

一、概要

比赛规则与赛道：

比赛没有赛道，只有电磁线。但赛道元素包括有直道、弯道、坡道、十字路口以及横断路障。室外电磁组原则上选择室外的马路、草坪、体育场组织比赛，场地内可能会存在高度不大于 2 厘米的硬质路坎、沙坑、深度不超过 2 厘米的水坑等。

选手制作的车模完成赛道运行一周。比赛时间从车模冲过起跑线到重新回到起跑线为止。

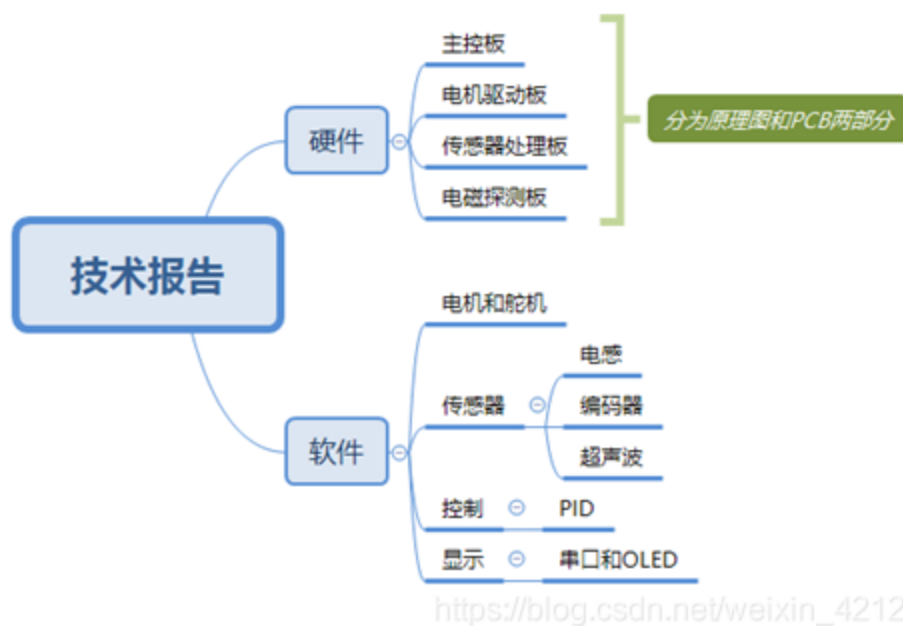
越野车外形



车的功能与实现：



本文脉络



二、硬件部分：

1.主控板

(1) 原理图：

K60引脚分配：

引脚通过查K60引脚功能来分配，除去特殊功能引脚外，其余引脚为普通I/O口

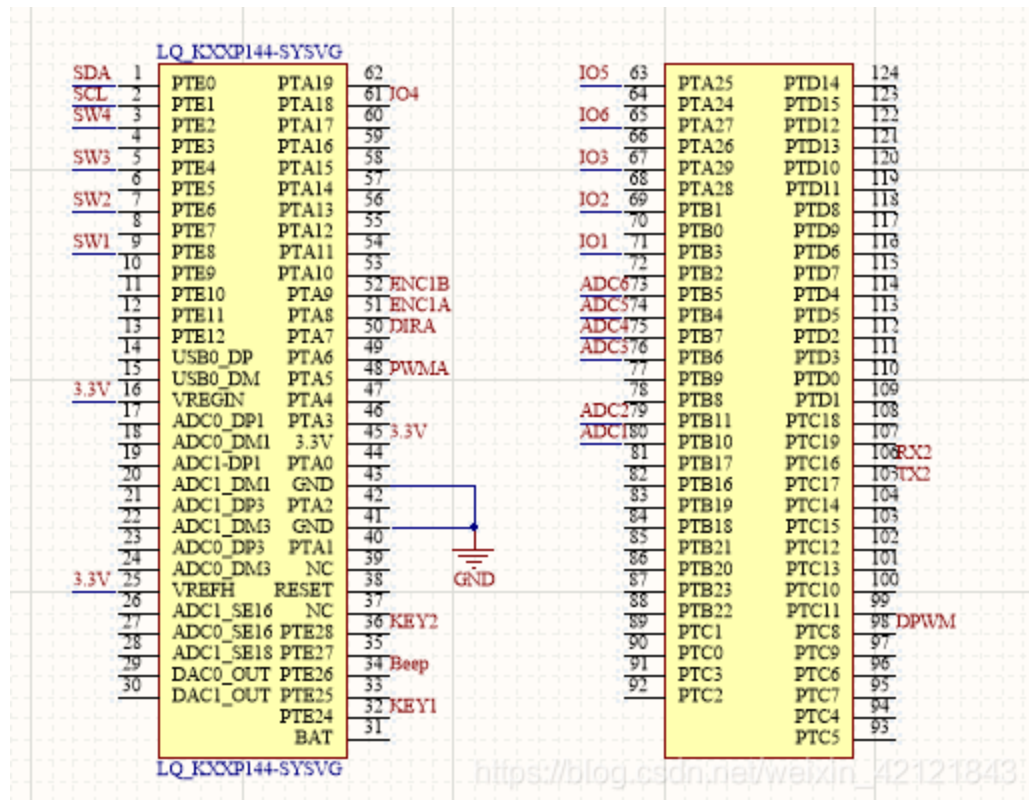
特殊引脚:

I2C通信: SDA, SCL

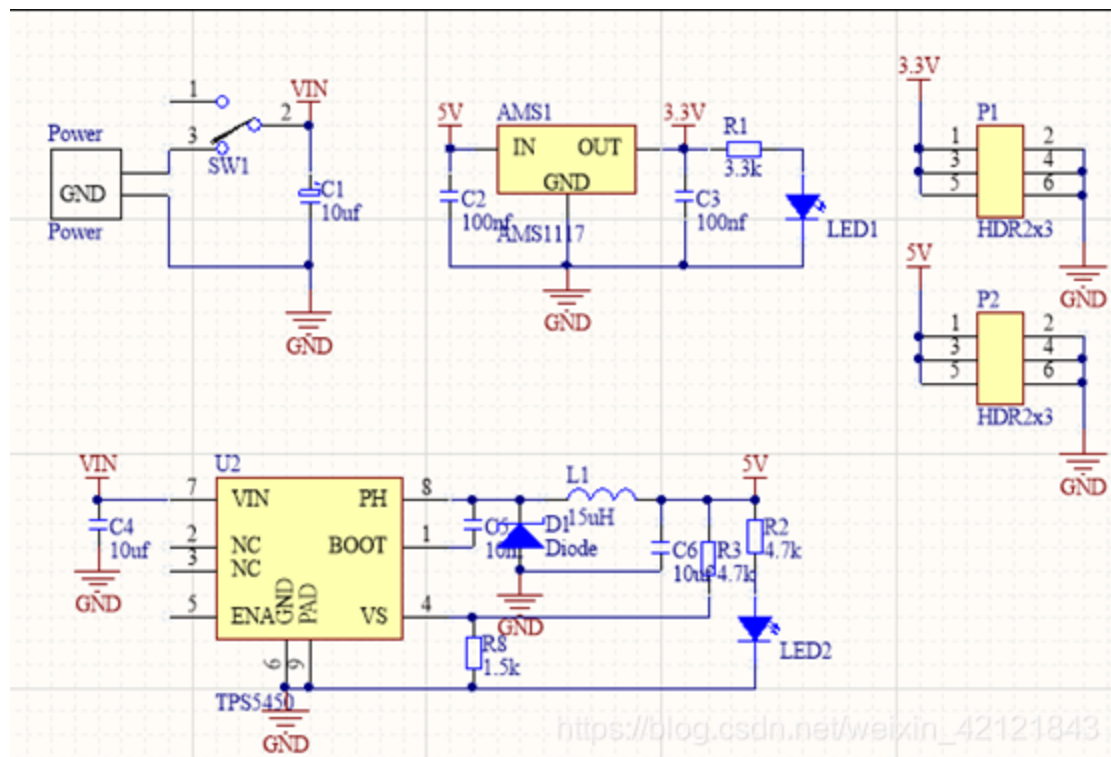
FTM (PWM输出): PWMA, DPWM

AD (模数转换): AD1~AD6

串口通信: TX2, RX2



电源部分及笔记: (7~8v输入, 5v和3.3v输出)



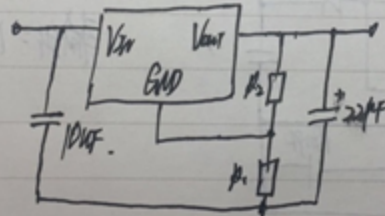
以下为原理笔记:

AMS1117 稳压芯片:

固定电压输出:



可调输出电压:



参数:

max: 20V

V_{IN} : 正常: 15V

V_{out} : AMS1117-3.3: 3.3V

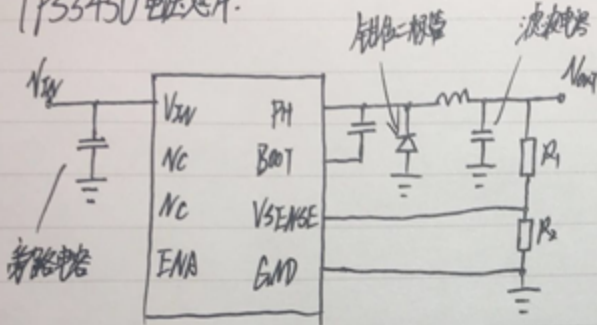
AMS1117-2.5: 2.5V

I_q : 10mA (max)

5mA (正常)

https://blog.csdn.net/weixin_42121843

TPS5450 电压芯片:



BOOT: BOOT与PH间连一个0.01µF的陶瓷电容, 用于提供高侧MOSFET栅极驱动电压

VSENSE: 反馈, 接输出电压的分压

ENA: 使能端, 低于0.5V不工作, 是定时ENABLE.

