWM占空比，使LED灯逐渐变暗。

2.5 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(USCIAB0RX_VECTOR))) uart_rx_isr()
36 {
37   volatile int led_on = !!UCA0RXBUF;
38
39   if (led_on)
40   {
41     P1OUT |= BIT0;
42     uart_puts("LED ON\n");
43   }
44   else
45   {
46     P1OUT &= ~BIT0;
47     uart_puts("LED OFF\n");
48   }
49 }
50
51 int main()
52 {
53   WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
54
```

```
55    P1DIR |= BIT0; // configure P1.0 as output
56    P1OUT &= ~BIT0;
57
58    DCOCTL = 0;
59    set_1mhz();
60
61    P1SEL = BIT1 | BIT2;
62    P1SEL2 = BIT1 | BIT2;
63
64    uart_init();
65
66    __bis_SR_register(LPM0_bits | GIE);
67    return 0;
68  }
```

2.5.1 现象和讨论

打开串口助手，当开发板串口接收到PC机发来的字符“1”时，LED1亮，并向PC机发送“LED_ON”；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。函数*uart_puts*用来向PC机发送字符串，中断函数接受PC机发来的字符，字符为0则灯灭，为1则灯亮。

2.6 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

```
1
2   #include "msp430g2553.h"
3   #include"I2C_OLED.H"
4   #include"zimo.h"
5
6   #define LEN(arr) (sizeof(arr)/sizeof(arr[0]))
7
8   int name_indexes[]={1,2,7,8,9};
9   int major_indexes[]={5,6,10,11,12,13,14};
10  int sid[]={2,2,9,2,0,2,0,2,2,0,4,6,2,2};
11
12  int main(void)
13  {
14     system_clock();
15     I2C_OLED_Init();
16     while(1)
17     {
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  }
```

2.6.1 现象和讨论

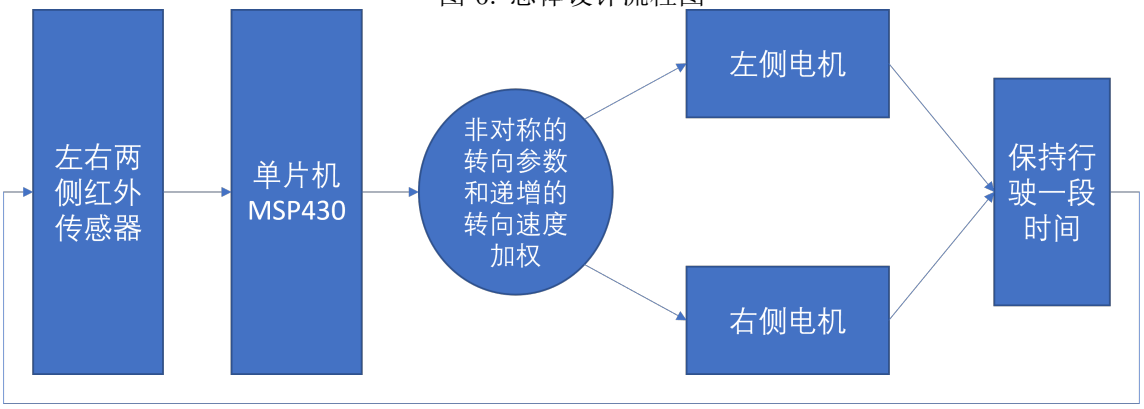
通过字模软件，取出自己的姓名、系别、学号的字模，然后填写到codetab.h文件中，同时在main.c函数中编辑显示的位置。一个汉字的宽度是16，高度值是2。字母和数字的宽度则是8。由于学号较长，所以使用较小的字体。

三、基于MSP430的智能小车行驶

3.1 电路与程序设计

3.1.1 需求分析和设计思路

图 6: 总体设计流程图



总体设计如图 6。题目要求我们实现能够在地图上右转并停止的智能小车，时间可以在10s-20s之间调整。示意图如图 7(a)。根据题目要求，我们需要实现转向功能。由于我们只有两个传感器，不能通过测量两次黑块的时间间隔获取速度信息，因此我们退而求其次，将车轮可能的转速分为10档，通过试验测定挡位与用时的关系。

由于电机和轮子的设计存在公差、导线长度等原因，两侧速度不能完全一致，因此实际轨迹可能如图 7(b) 所示具有较多微调转弯和少量改变方向的转弯。由于红外传感器只能分辨黑色/非黑色，不能区分白色与木板底色，在改变方向时由于两传感器都从白块穿过而跑出赛道的情况时有发生。因此需要设计更加稳定的、更加容错的转向方法。

图 7: 小车赛道

(a) 赛道示意图 (b) 轨迹示意图

3.1.2 非对称的转向参数

为了进行转向功能的具体实现，我们首先从需求出发进一步分析。由于赛道本身只向右转，因此所有的左转都是“微调转向”，而右转包括“微调转向”和“变向转向”，这意味着左转的需求是微小的转向，而右转的转向则需要兼顾小转向和大转向的需要。

根据这一特征，我们可以对小车的控制加入归纳偏置来提高系统的稳定性。具体而言，我们对左转右转的控制采用了非对称的参数结构。如表 2。

