华南理工大学

South China University of Technology

设计报告书

题目：一种包含时间、温湿度显示及闹钟功能的支持单、双人飞机战斗游戏的游戏机系统

教师评语	教师签名： 日期：
成绩评定	
备注	

目录

游戏机

一、选题背景

赛元单片机的 51 系列单片机，是增强型的高速 1T 8051 Flash 微控制器。利用其超高速内核和内部集成的 UART、SPI、IIC 等多种通信接口，便于与不同接口的器件和设备相连。本次赛元杯比赛，选择了设计游戏的赛题，一款完成度高的游戏，需要集成设备的输入输出控制，算法逻辑、UI 等软硬件的基础元素，也需要设计者具备想象力,软硬件整合能力。本赛题我们使用赛元单片机的硬件 SPI 协议通过 USCI0 串口驱动 TFT 显示屏，配合其他外设，制作了一款集成单人操作的飞行射击游戏模式、双人互动的飞行射击游戏模式以及兼有时钟显示、温湿度显示以及闹钟功能的参数显示模式。

二、方案论证(设计理念)

本次赛元科技竞赛，要求利用赛元 51 系列单片机，设计完成一个游戏机，要求游戏机能够实现一款飞机对战游戏，并包含显示时钟温湿度等功能。

一开始需要完成整个电路的焊接，我们先根据焊接指引文件完成焊接工作，之后顺利通过了游戏机成品板的系统测试。TFT 屏幕是单片机产品与用户交互的关键组件，游戏程序互动页面的搭建和温湿度数据的显示的基础，故首先通过封装一系列的函数,采用硬件触发的 SPI 协议完成了对 TFT 屏幕驱动的基础配置。在屏幕显示方面，我们采取只刷新变化的部分，不改变背景的动态显示方案。在游戏机 UI 设计方面，秉持简洁美观的设计理念，只预留了单、双人对战与温湿度及时间显示的三种模式，并对模式选择的阴影部分加以流水的效果显示。

首先，在游戏搭建方面，我们首先完成游戏角色图像的绘制，通过对 TFT 屏幕封装的绘制点和线的函数，完成了 28*28 像素的我方战机、16*16 像素的敌方飞机、2*2 像素的子弹、44*40 像素的 BOSS 等游戏元素的图像绘制，随后通过 TFT 屏幕动态显示的方案以及按键扫描函数搭建一步步实现敌我战机移动、子弹移动等动态效果。为了实现飞机战斗的游戏机制，我们在完成飞机和子弹的动态

移动后，通过计算飞机和子弹（敌方飞机与我方子弹、我方飞机与敌方子弹）的相对临界位置，进行积分和命损的判定，从而实现获得积分而关卡晋级和损失生命点数而死亡的效果。随后，为了扩展实现双人游戏，我们封装了一系列信息传输的函数，通过 UART0 串口将两台赛元 51 单片机连接，实时获取其按键及摇杆的数据并将“敌机”的图像显示在 TFT 屏幕上，实现双人对战。

其次，为了得到动态的音乐效果，我们采用 PWM 技术控制的蜂鸣器来实现音调与响度的变化，得到了游戏所用的背景音乐。

再者，为了实现实时时钟的显示，在每次进入时钟模式时，我们都需要通过 UART2 串口连接 ESP8266，再借由 AT 指令将其配置为 STA 模式。在初始化 ESP8266 完毕之后，通过一系列的 AT 指令使其连接到获取时钟信息的接口，获取最新的实时时钟信息。同时通过解析接收到的信息将其转换为年月日、时分秒的形式，再通过每秒软件递增的方式获得当前的系统时间。

闹钟功能也是游戏机的重要组成部分，基于 TIMER4 的定时器，我们设计了闹钟功能，在任何界面，一旦到达预设时间，便会弹出 TIMER OUT 的提示弹窗对玩家进行提醒，5s 以后游戏继续正常运行。