GND和PowerPAD接地

V_{IN} : -0.3~40V

BOOT: -0.3~50V

PH: -0.6~40V

ENA: -0.3~7V

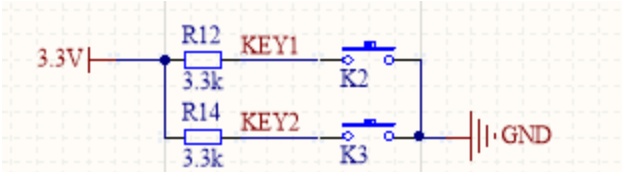
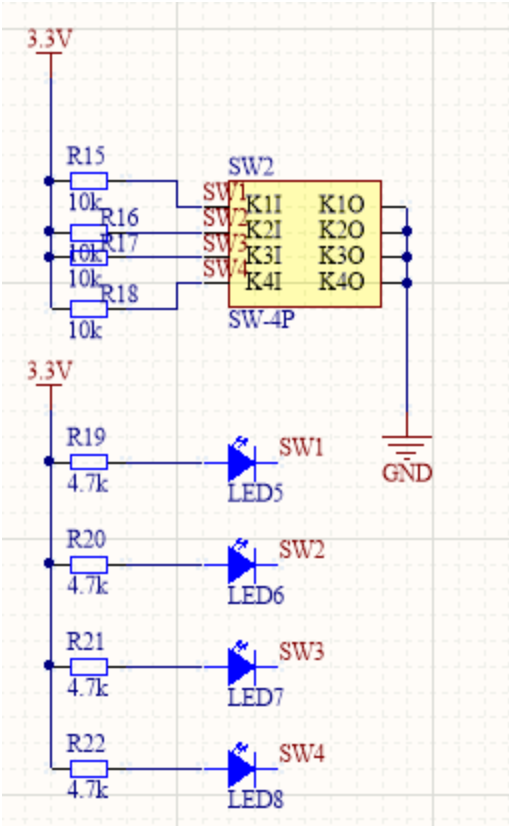
输出最大电流: 7.5A

输出电压计算:

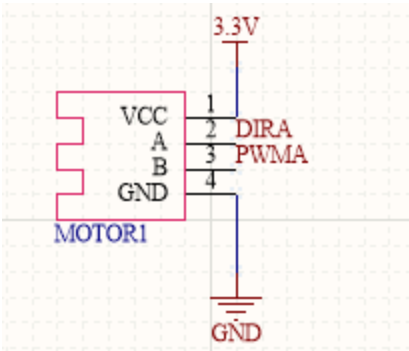
$$R_2 = \frac{R_1 \times 1.22}{V_{out} - 1.22}$$

