OLED 屏显示轻舟机器人状态

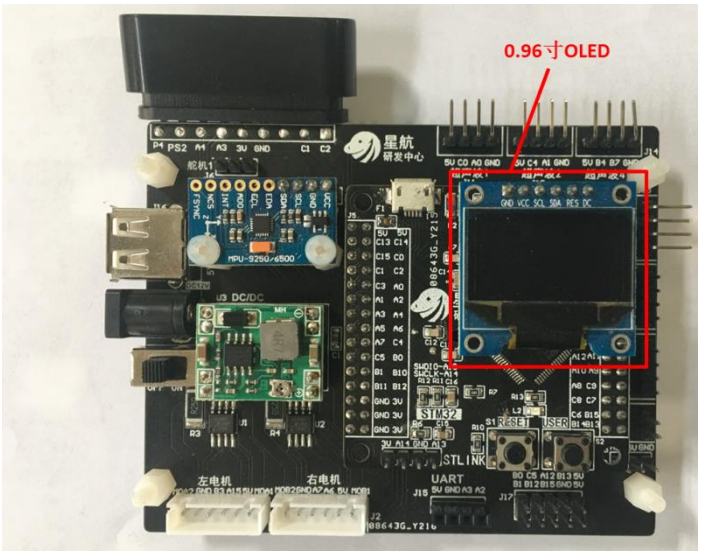
AI 航 团队

轻舟机器人用一块 0.96 寸的 OLED 屏幕显示系统状态，包括电机状态、舵机状态、自动/手动模式、电池电量等信息，本文将介绍如何使用 Stm32 将轻舟机器人状态在 OLED 屏幕上输出显示。

1. OLED 屏幕介绍

OLED，即有机发光二极管（Organic Light-Emitting Diode），又称为有机电激光显示（Organic Electroluminesence Display， OELD）。OLED 由于同时具备自发光，不需背光源、对比度高、厚度薄、视角广、反应速度快、可用于挠曲性面板、使用温度范围广、构造及制程较简单等优异之特性，被认为是下一代的平面显示器新兴应用技术。OLED 显示技术具有自发光的特性，采用非常薄的有机材料涂层和玻璃基板，当有电流通过时，这些有机材料就会发光，而且 OLED 显示屏幕可视角度大，并且能够节省电能，从 2003 年开始这种显示设备在 MP3 播放器上得到了应用。

LCD 都需要背光，而 OLED 不需要，因为它是自发光的。这样同样的显示，OLED 效果要来得好一些。以目前的技术，OLED 的尺寸还难以大型化，但是分辨率确可以做到很高。



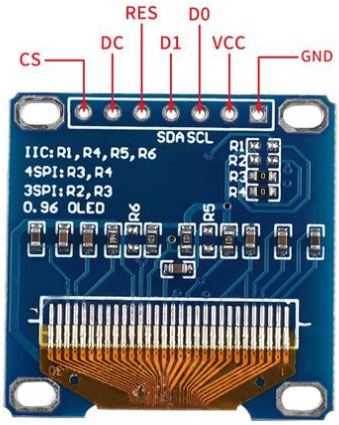
模块一般有单色和双色两种可选，单色为纯蓝色，而双色则为黄蓝双色。单色模块每个像素点只有亮与不亮两种情况，没有颜色区分，显示尺寸为 0.96 寸，分辨率为 128*64；常用的接口为 4 线的串行 SPI 接口方式、IIC 接口方式；不需要高压，接 3.3V 就可以工作。

2.OLED 屏幕参数

轻舟机器人采用的是 SPI 接口方式的 OLED 屏，一般有 7 个引脚与控制器连接，如下图，CS 为片选信号。

引脚定义：

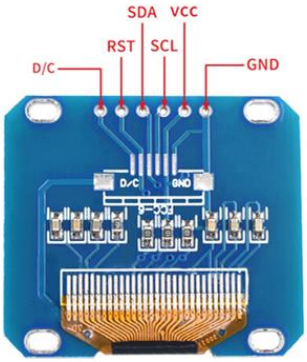
1.CS：片选信号
2.DC：数据或命令切换
3.RES：复位
4.D1：SPI接口为SPI数据线
5.DO：SPI接口为SPI时钟线
6.VCC：电源正3.3V-5V
7.GND：电源地



