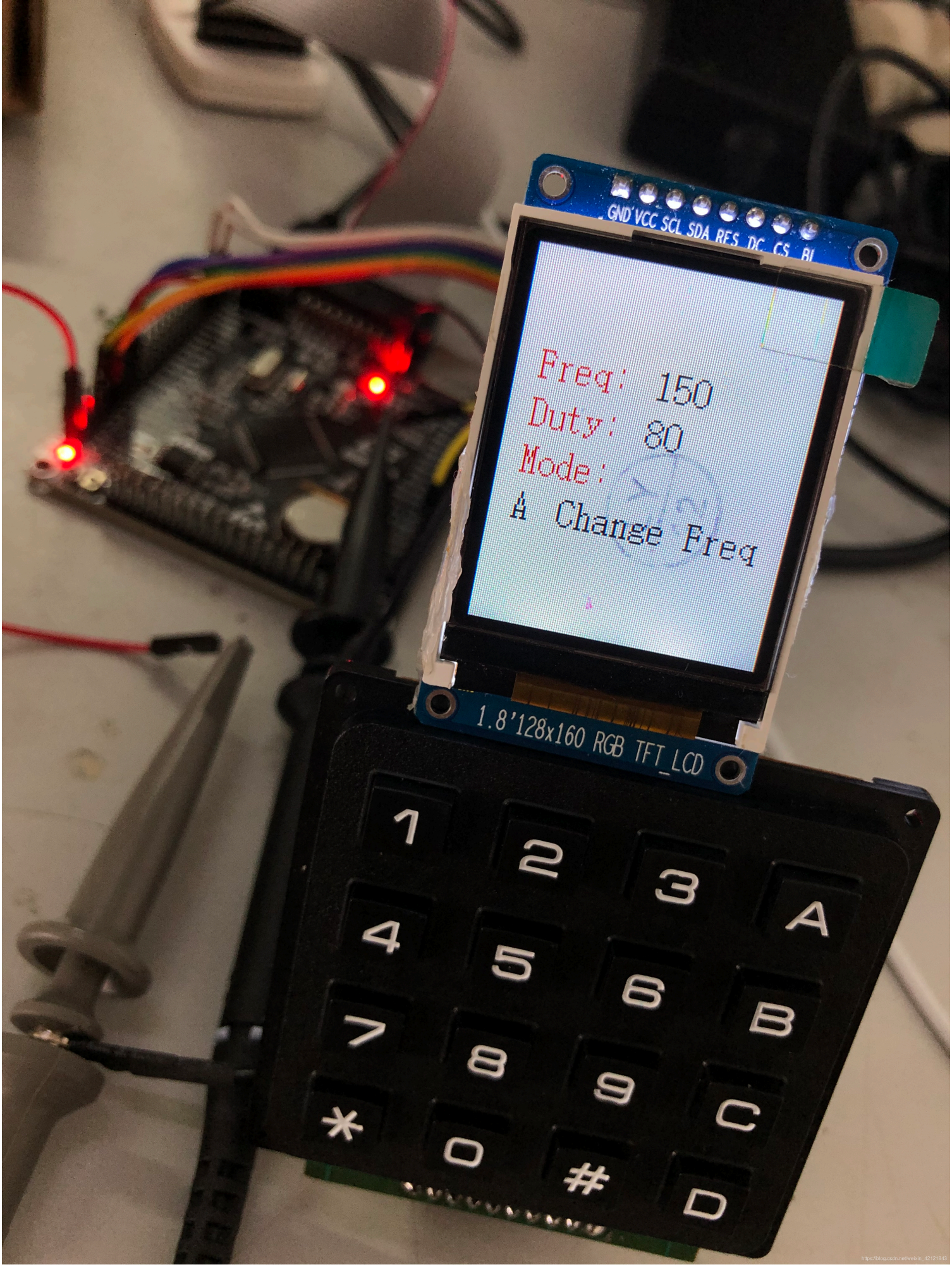
