# 2020 －2021 学年第 1 学期

编号：

****

### 实 验 报 告



实验课程名称 《嵌入式系统设计基础》

专 业 班 级 电子信息工程1901

学 生 姓 号 31902002

学 生 姓 名 曾瑶瑶

实验指导老师 朱胜

**浙大城市学院实验报告**

课程名称 《嵌入式系统设计基础》

实验项目名称 《实验一》《实验二》《实验三》

学生姓名 曾瑶瑶 专业班级 电子信息工程1901 学号 31902002

实验成绩 指导老师（签名 ） 日期

目录

实验一（GPIO接口）

实验二（Delay\_s函数）

实验三（按键输入）

实验四（串口通信）

实验五（定时器中断）

实验六（PWM输出）

实验七（PWM输入捕获）

实验八（TFTLCD 显示）

# 实验一（GPIO接口）

## 一. 实验目的和要求

了解IO 口的高低电平控制

了解IO 口作 为输出使用的方法

## 实验内容

①：灯1隔1s闪一次，并且一直闪烁；

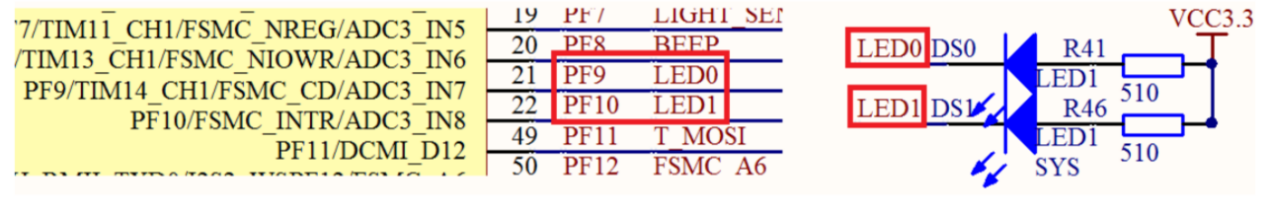
②：灯2隔2s闪一次，同时蜂鸣器叫一次；

③：灯2闪烁5次时，灯1和灯2灭，此时蜂鸣器长鸣5s；

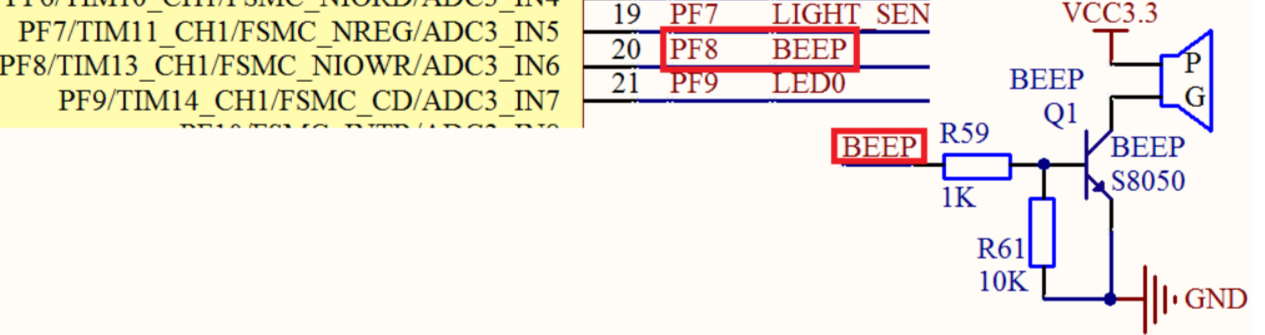
④：重复以上过程。

## 硬件设计

### （1）LED 与 STM32F4 连接原理图



### 蜂鸣器STM32F4 连接原理图



## 四. 实验代码



## 五. 实验结果与分析

①：灯1隔1s闪一次，并且一直闪烁；

②：灯2隔2s闪一次，同时蜂鸣器叫一次；

③：灯2闪烁5次时，灯1和灯2灭，此时蜂鸣器长鸣5s；

④：重复以上过程。

## 六. 讨论、心得

对 STM32F4 开发的软件和硬件平台都有了更较深入的了解了，通过实例，由浅入深，一步步的学习 STM32F4。STM32F4的内部资源非常丰富，最简单的外设，然后一步步深入。结合详细的代码及解释，入手 STM32F4 的各种外设，通过本篇的学习，学会 STM32F4 绝部分外设的使用。