在某些确定只需要 1 可 OLED 屏的场景下，可以直接将片选信号 CS 拉低，此时模块引出 6 个引脚如下图所示

引脚定义：

1.D/C：数据或命令切换
2.RST：复位
3.SDA：双向数据线
4.SCL：SPI时钟线
5.VCC：电源正3.3V-5V
6.GND：电源负



引脚作用：1. D/C 引脚为 SPI 数据/命令选择引脚， 0 为命令，1 为数据。
2. RST 引脚，复位，在 OLED 上电时需要复位一次。
3. SDA 双向数据线，用于传输 SPI 数据或命令。
4. SCL 引脚，SPI 时钟线
5. VCC 引脚，接电源正 3.3V-5V
6. GND 引脚，接电源负

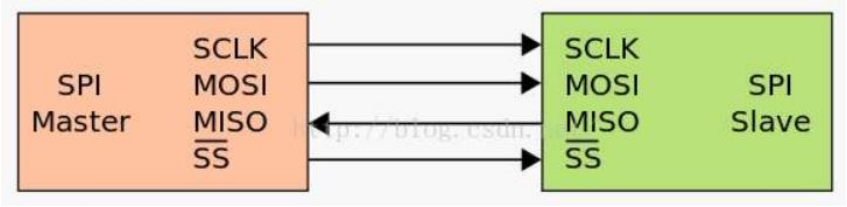
3. SPI 简介

SPI，是英语 Serial Peripheral interface 的缩写，顾名思义就是串行外围设备接口。是 Motorola 首先在其 MC68HCXX 系列处理器上定义的。SPI 接口主要应用在 EEPROM，FLASH，实时时钟，AD 转换器，还有数字信号处理器和数字信号解码器之间。SPI 是一种高速的，全双工，同步的通信总线，并且在芯片的管脚上只占用四根线，节约了芯片的管脚，同时为 PCB 的布局上节省空间，提供方便，正是出于这种简单易用的特性，现在越来越多的芯片集成了这种通信协议。

SPI 优点：支持全双工通信、通信简单、数据传输速率块。

SPI 缺点：没有指定的流控制，没有应答机制确认是否接收到数据，所以跟 IIC 总线协议比较在数据可靠性上有一定的缺陷。

（1）协议简介



SPI 的通信原理很简单，它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以(单向传输时)，我们的轻舟机器人就是使用量 SPI 的单向传输。

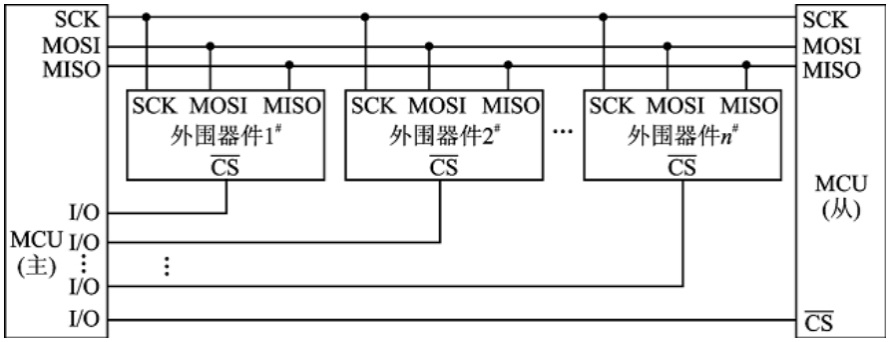
SPI 引脚包括：是 SDI(数据输入)、SDO(数据输出)、SCLK(时钟)、CS(片选)。

(1)SDO/MOSI－主设备数据输出，从设备数据输入;

(2)SDI/MISO－主设备数据输入，从设备数据输出;