表格 2: 左转右转的参数

	左转	右转
参数类型	固定	可变
转向轮速	左侧为右侧的80%以上	右侧为左侧的2%以下

3.1.3 递增的转向速度加权

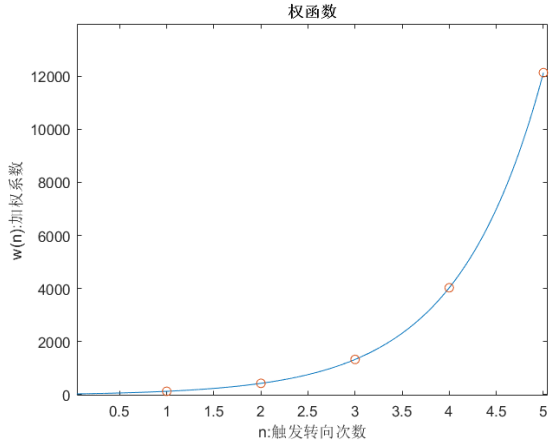
表格 3: 加权函数

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	$y(k;a,m,n)=m\cdot a^k+n$

为了适应转向大小的需求在行驶过程中的变化，我们对每一次转向进行动态的加权。我们设计了如表 3的加权函数来对每一次转弯进行加权。根据我们对行驶过程的观察，我们选择了指数加权的方式，主要有以下原因：

- 指数型函数的图像如图 8。这符合转向需要逐步增大的需要。
- 指数函数可以用如(1)的形式递推地通过多项式实现，可以减少进行复杂运算的需要。由于MSP430较弱的硬件性能，这能大大提高计算效率，防止由于计算过于缓慢导致小车行驶不正常。

图 8: 函数图像



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

经过这种加权处理，小车的转弯幅度依次增大，由于我们小车传感器右偏的设计，经过90°弯道之后几乎不会触发转弯，因此可以达到稳定转向的目的。

同时，由于初期转向有

1. 加权值较小
2. 权函数增长缓慢

的特点，小车在行驶初期较为稳定，避免了冲出赛道的事故发生。我们基于实践经验，将(1) 中的参数设置为$a = 3, b = 40$，获得了较好的效果。

3.1.4 方案合理性分析

为了对选择一特定函数作为加权的方案能够产生很好的效果这一点给出合理的解释，考察PID控制的公式(2)，

$$u(n) = K_p\{e(n) + \frac{T}{T_I}\sum_{i=0}^{n} e(i) + \frac{T_D}{T}[e(n) - e(n-1)]\} + u_0 \tag{2}$$

该公式要求

1. 对误差$e(n)$有一个较为准确的估计
2. 积分项$\frac{T}{T_I}\sum_{i=0}^{n} e(i)$需要保存前$n+1$次的误差值
3. 参数的计算涉及到浮点数的乘除

很显然，由于只有两个红外传感器，要求1十分困难，甚至几乎不可能完美的实现；由于MSP430极为有限的性能约束，要求2十分不现实，要求3很可能降低性能，然而积分控制是实现稳定的控制的必要要求，这产生了要求和能力之间的矛盾。

但是在现实环境中考量这一公式，对于一个长度不太长、变换不太大的赛道，转向的控制次数应当不太多，这意味着转向控制量的值的变化大概率能够使用一个简单的函数$w(n)$来进行相当精确的近似。即(3)。

$$u(n) \approx K_p \cdot w(n) \cdot u_0 \tag{3}$$

这里的$w(n)$就是我们选择的加权函数。这个加权函数的效果就是同时近似比例项、积分项、微分项。依赖于*转向次数不太多*这一假设，这样的近似可以是非常精确的。

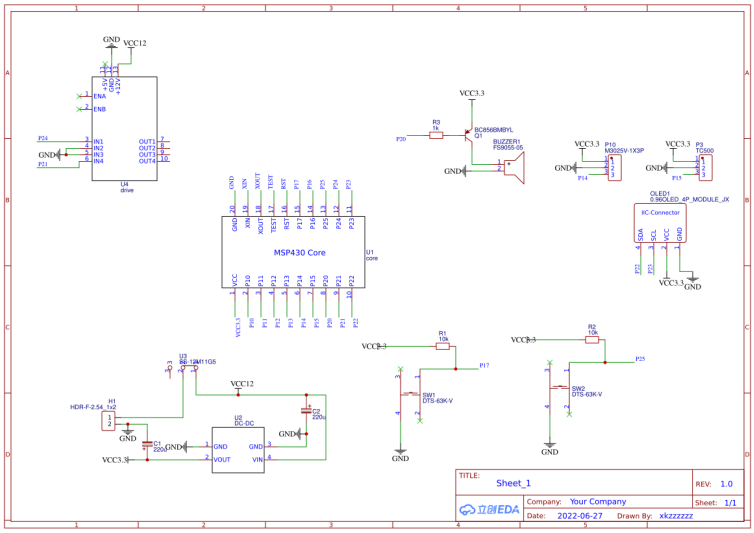
这种方案将能够解决上述提出的大多数困难。事实上，依赖于我们选择的指数型加权函数的递推形式，我们实现了移除所有的浮点数除法运算，将加法乘法的计算量和对过往状态的存储量都降低到了常数级（*与转向次数n无关*），这带来了可观的性能提升。

事实上，在只有两个传感器的情况下，我认为任何PID的实现都一定会引入一些（可能是隐式的）来自观察和试验的假设。而我们的工作将这中假设进行了有效的形式化，大大提升了控制程序的可解释性，更加方便了选择、设计、分析与调试。虽然我们的方案在一定程度上牺牲了泛用性，引入了转向次数不太多这一较强的假设，但是这种方案有相对完整的数学解释，并且在实际测试中有良好的表现。

3.1.5 电路设计

电路设计如图 9。系统组成如表 4

图 9: 原理图



表格 4: 系统组成

显示模块	OLED液晶显示
供电方式	3节18650锂电池
输入方式	红外循迹、按键
输出方式	小车（2驱动+1万向轮）OLED显示屏蜂鸣器
控制芯片	MSP430单片机（MSP430G2 launchpad）
制作工艺	印制电路板

3.2 测试方案与测试结果

1. **功能测试：**首先对小车的基本功能进行测试，一开始发现小车的电机运转方式奇怪而没有规律。但在按任何单片机外的按键的时候都会恢复正常。我们对小车的电路进行排查，由于按键会向单片机引脚输入低电平（接地），我们怀疑单片机接地不良引发了问题，因此进行测量，果然发现单片机接地处电位不正常，导致了单片机GPIO输出不正常。重新对GND对应的针脚进行焊接之后测量电位正常，测试小车工作正常。在测试场地调整红外传感器上的电位器后小车基本正常工作。