https://blog.csdn.net/weixin_42121843

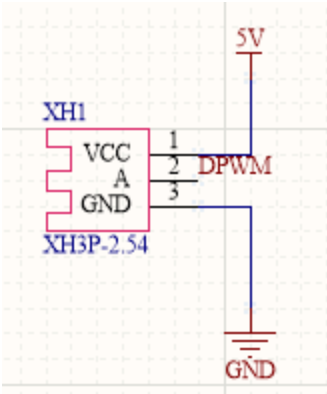
按键模块：（SW2为4并排按键，用于设置车的模式）



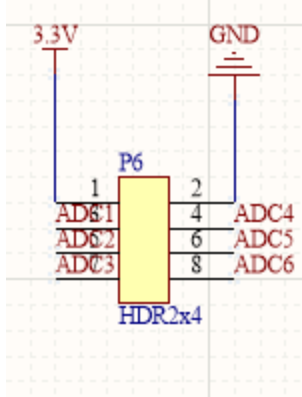
电机输出（为排针插座，接驱动板，DIRA为电机方向控制，PWMA为速度控制）：



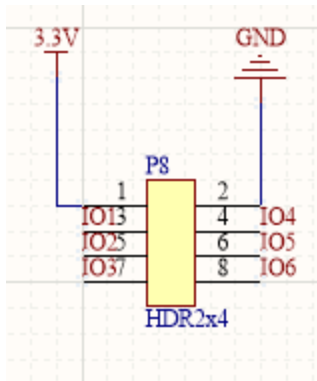
舵机输出（为排针插座，5v供电电压，DPWM为脉冲输出，以控制舵机转角）：



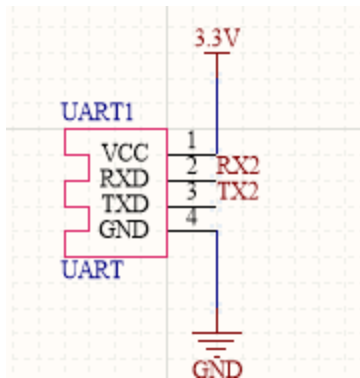
AD输入（为排针插座，接传感器板，6路电感值输入）：



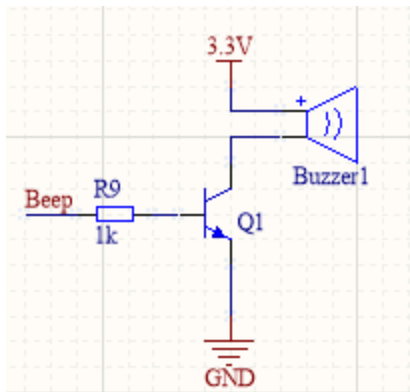
普通IO口：



串口：

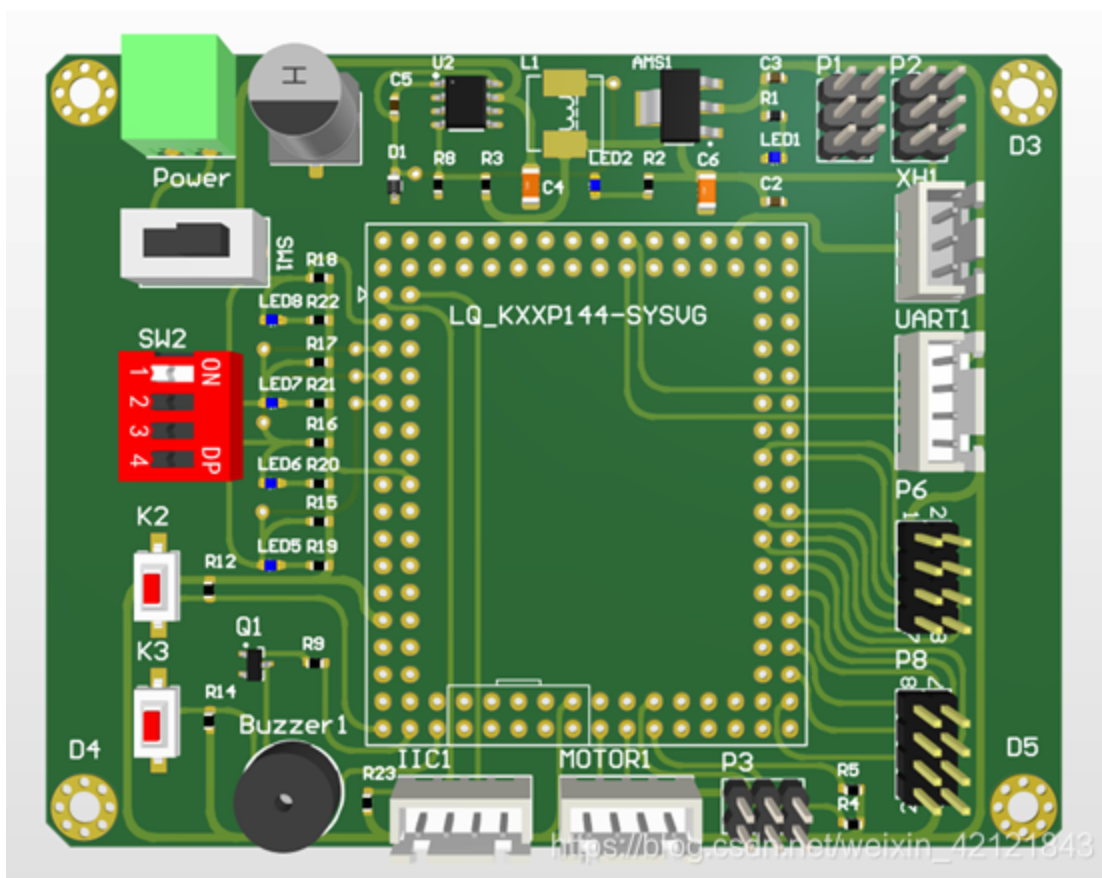
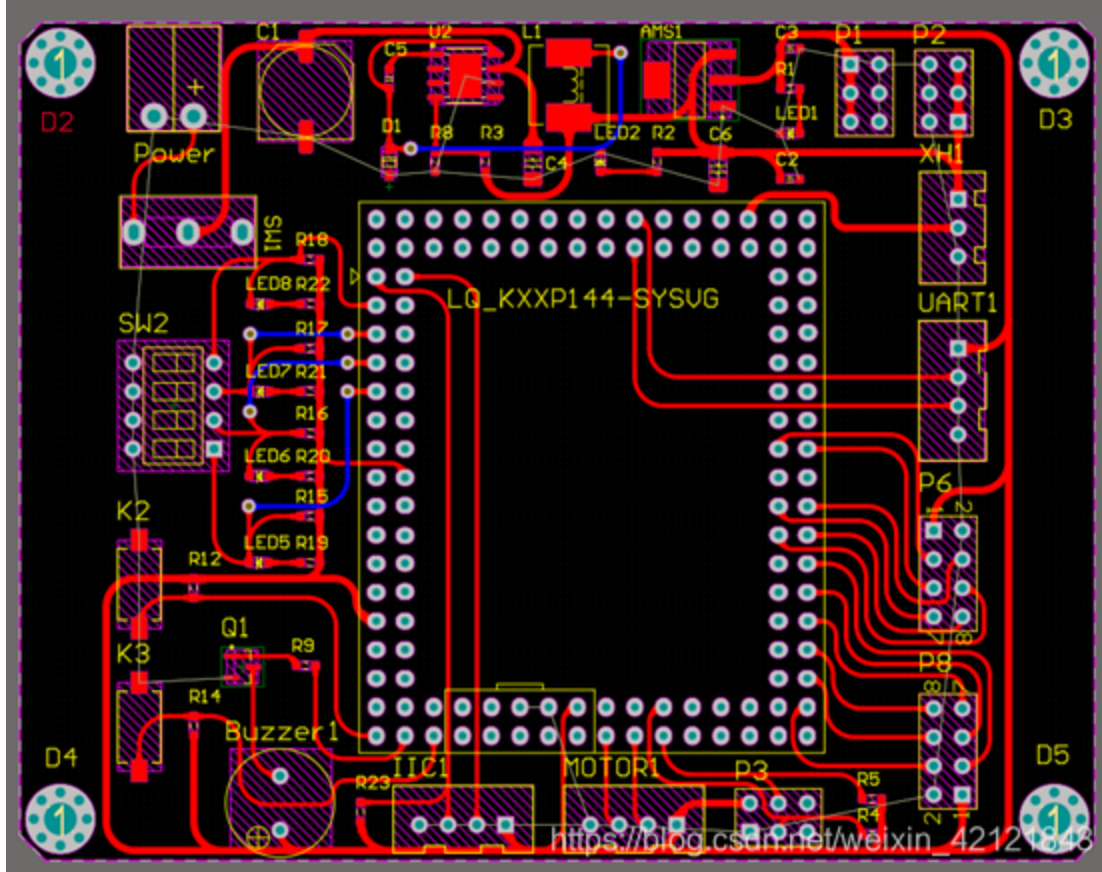