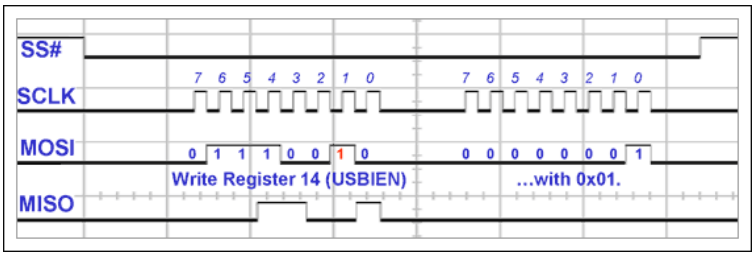
(3)SCLK－时钟信号，由主设备产生;

(4)CS/SS－从设备使能信号，由主设备控制。当有多个从设备的时候，因为每个从设备上都有一个片选引脚接入到主设备机中，当我们的主设备和某个从设备通信时将需要将从设备对应的片选引脚电平拉低或者是拉高。



要注意的是，SCK 信号线只由主设备控制，从设备不能控制信号线。同样，在一个基于 SPI 的设备中，至少有一个主控设备。这样的传输方式有一个优点，与普通的串行通讯不同，普通的串行通讯一次连续传送至少 8 位数据，而 SPI 允许数据一位一位的传送，甚至允许暂停，因为 SCK 时钟线由主控设备控制，当没有时钟跳变时，从设备不采集或传送数据，也就是说，主设备通过对 SCK 时钟线的控制可以完成对通讯的控制。

在点对点的通信中，SPI 接口不需要进行寻址操作，且为全双工通信，显得简单高效。在多个从设备的系统中，每个从设备需要独立的使能信号，硬件上比 I2C 系统要稍微复杂一些。SPI 的简单收发时序图如下图：



在轻舟机器人的 OLED 模块中，前边已经讲过，模块的 CS 片选引脚已经默认接地了，而我们只需要主机给从机发送数据，则没有从机给主机发送数据的引脚，所以 6 脚 OLED 模块参与 SPI 时序控制的只有 SCL、SDA 两个引脚，在配合 DC 引脚，完成 OLED 模块的控制。

4．轻舟机器人 OLED 控制程序解读

第一部分：OLED 屏幕驱动头文件（oled.h）

```
#ifndef __OLED_H
#define __OLED_H
#include "sys.h"

//-----------------OLED 端口定义----------------
#define OLED_RST_Clr() PCout(15)=0          //RST   复位引脚，OLED 在上电后需要一次复位
#define OLED_RST_Set() PCout(15)=1          //RST

#define OLED_RS_Clr() PCout(0)=0            //D/C   SPI 数据/命令选择引脚   0 命令
#define OLED_RS_Set() PCout(0)=1            // D/C   SPI 数据/命令选择引脚  1 数据

#define OLED_SCLK_Clr()   PCout(13)=0       //SCL 时钟线
#define OLED_SCLK_Set()   PCout(13)=1       //SCL

#define OLED_SDIN_Clr()   PCout(14)=0       //SDA 数据线
#define OLED_SDIN_Set()   PCout(14)=1       //SDA

#define OLED_CMD   0                        //写命令
#define OLED_DATA 1                         //写数据

//OLED 控制用函数
void OLED_WR_Byte(u8 dat,u8 cmd);
void OLED_Display_On(void);
void OLED_Display_Off(void);
void OLED_Refresh_Gram(void);
void OLED_Init(void);
void OLED_Clear(void);
void OLED_DrawPoint(u8 x,u8 y,u8 t);
void OLED_ShowChar(u8 x,u8 y,u8 chr,u8 size,u8 mode);
void OLED_ShowNumber(u8 x,u8 y,u32 num,u8 len,u8 size);
void OLED_ShowString(u8 x,u8 y,const u8 *p);
#endif
```