对GPIO更进一步的认识何使用，调用接口过程中，要注意接口函数的定义和引用。

# 实验二（Delay\_s函数）

## 一. 实验目的和要求

掌握时钟系统的结构和原理；

掌握时钟系统函数的编写；

利用时钟系统函数控制LED和蜂鸣器；

## 二．实验内容

①：编写函数，使用此函数把系统时钟变为42MHz；

②：编写delay\_s函数实现以秒为单位的延迟功能；

③：每2s，LED灯闪烁1次，每3s蜂鸣器响一次；

## 三．实验代码

#include "system.h"

#include "stm32f4xx.h"

#include "SysTick.h"

#include "led.h"

#include "beep.h"

int i=0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : RCC\_HSE\_Config

\* 函数功能 : 自定义系统时钟

\* 输 入 : pllm：VCO 输入时钟 分频因子，范围0-63

plln：VCO 输出时钟 倍频因子，范围192-432

pllp：PLLCLK 时钟分频因子，范围2, 4, 6, or 8

pllq：OTG FS,SDIO,RNG 时钟分频因子，范围4-15

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_s(u32 i)

{

while(i--){

delay\_ms(1000);}

}

int main()

{

SysTick\_Init(168);

RCC\_HSE\_Config(8,336,8,7);

LED\_Init();

BEEP\_Init();

while(1)

{

j++;

if(j%3==0)

{BEEP=1;

}

else

{BEEP=0;}

if(j%2==0){

LED1=0;}

else{LED1=1;}

delay\_s(1);

}

}

void delay\_s(u32 i)

{

while(i--){

delay\_ms(1000);}

}

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);

改变时钟的频率，由2变成8.

while(1)

{

j++;

if(j%3==0)

{BEEP=1;

}

else

{BEEP=0;}

if(j%2==0){

LED1=0;}

else{LED1=1;}

delay\_s(1);

}

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

## 四．实验结果与分析

①：编写函数，使用此函数把系统时钟变为42MHz；

②：编写delay\_s函数实现以秒为单位的延迟功能；

③：每2s，LED灯闪烁1次，每3s蜂鸣器响一次；

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);改变时钟的频率，由2变成8.

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

## 五. 讨论、心得

对于改变晶振时间后，对于时间的改变，以至于时间一直对不上。后来了解了晶振频率的含义后对程序进行改变后达到秒的效果。

# 实验三（按键输入）

## 一. 实验目的和要求

利用板载的 4 个按键，来控制板载的两个 LED 的亮灭和蜂鸣器的开关；

了解到 STM32F4 的 IO 口作为输入口的使用方法；

## 二．实验内容和原理

用查询模式，实现按键功能：

①：支持连续按和不连续按；

②：按key0，只亮LED0；

③：按key1，只亮LED1；

④：按key2，只响蜂鸣器；

⑤：按key\_up，LED0和LED1交替亮灭，同时蜂鸣器隔3s响一次。

## 三．实验代码

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);

改变时钟的频率，由2变成8.

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

#include "system.h"

#include "SysTick.h"

#include "led.h"

#include "beep.h"

#include "key.h"

u8 Keyscan(u8 mode);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

u8 key,i=0;

SysTick\_Init(168);

LED\_Init();

BEEP\_Init();

KEY\_Init();

while(1)

{

key=KEY\_Scan(0); //扫描按键

switch(key)

{

case KEY\_UP\_PRESS: LED2=0;LED1=0;BEEP=0;break;//D1D2亮

case KEY1\_PRESS:LED2=0;LED1=1;BEEP=0;break; //D1亮 case KEY2\_PRESS: LED2=1;LED1=1;BEEP=1;break; //蜂鸣器响

case KEY0\_PRESS: LED2=1;LED1=0;BEEP=0;break; //D2亮

}

}

}

key=KEY\_Scan(0); //扫描按键

switch(key)

{

case KEY\_UP\_PRESS: LED2=0;LED1=0;BEEP=0;break;//D1D2亮

case KEY1\_PRESS:LED2=0;LED1=1;BEEP=0;break; //D1亮 case KEY2\_PRESS: LED2=1;LED1=1;BEEP=1;break; //蜂鸣器响

case KEY0\_PRESS: LED2=1;LED1=0;BEEP=0;break; //D2亮

}

## 四．实验结果与分析

用查询模式，实现按键功能：

①：支持连续按和不连续按；

②：按key0，只亮LED0；

③：按key1，只亮LED1；

④：按key2，只响蜂鸣器；

⑤：按key\_up，LED0和LED1交替亮灭，同时蜂鸣器隔3s响一次。

## 五. 讨论、心得

主函数代码比较简单，先进行一系列的初始化操作，然后在死循环中调用按键扫描函数KEY\_Scan()扫描按键值。

# 实验四（串口通信）

## 一. 实验目的和要求

了解学习 STM32F4 的串口；

使用 STM32F4 的串口来发送和接收数据；

## 二．实验内容和原理

在屏幕显示，以下内容：

“欢迎来到串口通信实验，请输入功能号：”