蜂鸣器：



(2) PCB和3D图

PCB中，logo部分和敷铜（GND）已去除，以更清晰显示电路

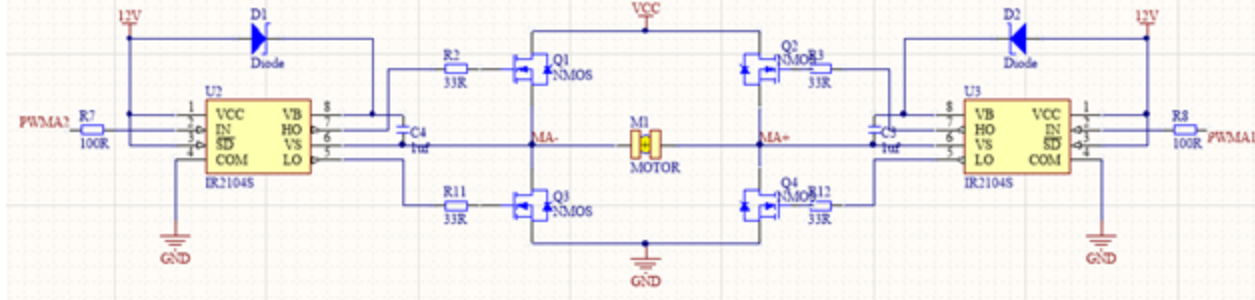


2.电机驱动板：用于驱动电机

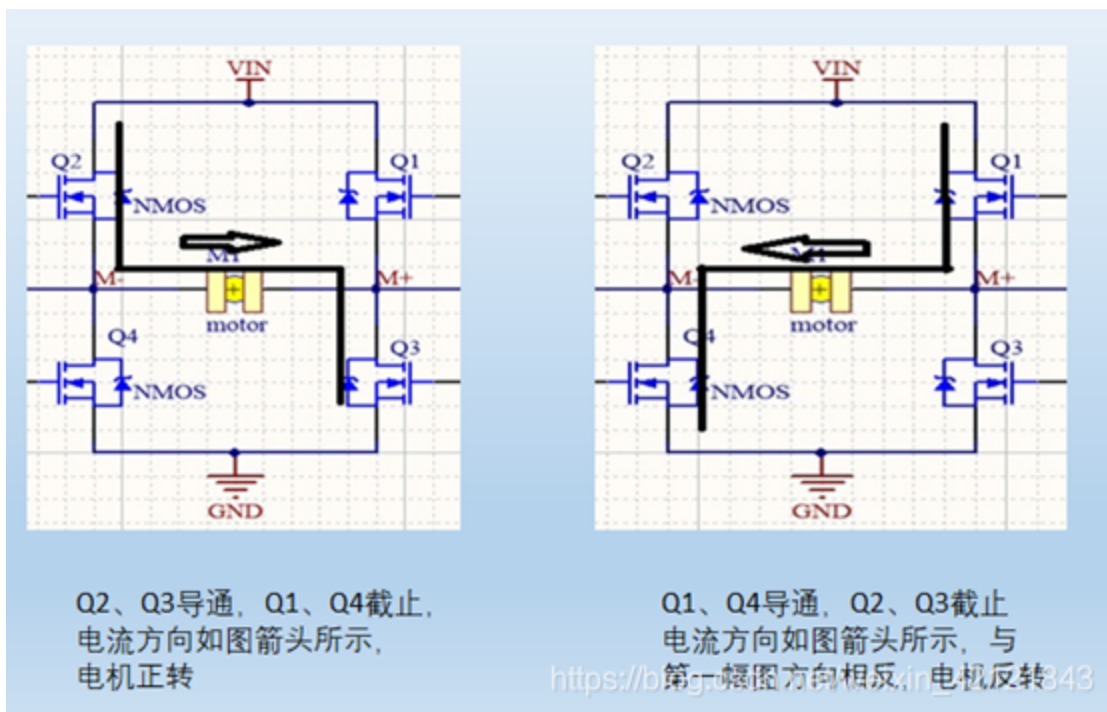
(1) 原理图

H桥电机驱动：

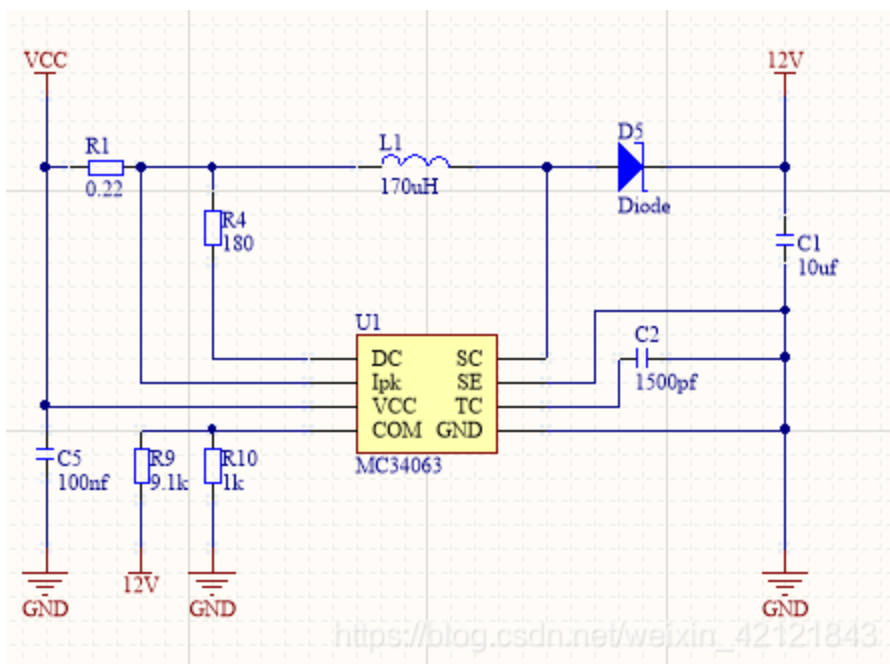
VCC直接接电池（7~8V）



H桥原理：

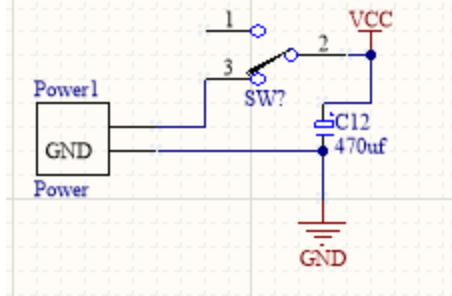


12v升压（用于IR2104s驱动mos管）：

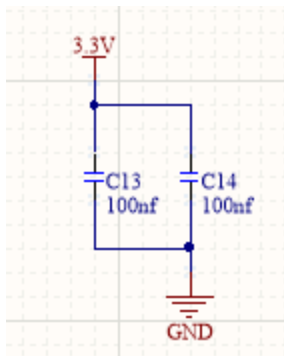
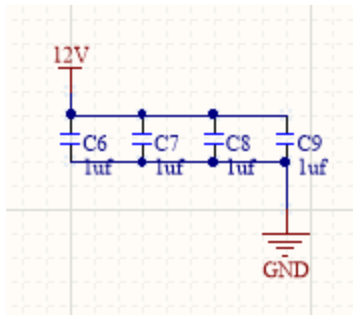


电源部分：

VCC是电机供电

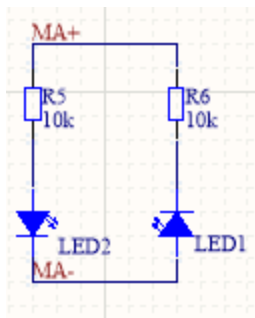


12V（用于IR2104s驱动MOS）和3.3v（逻辑芯片供电）：

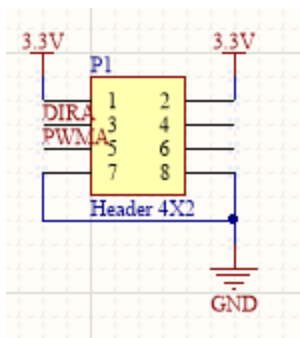


调速及转向：

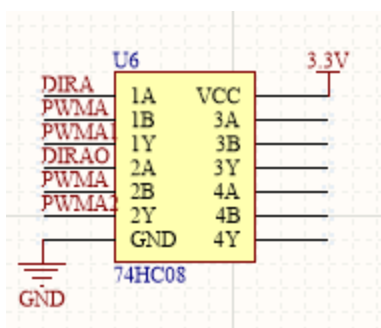
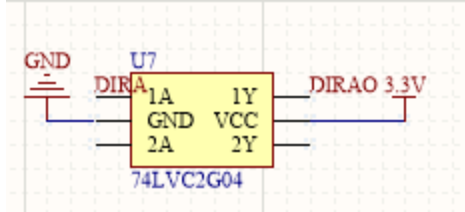
Led显示电机方向及速度大小（速度越大越亮）：