第二部分：OLED 屏幕驱动文件（oled.c）

```
#include "oled.h"
#include "stdlib.h"
#include "oledfont.h"
#include "delay.h"

u8 OLED_GRAM[128][8];

//向 OLED 写入一个字节。dat:要写入的数据/命令。cmd:数据/命令标志  0,表示命令;1,表示数据;
void OLED_WR_Byte(u8 dat,u8 cmd)
{
    u8 i;
    if(cmd)
      OLED_RS_Set();
    else
      OLED_RS_Clr();
    for(i=0;i<8;i++)
    {
        OLED_SCLK_Clr();        //时钟线拉低
        if(dat&0x80)            //判断当前数据位电平
          OLED_SDIN_Set();
        else
          OLED_SDIN_Clr();
        OLED_SCLK_Set();        //时钟线拉高
        dat<<=1;                //将数据左移一位
    }
    OLED_RS_Set();              //将 D/C 数据/命令引脚  置 1
}
```

```
/*函数功能：更新显存到 OLED 屏幕上*/
void OLED_Refresh_Gram(void)
{
	u8 i,n;		//定义页地址和列地址
	for(i=0;i<8;i++)
	{
		OLED_WR_Byte (0xb0+i,OLED_CMD);	//设置页地址（0~7）
		OLED_WR_Byte (0x00,OLED_CMD);	//设置显示位置—列低地址
		OLED_WR_Byte (0x10,OLED_CMD);	//设置显示位置—列高地址
		for(n=0;n<128;n++)OLED_WR_Byte(OLED_GRAM[n][i],OLED_DATA);
	}
}

//开启 OLED 显示
void OLED_Display_On(void)
{
	OLED_WR_Byte(0X8D,OLED_CMD);	//SET DCDC 命令
	OLED_WR_Byte(0X14,OLED_CMD);	//DCDC ON
	OLED_WR_Byte(0XAF,OLED_CMD);	//DISPLAY ON
}

//关闭 OLED 显示
void OLED_Display_Off(void)
{
	OLED_WR_Byte(0X8D,OLED_CMD);	//SET DCDC 命令
	OLED_WR_Byte(0X10,OLED_CMD);	//DCDC OFF
	OLED_WR_Byte(0XAE,OLED_CMD);	//DISPLAY OFF
}

//清屏函数,清完屏,整个屏幕是黑色的，和没点亮一样！清屏就是向 OLED 屏里写 0
void OLED_Clear(void)
{
	u8 i,n;
	for(i=0;i<8;i++)for(n=0;n<128;n++)OLED_GRAM[n][i]=0X00;	//将 OLED_GRAM 数组清零
	OLED_Refresh_Gram();	//更新显示
}

函数功能：在显存数组上画一个点
函数参数：x，y 为点的横纵坐标	t 为这个点的亮灭（1 亮 0 灭）
参数范围：x 0~128	y 0~64
每一个数据是 低位在前，高位在后*/
void OLED_DrawPoint(u8 x,u8 y,u8 t)
{
	u8 pos,bx,temp=0;
	if(x>127||y>63)return;	//超出范围了.
	pos=7-y/8;		//显存的页地址
	bx=y%8;			//显存的一个字节数据中 t 所在的位置
	temp=1<<(7-bx);
	if(t)OLED_GRAM[x][pos]|=temp;
	else OLED_GRAM[x][pos]&=~temp;
}

//在指定位置显示一个字符,包括部分字符，x:0~127，y:0~63，mode:0,反白显示;1,正常显示
//size:选择字体 16/12
void OLED_ShowChar(u8 x,u8 y,u8 chr,u8 size,u8 mode)
{
	u8 temp,t,t1;
	u8 y0=y;
	chr=chr-' ';				//得到偏移后的值
	for(t=0;t<size;t++)
	{
		if(size==12)temp=oled_asc2_1206[chr][t];	//调用 1206 字体
		else temp=oled_asc2_1608[chr][t];		//调用 1608 字体
```

```
            for(t1=0;t1<8;t1++)
              {
                    if(temp&0x80)OLED_DrawPoint(x,y,mode);
                    else OLED_DrawPoint(x,y,!mode);
                    temp<<=1;
                    y++;
                    if((y-y0)==size)
                    {
                          y=y0;
                          x++;
                          break;
                    }
              }
      }
}

//m^n 函数
u32 oled_pow(u8 m,u8 n)
{
       u32 result=1;
       while(n--)result*=m;
       return result;
}

//显示 2 个数字，x,y :起点坐标，len :数字的位数，size:字体大小，mode:模式 0,填充模式;1,叠加模式
//num:数值(0~4294967295);
void OLED_ShowNumber(u8 x,u8 y,u32 num,u8 len,u8 size)
{
       u8 t,temp;
       u8 enshow=0;
       for(t=0;t<len;t++)
       {
             temp=(num/oled_pow(10,len-t-1))%10;
             if(enshow==0&&t<(len-1))
             {