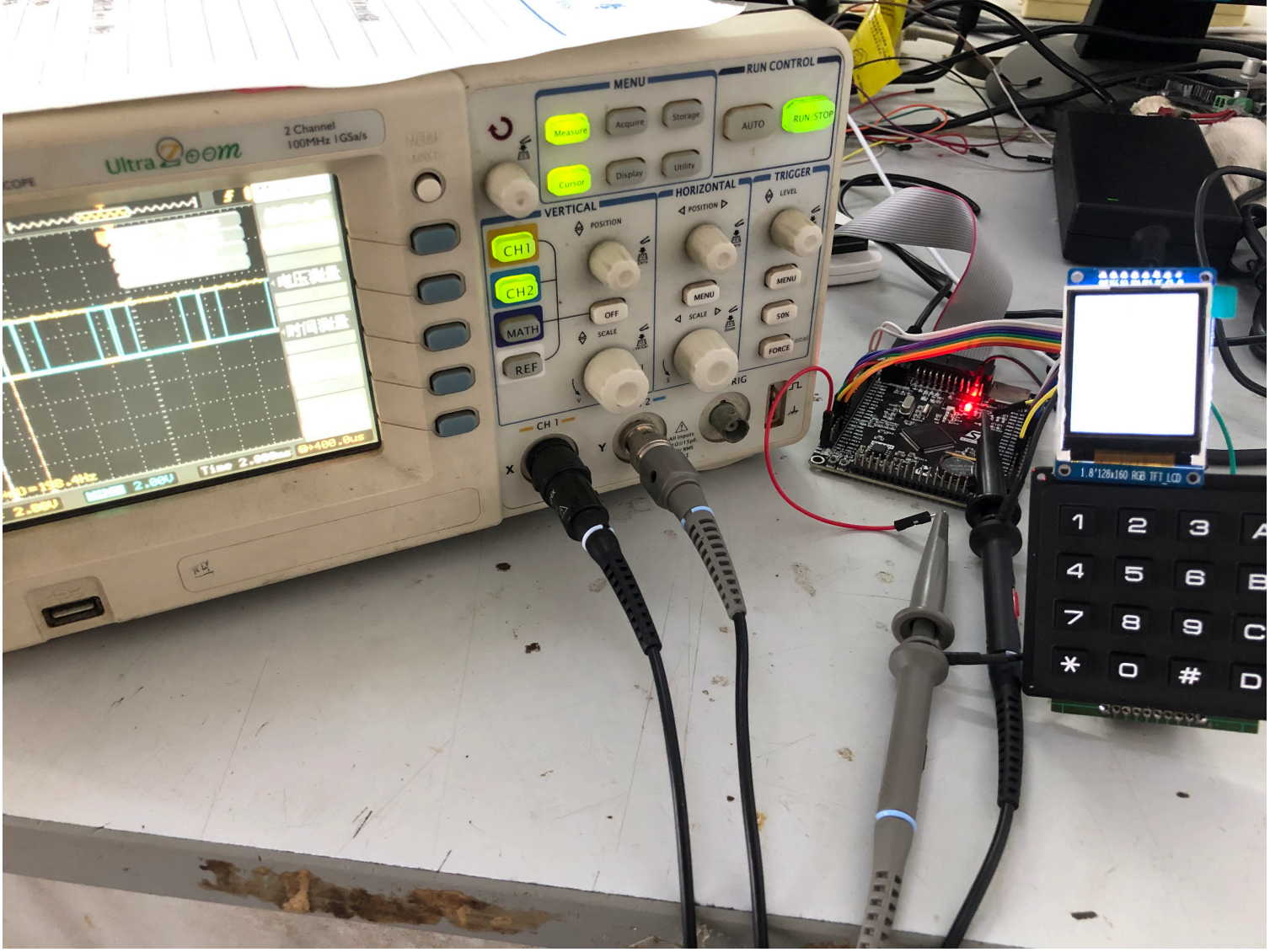
概述