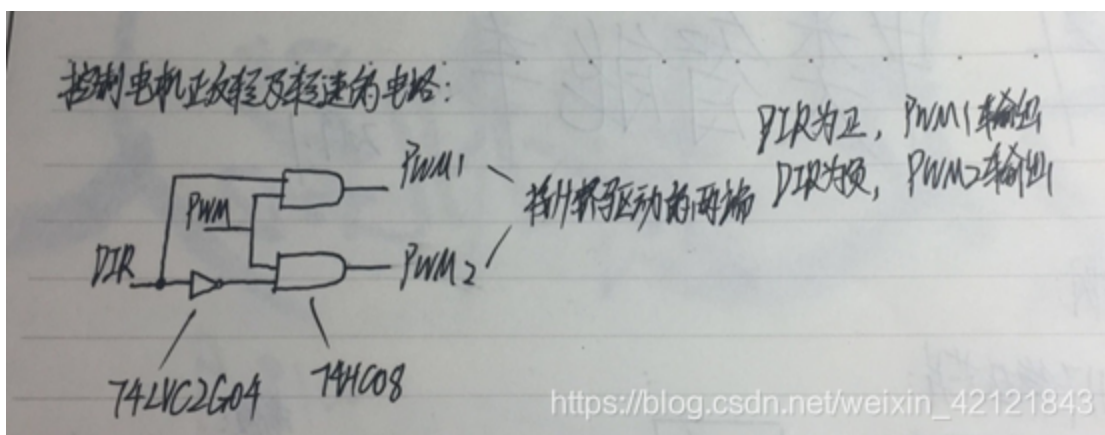
排针插座信号输入，3.3v用于逻辑芯片供电：



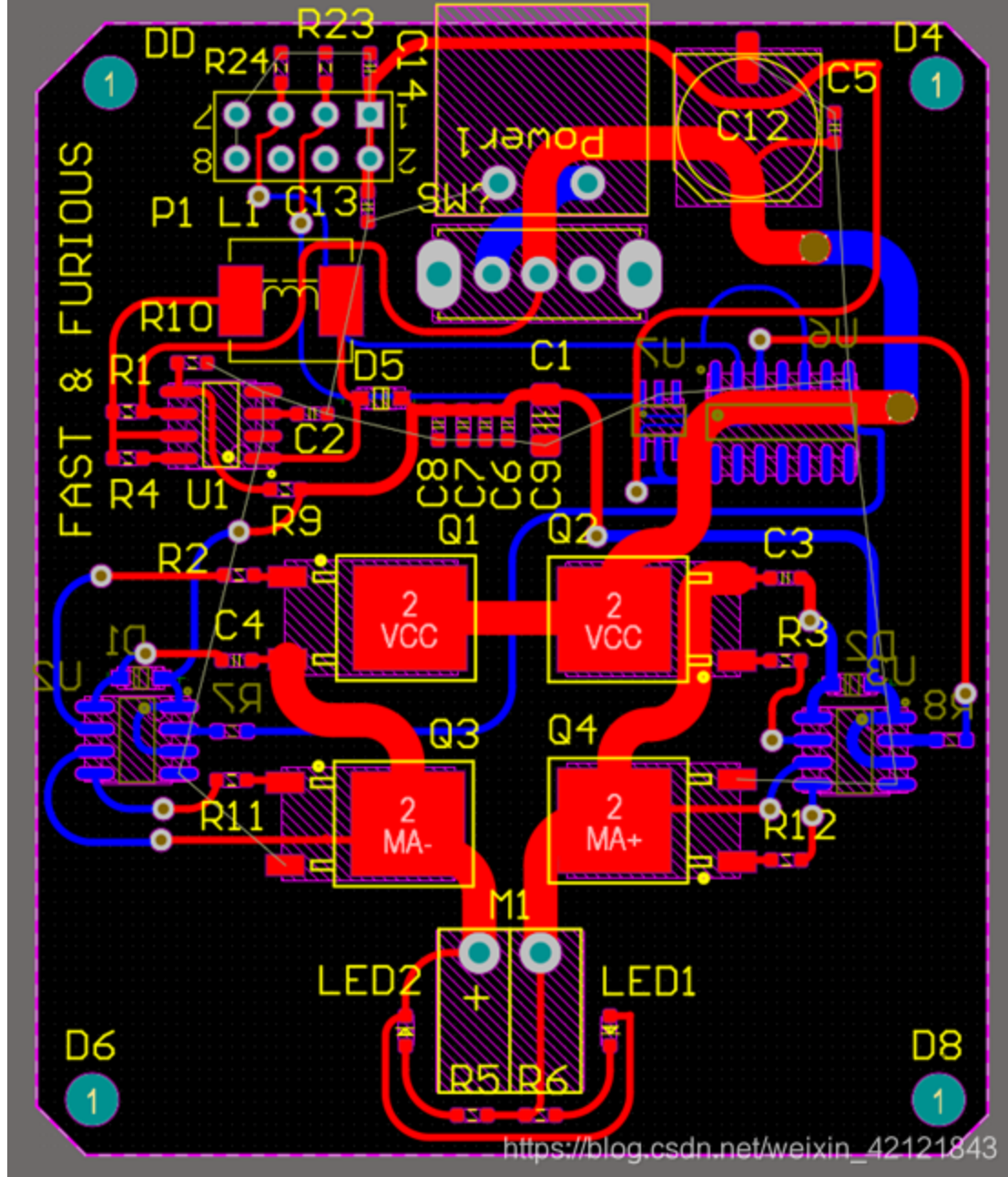
逻辑非和逻辑与芯片：

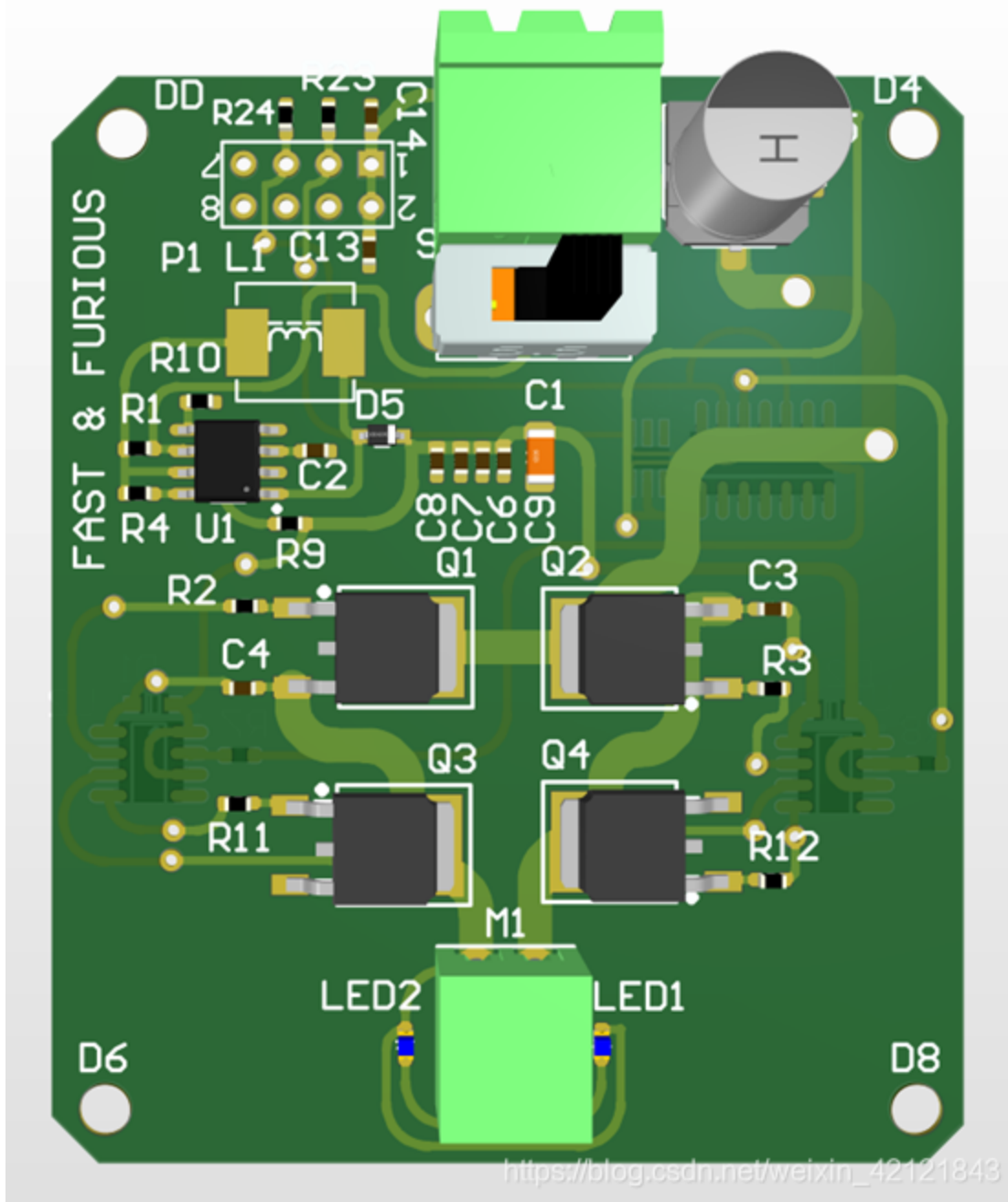


原理笔记:



(2) PCB (GND敷铜已去) 和3D图

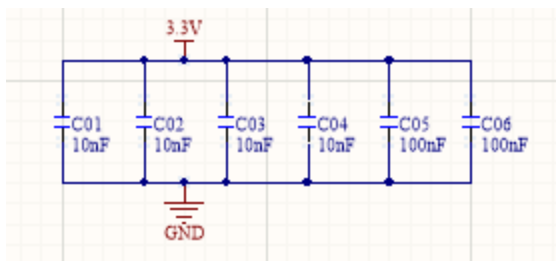




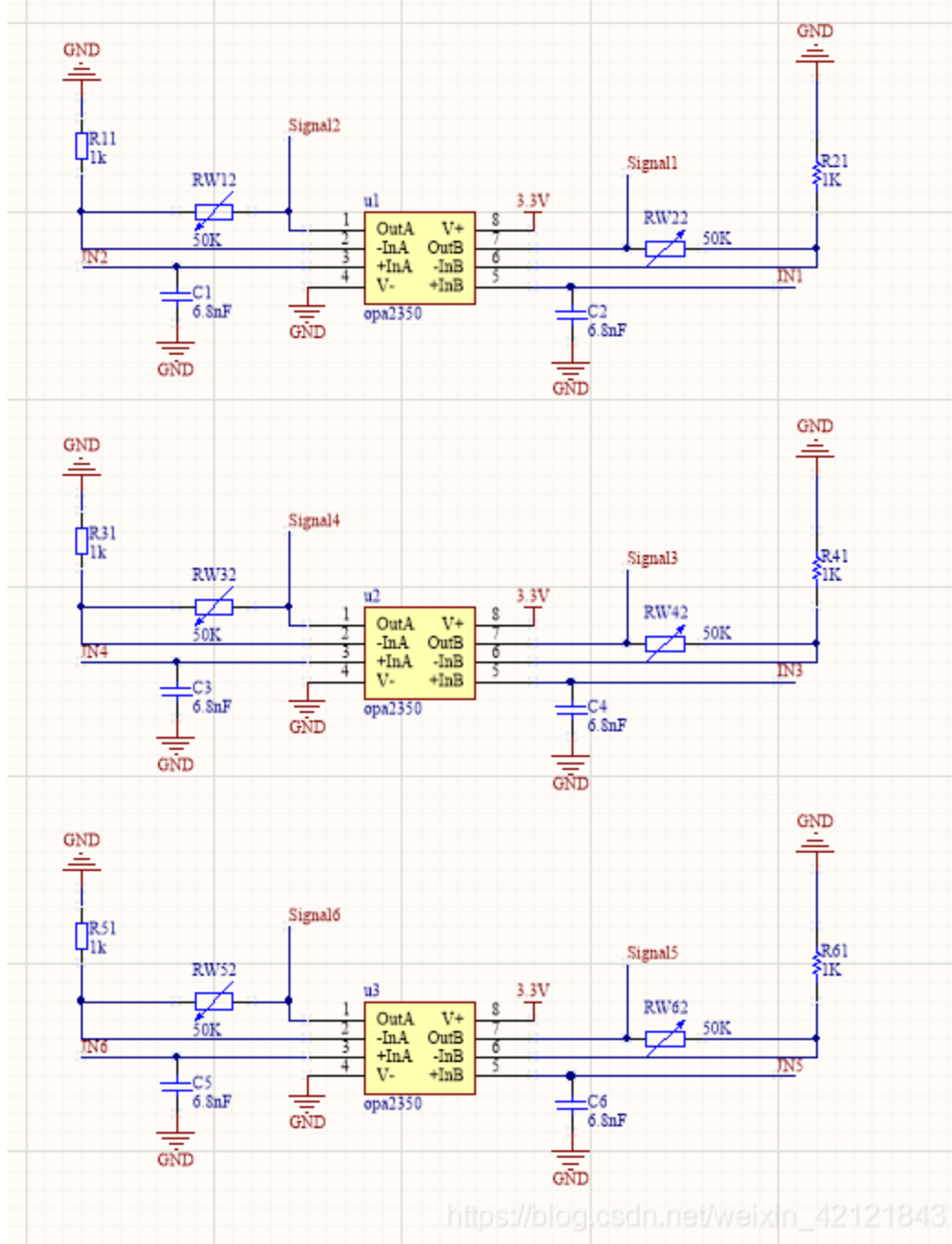
3.传感器板：处理电感数据

(1) 原理图

电源部分：

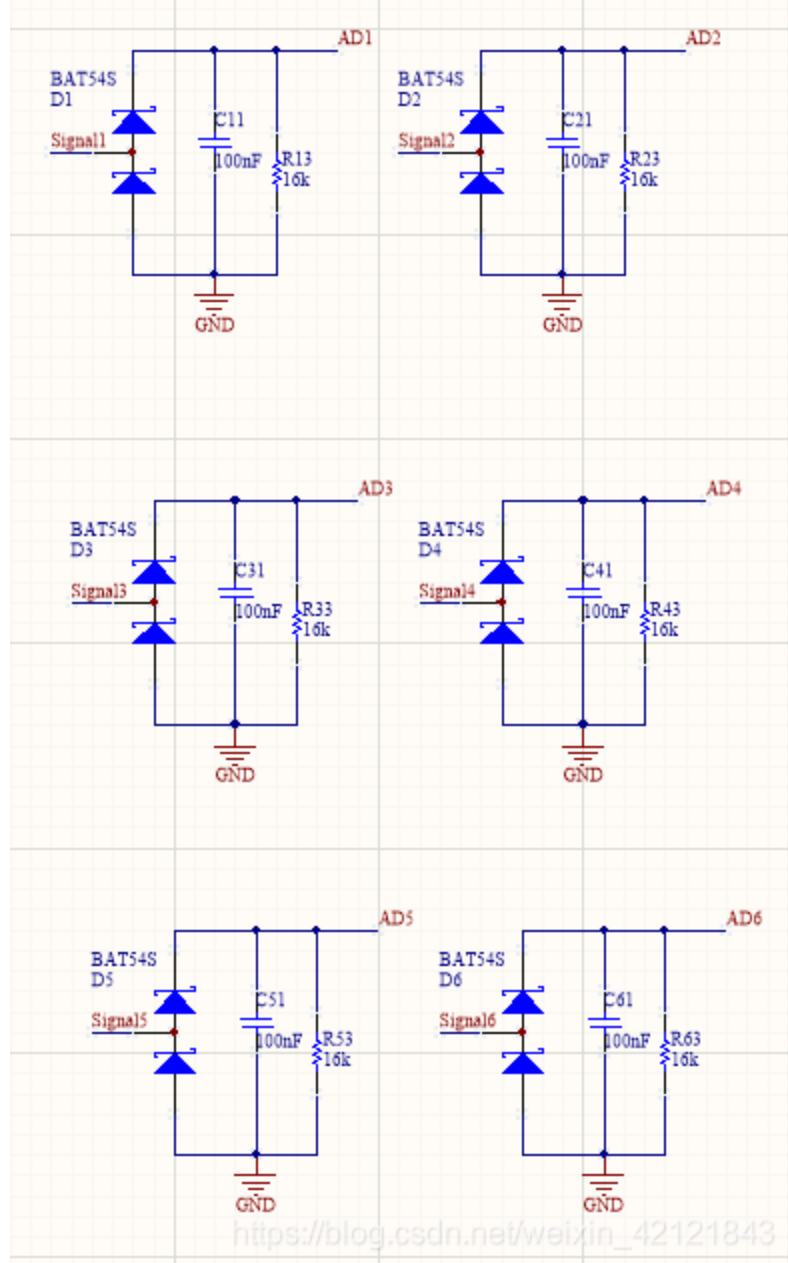


信号放大：(opa2350为运放，根据电阻，放大倍数最大50倍，输入输出为正弦波，共6路)