2. **稳定测试：**为了测试小车运行的稳定性，我们调整初始放置的角度进行测试，发现小车都能正确调整方向回到轨道。因此小车在方向上有较好的稳定性。同时调整赛道坡道进行测试，发现坡道角较大的时候由于小车侧滑，传感器无法扫描到停止线因此无法正常停止。我们因此调整了传感器安装的位置，解决了这个问题。

3. **挡位-时间测试：**为了控制速度，我们在设计程序时加入了10个速度级别，因此需要对速度级别对应的时间进行测量。进行计时之后得到了表 5。

表格 5: 时间测量

挡位	时间
10	20s
12	17s
14	15s
16	12s
18	10s

根据测量结果，我们能灵活调节小车的行驶时间。

3.3 本人所做的工作

在小车项目中我完成了程序的设计、测试和参数的调整。

3.4 经验总结

1. 设计PCB板要考虑到安装的合理性。由于设计的有些紧密，我们安装LED屏幕时需要使用线接出来，造成了一定的不便。

2. 在安装元件时要注意元件之间的相互关系。我们在焊接供电插口的时候由于没有注意插口朝向，与其他元件产生了干涉，不得不拆下来，由于残留焊锡堵住焊孔清理十分麻烦而处理了很长时间。

3. 在进行调整的时候要做到统筹兼顾。在调试时不能只对代码进行修改，也要前瞻性地设计能够方便利用按钮等调整的参数，以及通过微调小车的结构设计、传感器的间距、位置等根据需求进行恰当的统筹安排。

3.5 个人感受

通过这次制作小车的课题，我们在一个多星期的实训里收获颇丰。

一是对电子工艺设计中的理论和实际应用有了新理解。在实训过程中，我们需要对电路进行恰当的连接与设计，进而形成完整的小车结构，最后进行单片机的编程。这一系列方法既要求全面的思考，也要求细致的实现，更要求对相关知识的理解与掌握。通过这一过程，我们不仅练习了嵌入式编程，更掌握了一些电路板设计和控制论中的理论知识，还对这些知识的实际应用有了更深刻的体会。

二是团队协作和分工的能力得到了新提升。从设计，到电路板的选择和小车的焊接，到对参数调节和改进的一步步尝试，再到最后形成完整的通过验收的小车，团队协作和分工贯穿始终。从小车的设计到实际实现的代码编写、调整，都离不开分工和合作。在实训中，我们很好地锻炼了团队协作和分工的能力，让我们能够在今后的学习与生活中更好地与他人实现合作和交流。

三是实际编写代码实现想法和设计的能力得到了新锻炼。正所谓“力行而后知之真”，在我们确定了和其他组不一样的控制方式之后，必须通过编写代码将理论和设计的方案落到实处才能顺利完成实训

的任务。在这次实训中，我们使用PlatformIO这一新平台完成了代码的编写，这既涉及到对新环境的配置，也涉及到对我们设计的方案的正确实现。通过这次实训，我们培养的这些能力对以后的学习、竞赛乃至今后的工作都有着重要的意义。

四、人工智能入门

五、实验改进建议

附录：小车控制行驶参数部分代码

小车行驶参数控制（速度与转向）部分的代码如 7

代码 7: 作业6

```
1
2  #define TURN_DELAY 200
3
4  int turn_constant = 30;
5  const int turn_multipier = 3;
6  const int turn_bias = 40;
7  #define TURN_MAX 120
8
9  int delaytime;
10 void go()
11 {
12   P2DIR |= (BIT1 + BIT4);
13   P2SEL |= (BIT1 + BIT4);
14   TA1CCR0 = 20000;
15   TA1CCTL1 = OUTMOD_7;
16   TA1CCR1 = 0;
17   TA1CCTL2 = OUTMOD_7;
18   TA1CCR2 = 0;
19
20
21   TA1CTL = (TASSEL1 + ID0 + ID1 + MC0);
22
23   while (1)
24   {
25     OLED_ShowChar(80, 0, t_miao / 100 + '0');
26     OLED_ShowChar(88, 0, t_miao % 100 / 10 + '0');
27     OLED_ShowChar(96, 0, t_miao % 10 + '0');
28
29     if (hy2 != 0 && hy1 != 0)
30     {
31       TA1CCR1 = 0;
32       TA1CCR2 = 0;
33       oled_puts(0, 0, "end ");
34
35       while (1)
36       {
37         FM_L;
38         delay_ms(100);
39         FM_H;
40         delay_ms(100);
41       }
42     }
43     else