实现效果：


GND VCC SCL SDA RES DC CS BL
Freq: 150
Duty: 80
Mode:
A Change Freq
1.8'128x160 RGB TFT_LCD
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D



该程序实现的功能：用STM32f4输出可调频率（100~500HZ）和占空比（0—100）互补方波，通过矩阵键盘输入数据，用液晶显示屏显示参数。

修改占空比的方式：在TIM2的通用定时器的中断（21kHZ）中，用TIM_SetCompare函数设置比较值。

修改频率的方式：只需要修改预分频的系数就好了，我尝试了两个方式，第一种是用TIM_PrescalerConfig()函数修改预分频器的值，但是输出不准确。因此我采用初始化的函数TIM_PWM_Init()装值。

该程序难点在于矩阵键盘，屏幕和单片机三者的结合，包括如何正确读出矩阵键盘输入的字符串和数字，如何设置屏幕显示，如何设置代码的顺序，使整个程序各个部分能独立工作不相互干扰。从开始到完成的5个小时中，我4个多钟都在解决上面的难点。我这里设置了三个mode区分修改频率，修改占空比和正常输出状态。

话不多说，直接上代码（显示屏的底层代码略过）：

主函数代码

```
#include "sys.h"
#include "delay.h"
#include "usart1.h"
#include "usart2.h"
#include "led.h"
```

```
#include "pwm.h"
#include "timer.h"
#include "adc.h"
#include "pid.h"
#include "key.h"
#include "keyboard.h"
#include "data.h"
#include "string.h"
#include "math.h"
#include "Lcd_Driver.h"
#include "GUI.h"
#include "QDTFT_demo.h"
#include "encoder.h"

char clear[20];
char freq_buffer[3];//频率转化为字符串
char duty_buffer[3];//占空比转化为字符串
int frequency=300;//初始频率
u8 duty=50;//初始占空比
u8 mode=2;//初始mode，其中0:set freq 1:set duty 2:ordinary
u32 psc=1400;//分频系数，初始300

void CLEAR(int i)               //显示屏清除某行的多少的函数
{
        int n;
        for(n=0;n<i;n++)
        {
                clear[n]=' ';
        }
        clear[n]='\0';
}
void show_fd()//显示频率和占空比
{
        sprintf(freq_buffer,"%d",frequency);
        Gui_DrawFont_GBK16(60,40,BLACK,WHITE,freq_buffer);
        sprintf(duty_buffer,"%d",duty);
  Gui_DrawFont_GBK16(60,60,BLACK,WHITE,duty_buffer);
}

void set_freq()//设置频率
{
        u8 i=0;
        u8 receive_char[1],received_freq[4];
        LED1=0;
        Gui_DrawFont_GBK16(30,100,BLACK,WHITE,"Change Freq");//提示设置频率状态
        while(mode==0)
```

```
        {
                receive_char[0]=keyboard_output(0);
                if(receive_char[0]>='0'&& receive_char[0]<='9')//有输入
                {
                        received_freq[i]=receive_char[0]-'0';
                  i++;
                }
                if(receive_char[0]=='*')//确定按键
                {
                        frequency=100*received_freq[0]+10*received_freq[1]+received_freq[2]
                        if(frequency>500) frequency=500;
                        if(frequency<100) frequency=100;
                        psc=420000/frequency;
//                      TIM_Cmd(TIM1,DISABLE);//原本想用这个函数调整频率，但因不准确作罢
//                      TIM_PrescalerConfig(TIM1,psc-1,TIM_PSCReloadMode_Immediate);
//                      TIM_Cmd(TIM1,ENABLE);
                        TIM1_PWM_Init(psc,400-1);
                        delay_ms(50);
                        mode=2;
                }
        }
        LED1=1;//设置完成，灯熄灭
        CLEAR(12);
        Gui_DrawFont_GBK16(30,100,BLACK,WHITE,clear);//设置完之后清除提示
}

void set_duty()//设置占空比
{
        u8 i=0;
        u8 receive_char[1],received_duty[3];
        LED1=0;
        Gui_DrawFont_GBK16(30,100,BLACK,WHITE,"Change duty");//提示设置频率状态
        while(mode==1)
        {
                receive_char[0]=keyboard_output(0);
                if(receive_char[0]>='0'&& receive_char[0]<='9')//有输入
                {
                        received_duty[i]=receive_char[0]-'0';
                  i++;
                }
                if(receive_char[0]=='*')//确定按键
                {
                        duty=10*received_duty[0]+received_duty[1];
                        if(duty>=100) duty=100;
                        if(duty<=0) duty=0;
                        mode=2;
```

```
                }
        }
        LED1=1;
        CLEAR(12);
        Gui_DrawFont_GBK16(30,100,BLACK,WHITE,clear);//设置完之后清除提示
}
/*****************************************
函数名称:TIM_Init
功    能:时钟设置
具体功能:将时钟1设置成PWM，时钟2设置成中断
参    数:无
返 回 值:无
备    注:时钟1和时钟8为168MHz;时钟2~7为84MHz;
                                包括中断(timer)和PWM(pwm)
*****************************************/
void TIM_Init(void)
{
        TIM2_Int_Init(400-1,20-1);//通用定时器，21kHZ
        TIM1_PWM_Init(1400-1,400-1);//100HZ：4200 300HZ：1400 500HZ：840
}

/*****************************************
函数名称:Program_Init
功    能:系统初始化（包括各种初始化的定义）
参    数:无
返 回 值:无
*****************************************/
void Program_Init()
{
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分组2
        TIM_Init();                                     //初始化定时器（时钟）
        delay_init(168);        //初始化延时函数
        LED_Init();                                     //初始化单片机上的LED灯
        GPIO_E2_Init();                 //初始化GPIO PE2（控制端口）
        Keyboard_Init();                //初始化矩阵键盘
        {                                                                       //初
                SystemInit();   //System init.
                Lcd_Init();
                LCD_LED_SET;
                TFT_Init_Show();        //初始化显示（固定不变）
        }
}

//主函数
int main(void)
{
```

```
        char key[1]=' ';
        Program_Init(); //程序初始化，包括各种初始化代码
        while(1)
        {
                show_fd();
                key[0]=keyboard_output(0);
                Gui_DrawFont_GBK16(10,100,BLACK,WHITE,key);
                if(key[0]=='A')//设置频率
                        mode=0;
                if(key[0]=='B')//设置占空比
                        mode=1;
                while(mode==0)
                {
                        set_freq();
                }
                while(mode==1)
                {
                        set_duty();
                }
        }
}

void TIM2_IRQHandler(void)//在通用中断中修改占空比
{
        if(TIM_GetITStatus(TIM2,TIM_IT_Update)==SET)//溢出中断
        {
                TIM_SetCompare1(TIM1,psc*duty/100);//占空比的设置
        }
        TIM_ClearITPendingBit(TIM2,TIM_IT_Update);  //清除中断标志位
}
```