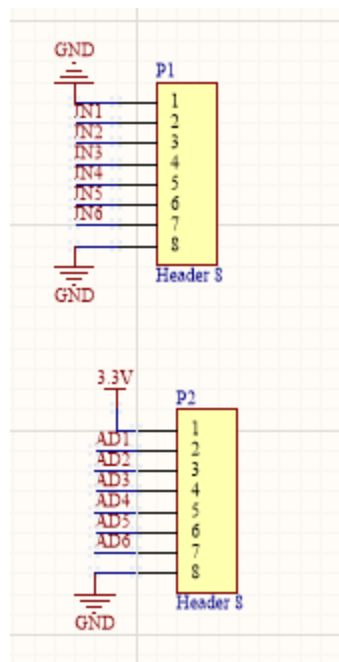


https://blog.csdn.net/weixin_42121843

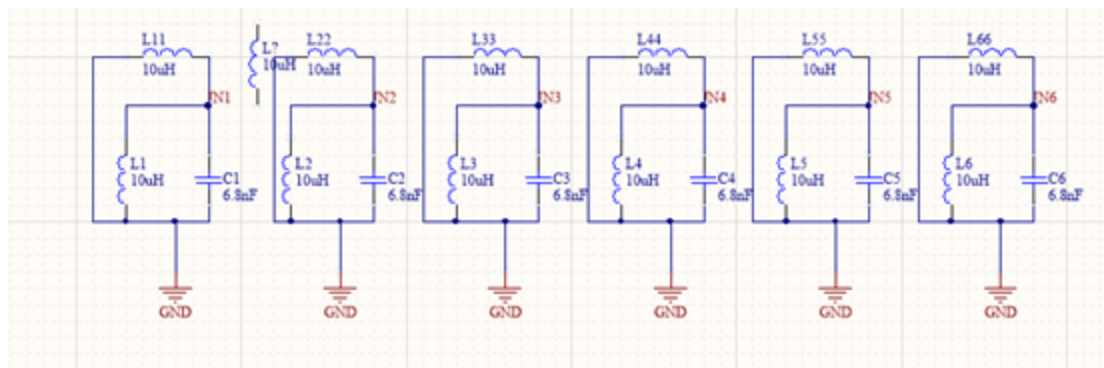
信号整流（经实验，连接AD输出和GND的电容会严重削弱信号，故去除）：



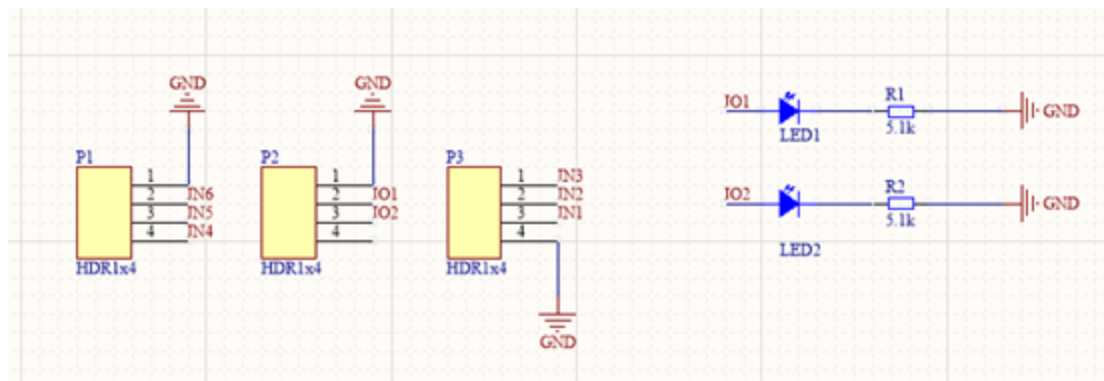
信号输入输出（为插座）：



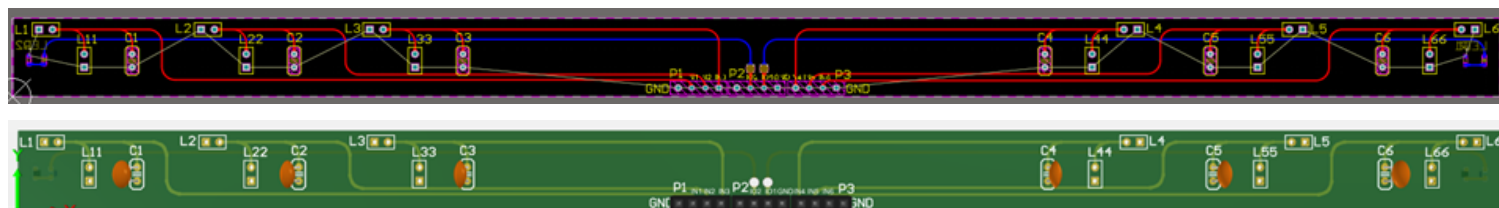
谐振部分：谐振可以很好放大信号，电感电容值根据20K频率谐振计算。



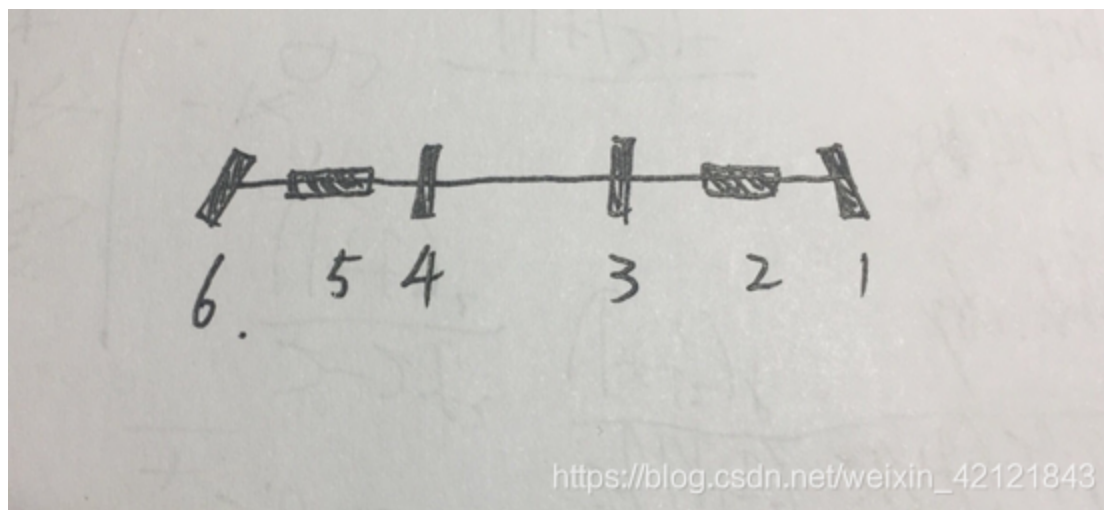
其他部分（LED和排针插座）：



(2) PCB (GND敷铜已去除) 和3D图



电感与电磁线垂直时信号最强，故六路电感位置摆放：



这样放置电感，可以满足在多方位的检测需求。

三、软件部分

智能车主要用到的几样硬件需要软件驱动，包括电机，舵机，编码器，ADC采集和超声波模块；还有一些调车的辅助模块，如OLED，蓝牙，串口，这些部分都采用了相应的例程做了需要的修改。

1. 电机和舵机

(1) .电机

电机部分需要主控芯片输出PWM波控制电机转速，编写了电机初始化及控制函数，包括了电机正反转的设置。

```
void Motor_Init()
{
    GPIO_Init(PORTA,7,1,1);//GPIO初始化
    FTM_PWM_Init(FTM0, FTM_CH2, 0,0);//PWM初始化
    Motor_Control(0,0);
}

void Motor_Control(int16 PWM,int DIR)//PWM为速度参数，DIR为正反转
{
    if(DIR==0)
    {
        GPIO_Ctrl(PORTA,7,0);
    }
    else
    {
        GPIO_Ctrl(PORTA,7,1);
    }
    FTM_PWM_Duty(FTM0,FTM_CH2,PWM);//PWM最大1000
}
```

(2) .舵机

舵机正常工作需要的50hz的PWM波来控制转角，通过控制占空比在5%到15%之间，可以实现舵机180度范围的旋转。在前期的调试过程中确定了舵机状态处于中点的占空比值，在这一值的基础上直接加减实现舵机控制左右转。

测试代码：使轮子到达两个最大转向角

```
FTM_PWM_Duty(FTM2,FTM_CH0,730);
time_delay_ms(2000);
FTM_PWM_Duty(FTM2,FTM_CH0,835);
time_delay_ms(2000);
```

2、传感器（电感，编码器和超声波）

越野组采用电感电容谐振来感应电磁引导线的磁场信号，信号经过谐振放大以及运放放大后通过芯片的ADC模块采集。由于采集的信号可能存在较大波动，采用了均值滤波的方式，选取四次采样平均值作为实际使用的数据。考虑到车前的三组电感相对电磁线的位置有区别，为了使收集到的数据尽量趋于线性变化，对数据做了线性化处理，根据三组电感的位置乘上了参数用作矫正。

```
void ADC_6()
{
    int i;
    volt[0]=ADC_Ave(ADC1,ADC1_SE14,ADC_16bit,4); //均值滤波
    volt[1]=ADC_Ave(ADC1,ADC1_SE15,ADC_16bit,4);
    volt[2]=ADC_Ave(ADC1,ADC1_SE12,ADC_16bit,4);
    volt[3]=ADC_Ave(ADC1,ADC1_SE13,ADC_16bit,4);
    volt[4]=ADC_Ave(ADC1,ADC1_SE10,ADC_16bit,4);
    volt[5]=ADC_Ave(ADC1,ADC1_SE11,ADC_16bit,4);
    sprintf(txt,"0%06d",volt[0]);
    LCD_P6x8Str(5,1,(uint8 *)txt); //LCD屏显示
    sprintf(txt,"1%06d",volt[1]);
    LCD_P6x8Str(5,2,(uint8 *)txt);
    sprintf(txt,"4%06d",volt[4]);
    LCD_P6x8Str(5,3,(uint8 *)txt);
    sprintf(txt,"5%06d",volt[5]);
    LCD_P6x8Str(5,4,(uint8 *)txt);
    sprintf(txt,"2%06d",volt[2]);
    LCD_P6x8Str(54,1,(uint8 *)txt);
    sprintf(txt,"3%06d",volt[3]);
    LCD_P6x8Str(54,2,(uint8 *)txt);
}
```

均值和终值滤波代码：

```
u16 ADC_Ave(ADCn_e adc_n,ADCn_Ch_e adc_ch,ADC_nbit bit,u16 N) //均值滤波
{
    u32 tmp = 0;
    u8 i;
    for(i = 0; i < N; i++)
        tmp += ADC_Mid(adc_n,adc_ch,bit);
    tmp = tmp / N;
    return (u16)tmp;
}

u16 ADC_Mid(ADCn_e adc_n,ADCn_Ch_e adc_ch,ADC_nbit bit) //中值滤波
{
    u16 i,j,k,tmp;
    //1.取3次A/D转换结果
    i = ADC_Once(adc_n,adc_ch,bit);
```

```

j = ADC_Once(adc_n,adc_ch,bit);
k = ADC_Once(adc_n,adc_ch,bit);
//2.取中值
if (i > j)
{
    tmp = i; i = j; j = tmp;
}
if (k > j)
    tmp = j;
else if(k > i)
    tmp = k;
else
    tmp = i;
return tmp;
}

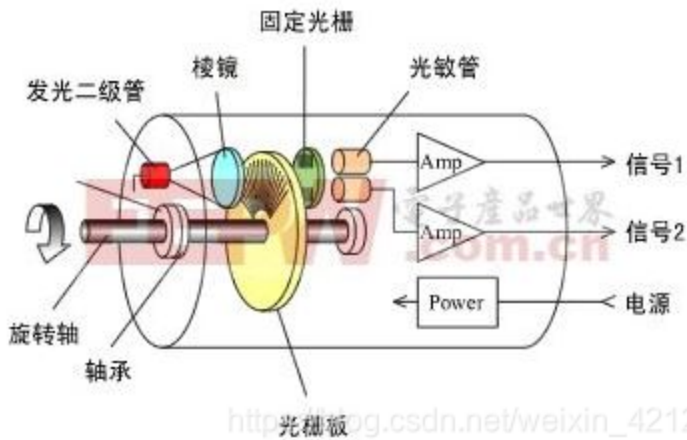
```