44     {
45       if (hy1 == 0)
46       {
47         oled_puts(0, 2, "hy1 detect    ");
48       }
49       else
50       {
```

```
51          oled_puts(0, 2, "hy1 not detect");
52        }
53
54        if (hy2 == 0)
55        {
56          oled_puts(0, 4, "hy2 detect     ");
57        }
58        else
59        {
60          oled_puts(0, 4, "hy2 not detect");
61        }
62
63        //  setmode=10-20  100-300 R
64        if (hy1 == 0 && hy2 != 0)
65        {
66          TA1CCR2 = (int)turn_constant * (100 + 20 * (setmode - 10));
67
68          turn_constant = turn_multipier * turn_constant + turn_bias;
69          if (turn_constant > TURN_MAX)
70          {
71            turn_constant = TURN_MAX;
72          }
73
74          if (setmode <= 13)
75          {
76            TA1CCR1 = (int)10.5 * 30 * 0.2;
77            delay_ms(TURN_DELAY);
78          }
79          else if (setmode <= 15)
80          {
81            TA1CCR1 = (int)10.5 * 50 * 0.2;
82            delay_ms(TURN_DELAY);
83          }
84          else if (setmode <= 17)
85          {
86            TA1CCR1 = (int)10.5 * 60 * 0.2;
87            delay_ms(TURN_DELAY);
88          }
89          else
90          {
91            TA1CCR1 = (int)10.5 * 70 * 0.2;
92            delay_ms(TURN_DELAY);
93          }
94          delay_ms(30);
95        }
96        else if (hy2 == 0 && hy1 != 0)
97        {
98          volatile int speed = 30.5 * (100 + 20 * (setmode - 10));
99          TA1CCR1 = (int)speed;
100
101         if (setmode <= 13)
102         {
103           TA1CCR2 = (int)speed;
104           delay_ms(30);
105         }
```

```
106         else if (setmode <= 16)
107         {
108           TA1CCR2 = (int)speed * 0.8;
109           delay_ms(30);
110         }
111         else
112         {
113           TA1CCR2 = (int)speed * 0.7;
114           delay_ms(30);
115         }
116       }
117       else if (hy1 == 0 && hy2 == 0)
118       {
119         TA1CCR1 = (int)30.5 * (100 + 20 * (setmode - 10));
120         TA1CCR2 = (int)32 * (100 + 20 * (setmode - 10));
121       }
122     }
123   }
124 }
```