                    if(temp==0)
                    {
                          OLED_ShowChar(x+(size/2)*t,y,' ',size,1);
                          continue;
                    }else enshow=1;
             }
             OLED_ShowChar(x+(size/2)*t,y,temp+'0',size,1);
       }
}

//显示字符串，x,y:起点坐标，*p:字符串起始地址，用 16 字体
void OLED_ShowString(u8 x,u8 y,const u8 *p)
{
#define MAX_CHAR_POSX 122
#define MAX_CHAR_POSY 58
      while(*p!='\0')
      {
            if(x>MAX_CHAR_POSX){x=0;y+=16;}
            if(y>MAX_CHAR_POSY){y=x=0;OLED_Clear();}
            OLED_ShowChar(x,y,*p,12,1);
            x+=8;
            p++;
      }
}

//初始化 OLED
void OLED_Init(void)
{
```

```
    GPIO_InitTypeDef GPIO_InitStructure;
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);     //使能 PC 端口时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);        //使能 AFIO 时钟
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;//端口配置
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;           //推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;            //2M
     GPIO_Init(GPIOC, &GPIO_InitStructure);                     //根据设定参数初始化 GPIO

    PWR_BackupAccessCmd(ENABLE);    //允许修改 RTC 和后备寄存器
    RCC_LSEConfig(RCC_LSE_OFF);     //关闭外部低部时钟信号功能后，PC13 /14 /15 才可以当普通 IO
    BKP_TamperPinCmd(DISABLE);      //关闭入侵检测功能，也就是 PC13，也可以当普通 IO 使用
    PWR_BackupAccessCmd(DISABLE);   //禁止修改后备寄存器

    OLED_RST_Clr();                 //复位拉低
    delay_ms(100);
    OLED_RST_Set();                 //复位拉高

    OLED_WR_Byte(0xAE,OLED_CMD);    //关闭显示
    OLED_WR_Byte(0xD5,OLED_CMD);    //设置时钟分频因子,震荡频率
    OLED_WR_Byte(80,OLED_CMD);      //[3:0],分频因子;[7:4],震荡频率
    OLED_WR_Byte(0xA8,OLED_CMD);    //设置驱动路数
    OLED_WR_Byte(0X3F,OLED_CMD);    //默认 0X3F(1/64)
    OLED_WR_Byte(0xD3,OLED_CMD);    //设置显示偏移
    OLED_WR_Byte(0X00,OLED_CMD);    //默认为 0

    OLED_WR_Byte(0x40,OLED_CMD);    //设置显示开始行 [5:0],行数.

    OLED_WR_Byte(0x8D,OLED_CMD);    //电荷泵设置
    OLED_WR_Byte(0x14,OLED_CMD);    //bit2，开启/关闭
    OLED_WR_Byte(0x20,OLED_CMD);    //设置内存地址模式
    OLED_WR_Byte(0x02,OLED_CMD);    //[1:0],00，列地址模式;01，行地址模式;10,页地址模式;默认 10;
    OLED_WR_Byte(0xA1,OLED_CMD);    //段重定义设置,bit0:0,0->0;1,0->127;
    OLED_WR_Byte(0xC0,OLED_CMD);    // 设 置 COM 扫 描 方 向 ;bit3:0, 普 通 模 式 ;1, 重 定 义 模 式
COM[N-1]->COM0;N:驱动路数
    OLED_WR_Byte(0xDA,OLED_CMD);    //设置 COM 硬件引脚配置
    OLED_WR_Byte(0x12,OLED_CMD);    //[5:4]配置

    OLED_WR_Byte(0x81,OLED_CMD);    //对比度设置
    OLED_WR_Byte(0xEF,OLED_CMD);    //1~255;默认 0X7F (亮度设置,越大越亮)
    OLED_WR_Byte(0xD9,OLED_CMD);    //设置预充电周期
    OLED_WR_Byte(0xf1,OLED_CMD);    //[3:0],PHASE 1;[7:4],PHASE 2;
    OLED_WR_Byte(0xDB,OLED_CMD);    //设置 VCOMH 电压倍率
    OLED_WR_Byte(0x30,OLED_CMD);    //[6:4] 000,0.65*vcc;001,0.77*vcc;011,0.83*vcc;

    OLED_WR_Byte(0xA4,OLED_CMD);    //全局显示开启;bit0:1,开启;0,关闭;(白屏/黑屏)
    OLED_WR_Byte(0xA6,OLED_CMD);    //设置显示方式;bit0:1,反相显示;0,正常显示
    OLED_WR_Byte(0xAF,OLED_CMD);    //开启显示
    OLED_Clear();
}
```