4*4矩阵键盘的代码

```
#include "keyboard.h"
#include "delay.h"
#include "sys.h"
#include "usart1.h"

/*四行输出，四列输入*/
void R_Out_C_Input(void){
    GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);//使能GPIOD
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;                                   //普通输出模式
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;                          //开漏输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;  //100MHz
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;                                    //输入模式
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;                                    //上拉
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}

/*四列输出，四行输入*/
void C_Out_R_Input(void){
                GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);       //使能GPIOD

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;                                           //普
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;                                  //开漏输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;          //100MHz
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;                                            //
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;                                            //
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}

//初始化Keyboard，即将扫描方式调整成为四行输出，四列输入
void Keyboard_Init(void)
{
        R_Out_C_Input();
}

u8 matrixkey(int mode){
        u8 row = 0;                                             // 行号
        u8 column = 0;                          // 列号
        static u8 key =1;
        GPIO_Write(GPIOD,0);  // 这里主函数里初始化后，PD0-3输出，PD4-7输入，让PD0-3输出0
        // 按键没有按下的时候，PD4-7读入均为1，按下后对应位为0
        if(key&&((GPIO_ReadInputData(GPIOD)&0xF0)!=0xF0)){
                delay_ms(10);
                column =  GPIO_ReadInputData(GPIOD)&0XF0; // 获取列值
                C_Out_R_Input();
```

```
                delay_ms(1);
                GPIO_Write(GPIOD,0);
                row = GPIO_ReadInputData(GPIOD)&0X0F;                   // 获取行值
                key = 0;
                R_Out_C_Input();
        }
        if(mode)key = 1;
        if( (GPIO_ReadInputData(GPIOD)&0XF0) == 0XF0){
                delay_ms(10);
                key = 1;
        }
  return row+column;

}
/*****************************************
函数名称:keyboard_output
功    能:矩阵键盘检测输出函数
参    数:int mode(应该是连按相关)
返 回 值:char
*****************************************/
char keyboard_output(int mode)
{
        char keyboard_out=' ';
        switch(matrixkey(0)){                                   //将按下按键的值存入keyboard_o
     /*第一行*/
                                case 0xee:keyboard_out='1';; break; case 0xde:keyboard_out=
                                case 0xbe:keyboard_out='3';;break; case 0x7e:keyboard_out='

                /*第二行*/
                                case 0xed:keyboard_out='4'; break; case 0xdd:keyboard_out='
                                case 0xbd:keyboard_out='6'; break; case 0x7d:keyboard_out='

                /*第三行*/
                                case 0xeb:keyboard_out='7'; break; case 0xdb:keyboard_out='
                                case 0xbb:keyboard_out='9';break; case 0x7b:keyboard_out='C

    /*第四行*/
                                case 0xe7:keyboard_out='*';break; case 0xd7:keyboard_out='0
                                case 0xb7:keyboard_out='#';break; case 0x77:keyboard_out='D
                default: keyboard_out=' ';break;
               }
        return keyboard_out;
}
```

对今天的工作总结就到这，继续加油~