最后，为了获得温湿度数据，我们通过将 USCI3 配置为 TWI 协议（兼容 I2C），既使用硬件触发 TWI 协议连接 SHT30 温湿度模块，将其配置为单次读取数据模式，实现了温湿度显示功能。

三、过程论述

1.TFT 屏幕驱动配置

（1）利用定时器 Timer2 的溢出中断，封装延时函数（SC_it.c\Delay.h）

```
105 void Timer2Interrupt()      interrupt 5
106 {
107     /*TIM2_it write here begin*/
108     /*TIM2_it write here*/
109     if (delay_begin == 1)
110     {
111         if (ms_cnt > 0)
112         {
113             ms_cnt --;
114         }
115         else
116         {
117             delay_begin = 0;
118         }
119     }
```

```
1  //Delay.c
2  #include "Delay.h"
3  #include "SC_it.h"
4
5  //int16_t ms_cnt = 0;
6  //uint8_t delay_begin = 0;
7
8  void Delay_ms(int16_t ms)
9  {
10     ms_cnt = ms;
11     delay_begin = 1;
12     while (delay_begin != 0){}
13 }
14
```

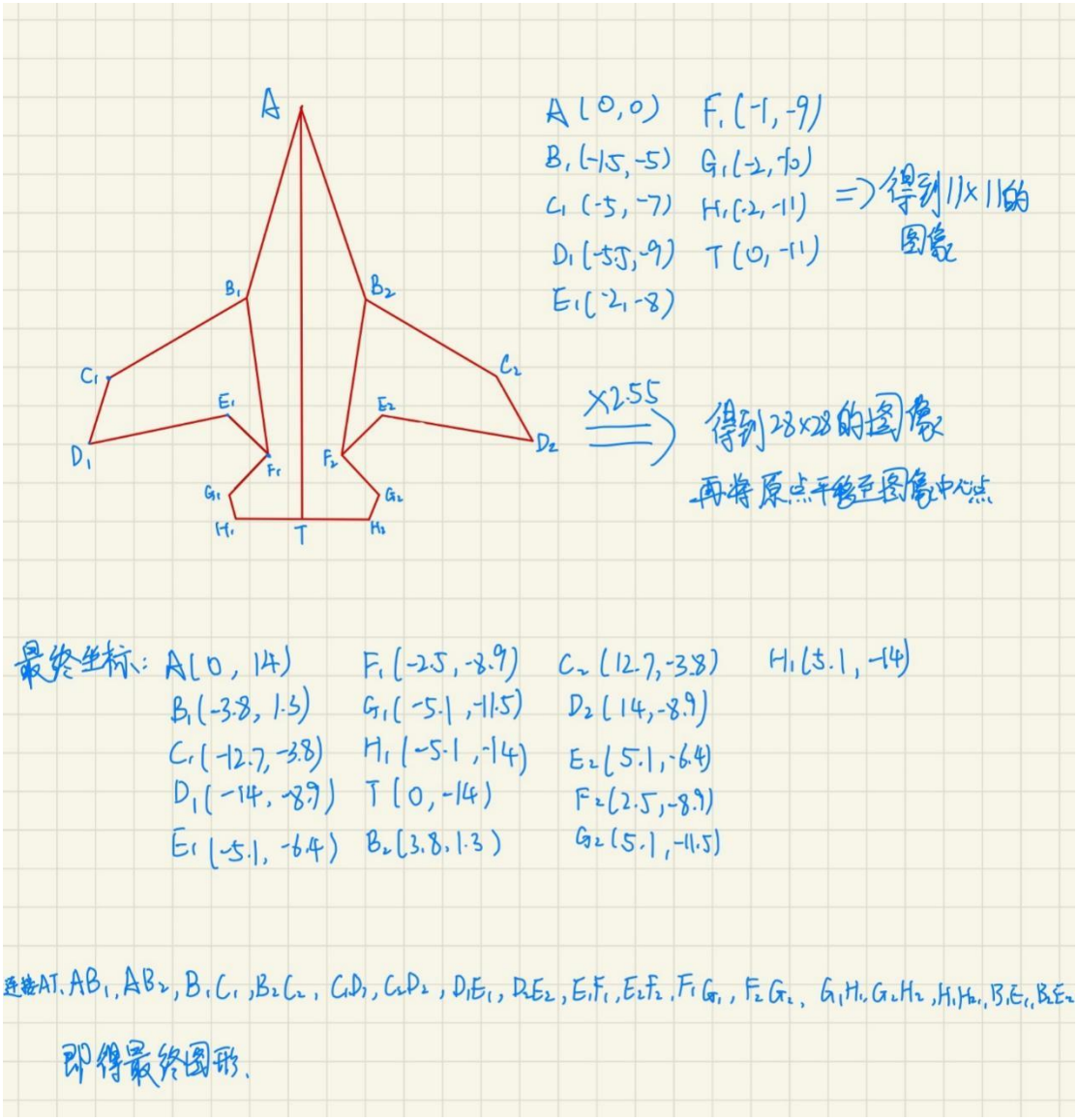
（2）配置屏幕以及基础函数封装（展示头文件中部分配置代码 Lcd_Driver.h\GUI.h）