①：跑马灯

②：关闭所有灯

③：蜂鸣器

④：关闭蜂鸣器

## 三．实验代码

int main()

{

u8 i=0;

SysTick\_Init(168); //时钟初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级 分两组

LED\_Init();

BEEP\_Init();//蜂鸣器初始化

USART1\_Init(115200); //波特率设置

printf("欢迎来到串口通信实验\r\n");

printf("请输入功能号\r\n");

printf("1.跑马灯\r\n");

printf("2.关闭所有灯\r\n");

printf("3.·蜂鸣器\r\n");

printf("4.关闭蜂鸣器÷\r\n");

while(1)

{

if(USART1\_RX\_STA&0x8000) //接收完成

{

i = USART1\_RX\_BUF[0];

printf("请输入你的功能号：c\n",i); //向串口1发送数据

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET); //等待发送结束

USART1\_RX\_STA = 0;

}

switch(i)

{

case '1':

LED1 = 0;

LED2 = 1;

delay\_ms(500);

LED1 = 1;

LED2 = 0;

delay\_ms(500);

case '2':

LED1 = LED2 = 1;

break;

case '3':

BEEP = 1;

break;

case '4':

BEEP = 0;

break;

}

delay\_ms(10);

}

}

switch(i)

{

case '1':

LED1 = 0;

LED2 = 1;

delay\_ms(500);

LED1 = 1;

LED2 = 0;

delay\_ms(500);

case '2':

LED1 = LED2 = 1;

break;

case '3':

BEEP = 1;

break;

case '4':

BEEP = 0;

break;

}

delay\_ms(10);

}

}

用switch查找功能号来分别实现内容。

## 五. 实验结果与分析

直接对管脚进行高低电平的设置，进行定时设置，不适用延时。

在屏幕显示，以下内容：

“欢迎来到串口通信实验，请输入功能号：”

①：跑马灯

②：关闭所有灯

③：蜂鸣器

④：关闭蜂鸣器

## 六. 讨论、心得

因为使用到了串口的中断接受，所以一定要进行设置，否则函数不能配置中断使能。还要把中断的优先级分组。

了解学习 STM32F4 的串口；使用 STM32F4 的串口来发送和接收数据；

# 实验五（定时器中断）

## 一. 实验目的和要求

了解使用 STM32F4 的通用定时器，STM32F4 的定时器功能十

分强大，有TIME1和TIME8等高级定时器，也有 TIME2~TIME5，TIM9~TIM14 等通用定时器，还有 TIME6 和 TIME7 等基本定时器，总共达 14 个定时器之多；

## 二．实验内容

①：定时器1实现LED1每隔1s闪烁一次

②：定时器2实验蜂鸣器每隔2s响一次

③：屏幕循环输出：

“正在进行定时器中断实验

LED1隔1s闪烁

蜂鸣器隔2s响”

## 四. 实验代码

#include "system.h"

#include "SysTick.h"

#include "led.h"

#include "time.h"

#include "beep.h"

#include "usart.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

SysTick\_Init(168);

LED\_Init();

BEEP\_Init();

USART1\_Init(115200);

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级分组 分2组

TIM3\_Init(10000-1,8400-1);

TIM4\_Init(20000-1,8400-1);

while(1)

{

printf("正在进行定时器中断实验，LED1隔1s闪烁，蜂鸣器隔2s响\n");

delay\_ms(2000);

}

}

直接对管脚进行高低电平的设置，进行定时设置，不适用延时。

TIM3\_Init(10000-1,8400-1); //定时器时钟84MHz 分频系数8400 计10000次为1s

TIM4\_Init(20000-1,8400-1);

LED1隔1s闪烁，蜂鸣器隔2s响

printf("正在进行定时器中断实验，LED1隔1s闪烁，蜂鸣器隔2s响\n");

delay\_ms(2000);