(2) 编码器

实物图



AB相增量式编码器原理：



AB相输出：

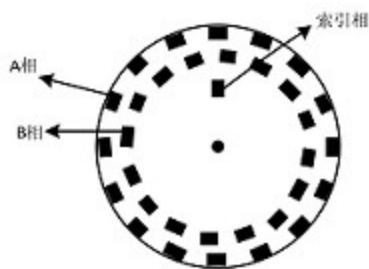


图-1 增量式正交编码盘示意图

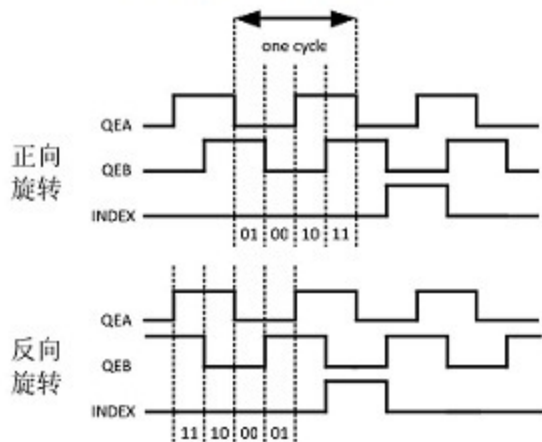
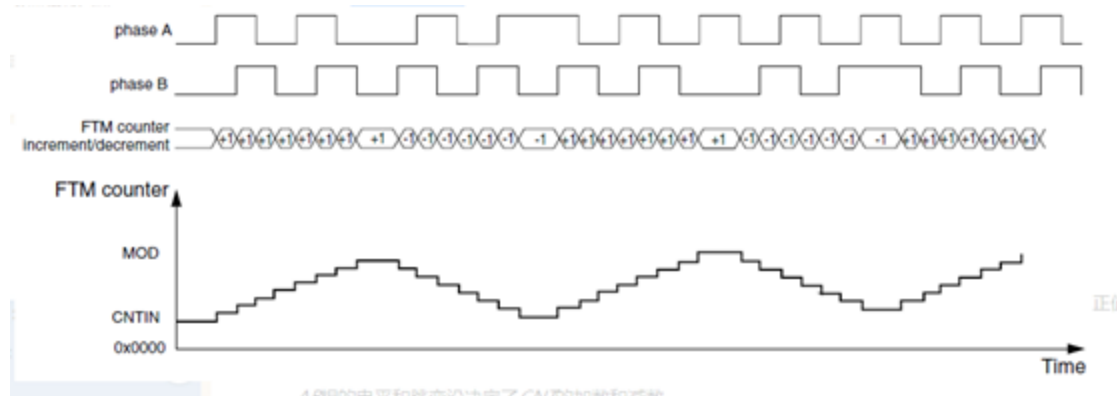


图-2 三相信号正向/反向旋转时序关系

发光二极管发射的光通过光栅到达光敏管，引起电平变化。如果正转，A相输出超前B相90度，如果反转A相滞后B相90度。

AB相正交解码原理：



CNT寄存器：寄存器通过正解解码的得到的脉冲值，最后只要读取CNT的值就可以得到编码器转数（正值为正转，负值为反转）。

CNTIT寄存器：CNT计数的初始值。

AB相的电平和跳变沿决定了CNT的加数和减数。

CNT增计数时：

A上升沿，B逻辑低

B上升沿，A逻辑高

B下降沿，A逻辑低

A下降沿，B逻辑高

CNT减计数是：

A下降沿，B逻辑低

B下降沿，A逻辑高

B上升沿，A逻辑低

A上升沿，B逻辑高

编码器数值的读取我们使用了k60芯片的FTM模块中的正交解码功能来实现，直接调用库中的正交解码函数就可以获得脉冲数，通过比例换算可以获得当前速度。

```
SPEED=FTM_AB_Get(FTM1);    //读取正交解码数值
```

获取FTM正交解码脉冲数的函数：

```
int16 FTM_AB_Get(FTMn_e ftmn)
{
    uint32 val;
    val = FTM_CNT_REG(FTMN[ftmn]); // 计数器，只有低16位可用（写任何值到此寄存器，都会加载 CNTIN 的值
    FTM_CNT_REG(FTMN[ftmn]) = 0;
    return val;
}
```

(3) .超声波

超声波模块的使用主要根据模块的测距原理编写，首先使Trig口输出不少于的10us的高电平信号，随后通过k60低功耗计时器（lptmr）计算Echo端高电平持续时间，即超声波从发送到接收的时间，最后用时间计算出距离。

```
void US_Init()
{
    GPIO_Init(PORTA,27,1,0);
    GPIO_Init(PORTA,29,0,0);
}
```

```
void Distance_Get()
{
    i=0;
    char txt[20]=' ';
    GPIO_Ctrl(PORTA,27,1);
    LPTMR_delay_us(20);
    GPIO_Ctrl(PORTA,27,0);
    while(GPIO_Get(PTA29)==0);
    LPTMR_time_start_us();
    while(GPIO_Get(PTA29)==1);
    i = LPTMR_time_get_us();
    if(i > 65535)
    {
```

```

        d=10000;
    }
    else
    {
        d=(int)(i*0.2);
    }
    sprintf(txt,"dis= %06dmm",d);
    UART_Put_Str(UART4,txt); //串口输出
    LCD_P6x8Str(28,7,(uint8 *)txt); //LCD输出
}

```

3、控制部分：舵机和电机

(1) PID代码

智能车的控制主要采用了经典的PID控制算法，分为电机速度控制和舵机方向控制。

舵机的控制是最重要的部分，是巡线的关键。通过ADC采集的三组电感的电压值来判断当前车身的状态，将当前值与期望值比较得到误差，代入增量式PID，调节出合适的参数，从而能够输出调整量使车能够始终巡线运行。

电机控制要求精度不高，只采用了PI控制，保证车速相对稳定，不至于产生太大波动，主要为了应对粗糙路面以及上下坡的情况。通过编码器传回的速度值判断当前车速，调整至稳定值。主要应用于上坡时，通过PID增大电机电流。

PID及其参数特点：

比例、积分、微分控制各有所长，根据实际需求对几种控制律加以结合，一般采用PI、PD或者PID控制^[1]。

P控制

P（比例）控制实质上是一个可调增益放大器，只改变被调信号的幅值而不改变相位。加大增益可以提高系统开环增益，减小系统稳态误差，提高控制精度，但P控制不能消除稳态误差，且大增益会降低系统相对稳定性，甚至造成系统不稳定。

I控制

I（积分）控制可以消除稳态误差，但控制作用慢，有可能降低系统稳定性。

D控制

D（微分）控制可以超前控制，动作十分迅速，可以改善滞后的情况，但不能消除稳态误差，且当出现脉冲时容易有过激动作，而在恒差时不动作。

https://blog.csdn.net/weixin_42121843

增量式PID介绍：

根据位置式PID控制公式，写出n-1时刻的控制量：

$$u[n-1] = K_p \left\{ e[n-1] + \frac{T}{T_i} \sum_{i=0}^{n-1} e[i] + \frac{T_d}{T} (e[n-1] - e[n-2]) \right\}$$

设

$$\Delta u[n] = u[n] - u[n-1]$$

得到

$$\Delta u[n] = K_p (e[n] - e[n-1]) + \frac{K_p T}{T_i} e[n] + \frac{K_p T_d}{T} (e[n] - 2e[n-1] + e[n-2])$$

令 $K_i = K_p \frac{T}{T_i}$ 为积分系数； $K_d = K_p \frac{T_d}{T}$ 为微分系数，可以将上式简化为

$$\Delta u[n] = K_p (e[n] - e[n-1]) + K_i e[n] + K_d (e[n] - 2e[n-1] + e[n-2])$$

实现代码：

```
typedef struct //把PID定义为结构体形式
{
    float Kp,Ki,Kd;
    float Feedback;
    float Target;
    float error0,error1,error2;
    float Output;
    float Range; //控制输出范围，防止输出过大
}Type_PID;

void PID(Type_PID* PID)
{
    PID->error0 = (PID->Target) - (PID->Feedback);
    PID->Output = (PID->Kp)*(PID->error0)+PID->Kd*(PID->error0-2*PID->error1+PID->error2)+PID->
    //更新偏差
    PID->error2 = PID->error1;
    PID->error1 = PID->error0;
    //对PID死区控制，因输出电流太小不足以驱动电机
    if (abs(PID->Output) < 20)
        PID->Output = 0;
    PID->Output = limit(PID->Output, PID->Range, -PID->Range); //把output限制在+-range之内
}
```

(2) 主函数中PID代码

```
//pid初始设置
pid=(Type_PID *)malloc(sizeof(Type_PID));
memset(pid,0,sizeof(Type_PID));
pid_speed=(Type_PID *)malloc(sizeof(Type_PID));
memset(pid_speed,0,sizeof(Type_PID));
```

```
//电机速度PID
pid_speed->Kp=1;
pid_speed->Ki=4;
pid_speed->Range=700;
//舵机转角PID
pid->Kp=1.5;
pid->Ki=0.7;
pid->Kd=7.5;
pid->Range=142;
```

4.显示部分

(1) 串口与蓝牙

蓝牙模块用的是HC-05，可以与串口共用。

下面给出库中的串口发送字符串函数，不赘述。

```
//例子: uart_putchar (UART4, "123456789");实际发送9个字节
void UART_Put_Str (UARTn_e uratn, char *str) //参数: uratn:模块名如: UART0 , str: 发送的地址
{
    while(*str)
    {
        UART_Put_Char(uratn, *str++);
    }
}
```

(2) LCD屏

列出库中函数，具体实现不赘述。

```
LCD_P6x8Str(unsigned char x,unsigned char y,unsigned char *p)// 写入一组标准ASCII字符串，显示的
void LCD_P14x16Str(unsigned char x,unsigned char y,unsigned char ch[]) //输出汉字字符串
```