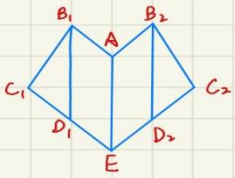
```
1  #ifndef __LCD_DRIVER_H__
2  #define __LCD_DRIVER_H__
3
4  #include "SC_Init.h"  // MCU initialization header file, including all firmware library header files
5
6  extern u16 POINT_COLOR;
7  extern u16 BACK_COLOR;
8
9  #define RED   0xf800
10 #define GREEN 0x07e0
11 #define BLUE  0x001f
12 #define WHITE 0xffff
13 #define BLACK 0x0000
14 #define YELLOW  0xFFE0
15 #define GRAY0   0xEF7D    //灰色0 3165 00110 001011 00101
16 #define GRAY1   0x8410      //灰色1      00000 000000 00000
17 #define GRAY2   0x4208      //灰色2  1111111111011111
18 #define TFT_DC(x)   ((x == 0)? (GPIO_WriteLow(GPIO0, GPIO_PIN_2)) :(GPIO_WriteHigh(GPIO0, GPIO_PIN_2)))
19 #define TFT_REST(x)    ((x == 0)? (GPIO_WriteLow(GPIO0, GPIO_PIN_1)) :(GPIO_WriteHigh(GPIO0, GPIO_PIN_1)))
20 #define TFT_CS(x)   ((x == 0)? (GPIO_WriteLow(GPIO0, GPIO_PIN_3)) :(GPIO_WriteHigh(GPIO0, GPIO_PIN_3)))
21 #define TFT_BL(x)   ((x == 0)? (GPIO_WriteLow(GPIO0, GPIO_PIN_0)) :(GPIO_WriteHigh(GPIO0, GPIO_PIN_0)))
22
23
24 void LCD_GPIO_Init(void);
25 void Lcd_WriteIndex(u8 Index);
26 void Lcd_WriteData(u8 Data);
27 void Lcd_WriteReg(u8 Index,u8 Data);
28 u16 Lcd_ReadReg(u8 LCD_Reg);
29 void Lcd_Reset(void);
30 void Lcd_Init(void);
31 void Lcd_Clear(u16 Color);
32 void Lcd_Clearsome(u16 x1,u16 y1,u16 x2,u16 y2,u16 Color);
33 void Lcd_SetXY(u16 x,u16 y);
34 void Gui_DrawPoint(u16 x,u16 y,u16 Data);
```

```c
#ifndef __GUI_H__
#define __GUI_H__

#include "SC_Init.h"    // MCU initialization header file, including all firmware library header files

u16 LCD_BGR2RGB(u16 c);
void Gui_Circle(u16 X,u16 Y,u16 R,u16 fc);
void Gui_DrawLine(u16 x0, u16 y0,u16 x1, u16 y1,u16 Color);
void Gui_box(u16 x, u16 y, u16 w, u16 h,u16 bc);
void Gui_box2(u16 x,u16 y,u16 w,u16 h, u8 mode);
void DisplayButtonDown(u16 x1,u16 y1,u16 x2,u16 y2);
void DisplayButtonUp(u16 x1,u16 y1,u16 x2,u16 y2);
void Gui_DrawFont_GBK16(u16 x, u16 y, u16 fc, u16 bc, u8 *s);
void Gui_DrawFont_GBK24(u16 x, u16 y, u16 fc, u16 bc, u8 *s);
void Gui_DrawFont_Num32(u16 x, u16 y, u16 fc, u16 bc, u16 num) ;

#endif
```