分别每隔2S的时间进行一次打印

## 五. 实验结果与分析

①：定时器1实现LED1每隔1s闪烁一次

②：定时器2实验蜂鸣器每隔2s响一次

③：屏幕循环输出：

“正在进行定时器中断实验

LED1隔1s闪烁

蜂鸣器隔2s响”

对于中断的设置，一定要清楚晶振的大小，以确定一个机器的周期。对于一开始没有了解时间的公式乱改数据，导致实验结果有很大的偏差。每个机器的的不同导致开始设置的时间是不同的。

## 六. 讨论、心得

了解使用 STM32F4 的通用定时器，STM32F4 的定时器功能十

分强大，有TIME1和TIME8等高级定时器，也有 TIME2~TIME5，TIM9~TIM14 等通用定时器，还有 TIME6 和 TIME7 等基本定时器，总共达 14 个定时器之多；

# 实验六（PWM输出）

## 一. 实验目的和要求

了解使用 STM32F4 的通用定时器；

使用 STM32F4 的 TIM4 来产生 PWM 输出；

## 二．实验内容

呼吸灯：用两个定时器控制两个LED灯实现呼吸灯功能要求一路频率为2KHZ,另一路为3KHZ。

## 三. 实验代码

int main()

{

u16 i;

u8 fx=0;

u16 a;

u8 fy=0;

SysTick\_Init(168);

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级分组 分2组

LED\_Init();

BEEP\_Init();

TIM14\_CH1\_PWM\_Init(500-1,84-1);

//时钟频率84Mhz 分频系数84 计数频率1mhz 重装载值500 pwm频率1/500=2khz

TIM13\_CH1\_PWM\_Init(333-1,84-1);

while(1)

{

if(fx==0)

{

i++;

if(i==300)

{

fx=1;

}

}

else

{

i--;

if(i==0)

{

fx=0;

}

}

if(fy==0)

{

a++;

if(a==400)

{

fy=1;

}

}

else

{

a--;

if(a==0)

{

fy=0;

}

}

TIM\_SetCompare1(TIM14,i); //i值最大可以取499，因为ARR最大值是499.

TIM\_SetCompare1(TIM13,a);

delay\_ms(10);

}

}

## 五. 实验结果与分析

呼吸灯：用两个定时器控制两个LED灯实现呼吸灯功能要求一路频率为2KHZ,另一路为3KHZ。

直接对管脚进行高低电平的设置，进行定时设置，不适用延时。

TIM14\_CH1\_PWM\_Init(500-1,84-1);

//时钟频率84Mhz 分频系数84 计数频率1mhz 重装载值500 pwm频率1/500=2khz

TIM13\_CH1\_PWM\_Init(333-1,84-1);

//时钟频率84Mhz 分频系数84 计数频率1mhz 重装载值333 pwm频率1/333=2khz

呼吸灯和蜂鸣器的频率已知，要根据时钟频率和计数频率设置参数，使得两路不同的频率得到配置。

## 六. 讨论、心得

分强大，有TIME1和TIME8等高级定时器，也有 TIME2~TIME5，TIM9~TIM14 等通用定时器，还有 TIME6 和 TIME7 等基本定时器，总共达 14 个定时器之多；

了解使用 STM32F4 的通用定时器；使用 STM32F4 的 TIM3 来产生 PWM 输出；

# 实验七（PWM输入捕获）

## 一. 实验目的和要求

了解使用 STM32F4 的通用定时器；

通用定时器作为输入捕获的使用。在本章中，我们将用 TIM5 的通道1（PA0）来做输入捕获，捕获 PA0 上高电平的脉宽（用 KEY\_UP 按键输入高电平），通过串口打印高电平脉宽时间；

## 二．实验内容

呼吸灯：使用输入捕获测量PWM输出的信号高电平脉宽。

需要使用一根线讲PWM输出管脚与PAO输入捕获通道连接。

## 四. 实验代码

int main()

{

u16 i;

u8 fx=0;

u16 a;

u8 fy=0;

SysTick\_Init(168);

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级分组 分2组

LED\_Init();

BEEP\_Init();

TIM14\_CH1\_PWM\_Init(500-1,84-1);

//时钟频率84Mhz 分频系数84 计数频率1mhz 重装载值500 pwm频率1/500=2khz

TIM13\_CH1\_PWM\_Init(333-1,84-1);

while(1)

{

if(fx==0)

{

i++;

if(i==300)

{

fx=1;

}

}

else

{

i--;

if(i==0)

{

fx=0;

}

}

if(fy==0)

{

a++;

if(a==400)

{

fy=1;

}

}

else

{

a--;

if(a==0)

{

fy=0;

}

}

TIM\_SetCompare1(TIM14,i); //i值最大可以取499，因为ARR最大值是499.