第三部分 在 OLED 屏上显示轻舟机器人状态

```
int accz;
void oled_show(void)
{
    //su 第 1 行
    OLED_ShowString(00,0,"A");
    OLED_ShowNumber(15,0, Distance_A,3,12);   //PS2 的数据 PS2_LX Distance_A
    OLED_ShowString(40,0,"B");
    OLED_ShowNumber(55,0, Distance_B,3,12);   //PS2 的数据 PS2_LY Distance_B
    OLED_ShowString(80,0,"C");
    OLED_ShowNumber(95,0, Distance_C,3,12);  //PS2_RX   Distance_C

    //su 第 2 行
```

```
    OLED_ShowString(0,10,"KEY");
    OLED_ShowNumber(25,10,PS2_KEY,2,12);

    //=============第 3 行显示左电机的状态=====================//
    if( Target_Left<0)        OLED_ShowString(00,20,"-"),
                              OLED_ShowNumber(15,20,-Target_Left,5,12);    //su 电机目标值
     else                     OLED_ShowString(0,20,"+"),
                              OLED_ShowNumber(15,20, Target_Left,5,12);
     if( Encoder_Left<0)      OLED_ShowString(80,20,"-"),
                              OLED_ShowNumber(95,20,-Encoder_Left,4,12); //su 编码器脉冲数
     else                     OLED_ShowString(80,20,"+"),
                              OLED_ShowNumber(95,20, Encoder_Left,4,12);
    //=============第 4 行显示右电机的状态=====================//
    if( Target_Right<0)      OLED_ShowString(00,30,"-"),
                             OLED_ShowNumber(15,30,-Target_Right,5,12);
    else                     OLED_ShowString(0,30,"+"),
                             OLED_ShowNumber(15,30, Target_Right,5,12);
     if( Encoder_Right<0)    OLED_ShowString(80,30,"-"),
                             OLED_ShowNumber(95,30,-Encoder_Right,4,12);
    else                     OLED_ShowString(80,30,"+"),
                             OLED_ShowNumber(95,30, Encoder_Right,4,12);
     //=============第 5 行显示舵机的状态=====================//
    OLED_ShowString(00,40,"Servo:"),        //舵机状态
    OLED_ShowNumber(60,40, Servo,4,12);
     //=============第 6 行显示电压模式等=====================//
     OLED_ShowString(48,50,".");
    OLED_ShowString(70,50,"V");
    OLED_ShowNumber(35,50,Voltage/100,2,12);//电压
    OLED_ShowNumber(58,50,Voltage%100,2,12);
     if(Voltage%100<10)     OLED_ShowNumber(52,50,0,2,12);

     //电机使能/使能显示
    if(Flag_Stop==0)            OLED_ShowString(103,50,"O-N");//根据 Flag_Stop 标志位显示电机的状态
    if(Flag_Stop==1)            OLED_ShowString(103,50,"OFF");
    if(operationMode==0)    OLED_ShowString(0,50,"AUT");//不同的模式
    else if(operationMode==1)       OLED_ShowString(0,50,"PS2");
    OLED_Refresh_Gram();        //刷新
}
```