TIM\_SetCompare1(TIM13,a);

delay\_ms(10);

}

}

## 五. 实验结果与分析

使用输入捕获测量PWM输出的信号高电平脉宽。需要使用一根线讲PWM输出管脚与PAO输入捕获通道连接。

直接对管脚进行高低电平的设置，进行定时设置，不适用延时。

呼吸灯和蜂鸣器的频率已知，要根据时钟频率和计数频率设置参数，使得两路不同的频率得到配置。

## 六. 讨论、心得

了解使用 STM32F4 的通用定时器，STM32F4 的定时器功能十

分强大，有TIME1和TIME8等高级定时器，也有 TIME2~TIME5，TIM9~TIM14 等通用定时器，还有 TIME6 和 TIME7 等基本定时器，总共达 14 个定时器之多；

了解使用 STM32F4 的通用定时器；使用 STM32F4 的 TIM3 来产生 PWM 输出；

使用输入捕获测量PWM输出的信号高电平脉宽。需要使用一根线讲PWM输出管脚与PAO输入捕获通道连接。

# 实验八（TFTLCD 显示）

## 一. 实验目的和要求

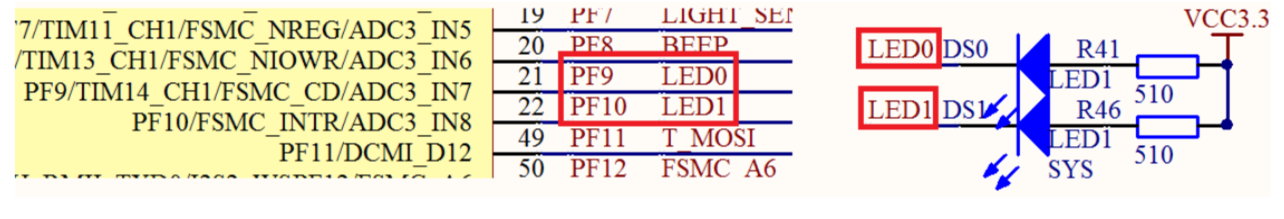
了解ALIENTEK 2.8 寸 TFT LCD 模块，该模块采用 TFTLCD 面板，可以显示 16 位色的真彩图片。在本章中，我们将使用探索者 STM32F4 开发板上的 LCD接口，来点亮 TFTLCD，并实现 ASCII 字符和彩色的显示等功能，并在串口打印 LCD 控制器ID，同时在 LCD 上面显示。

## 二．实验内容

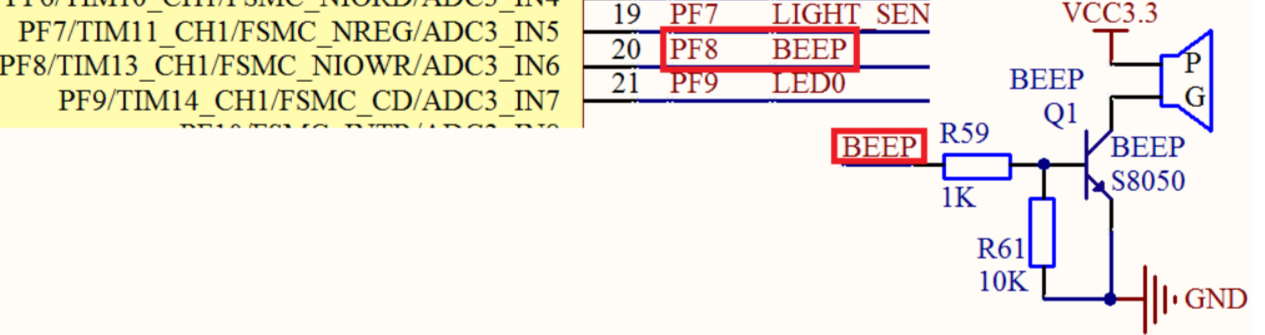
LCD实验将屏幕划成4份，屏幕显示奥运五环，并显示奥运五环四个字，并且创意设计。

## 三．硬件设计

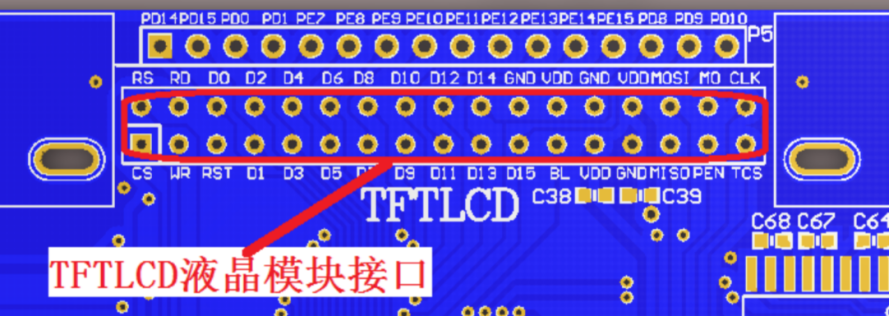
### （1）LED 与 STM32F4 连接原理图



### （2）蜂鸣器STM32F4 连接原理图



### （3）TFTLCD 模块



## 四. 实验代码

#include "system.h"

#include "SysTick.h"

#include "led.h"

#include "usart.h"

#include "tftlcd.h"

#include "picture.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

int i=0;

int j=0;

u16 color=0;

SysTick\_Init(168);

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级分组 分2组

LED\_Init();

USART1\_Init(115200);

TFTLCD\_Init(); //LCD初始化

FRONT\_COLOR=BLACK;

LCD\_Fill(159,0,319,239,BLUE); //xy坐标 返回值 颜色

LCD\_Fill(0,239,159,479,RED);

LCD\_Fill(159,239,319,479,GRAY);

LCD\_ShowFontHZ(100,100,"奥运五环");

for(i=0;i<=319;i++)

for(j=0;j<=479;j++)

{

if((i-80)\*(i-80)+(j-200)\*(j-200)>=900&&(i-80)\*(i-80)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,BLUE);

if((i-160)\*(i-160)+(j-200)\*(j-200)>=900&&(i-160)\*(i-160)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,BLACK);

if((i-240)\*(i-240)+(j-200)\*(j-200)>=900&&(i-240)\*(i-240)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,RED);

if((i-120)\*(i-120)+(j-240)\*(j-240)>=900&&(i-120)\*(i-120)+(j-240)\*(j-240)<=1500)

LCD\_Fill(i,j,i,j,YELLOW);

if((i-200)\*(i-200)+(j-240)\*(j-240)>=900&&(i-200)\*(i-200)+(j-240)\*(j-240)<=1500)

LCD\_Fill(i,j,i,j,GREEN);

}

}

对于添色的时候不断的调试，以达到最好的效果。五环的操作，找到圆心，对圆的圈的大小进行调试。

对于4个模块进行分割

LCD\_Fill(159,0,319,239,BLUE);

LCD\_Fill(0,239,159,479,RED);

LCD\_Fill(159,239,319,479,GRAY);

剩下的成为白色

LCD\_ShowFontHZ(100,100,"奥运五环");

for(i=0;i<=319;i++)

for(j=0;j<=479;j++)

{

if((i-80)\*(i-80)+(j-200)\*(j-200)>=900&&(i-80)\*(i-80)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,BLUE);

if((i-160)\*(i-160)+(j-200)\*(j-200)>=900&&(i-160)\*(i-160)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,BLACK);

if((i-240)\*(i-240)+(j-200)\*(j-200)>=900&&(i-240)\*(i-240)+(j-200)\*(j-200)<=1500)

LCD\_Fill(i,j,i,j,RED);

if((i-120)\*(i-120)+(j-240)\*(j-240)>=900&&(i-120)\*(i-120)+(j-240)\*(j-240)<=1500)

LCD\_Fill(i,j,i,j,YELLOW);

if((i-200)\*(i-200)+(j-240)\*(j-240)>=900&&(i-200)\*(i-200)+(j-240)\*(j-240)<=1500)

LCD\_Fill(i,j,i,j,GREEN);

}

}

这一段则是附上文字和奥运五环。

## 五. 实验结果与分析

LCD实验将屏幕划成4份，屏幕显示奥运五环，并显示奥运五环四个字，并且创意设计。

## 六. 讨论、心得

了解ALIENTEK 2.8 寸 TFT LCD 模块，该模块采用 TFTLCD 面板，可以显示 16 位色的真彩图片。在本章中，我们将使用探索者 STM32F4 开发板上的 LCD接口，来点亮 TFTLCD，并实现 ASCII 字符和彩色的显示等功能，并在串口打印 LCD 控制器ID，同时在 LCD 上面显示。