2.基本图像绘制（敌我战机、子弹）

通过计算像素点的相对位置坐标，利用 tft 屏幕封装的画点和画线函数绘制像素图像（展示设计过程及部分代码 basic_draw.h）





A(0,0) E(0,-3)
B1(-1,1) B2(1,1)
C1(-2,-1) C2(2,-1)
D1(-1,-2) D2(1,-2)

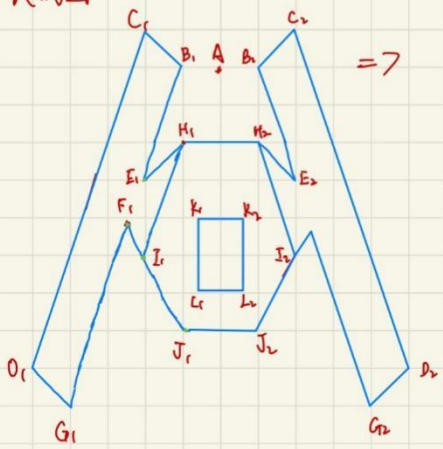
×4 ⇒ 得到16×16的图像
再将原点平移至图像中心

最终坐标：
A(0,4) E(0,-8)
B1(-4,8) B2(4,8)
C1(-8,0) C2(8,0)
D1(-4,-4) D2(4,-4)

连接 AB_1, AB_2, B_1C_1, B_2C_2, C_1E, C_2E, B_1D_1, AE, B_2D_2
得到最终图像

效果图

实物图



=7

A(0,0)
B1(-1,0)
C1(-2,1)
D1(-5,-8)
G1(-4,-9)
J1(-1,-8)

E1(-2,-3)
F1(-7/3,-4)
I1(-2,-5)
H1(-1,-2)
K1(-1/2,-4)
L1(-1/2,-6)

=> 10×9的图像

↓ 坐标×4.4

44×40的图像

再将原点平移至图像中心，得到最终坐标

最终坐标：A(0,17.6)
B1(-4.4,17.6)
C1(-8.8,22)
D1(-22,-17.6)
G1(-17.6,-22)
J1(-4.4,-13.2)

E1(-8.8,4.4)
F1(-10.3,0)
I1(-8.8,-4.4)
H1(-4.4,8.8)
K1(-2.2,0)
L1(-2.2,-8.8)

B2(4.4,17.6)
C2(8.8,22)
D2(22,-17.6)
G2(17.6,-22)
J2(4.4,-13.2)
E2(8.8,4.4)

F2(10.3,0)
I2(8.8,-4.4)
H2(4.4,8.8)
K2(2.2,0)
L2(22,-8.8)

连接各点得到图像

```
void draw_my_plane(int x, int y){
    Gui_DrawLine(x, y-14, x-3.8, y-1.3,POINT_COLOR);
    Gui_DrawLine(x-3.8, y-1.3, x-12.7,y+3.8,POINT_COLOR);
    Gui_DrawLine(x-12.7, y+3.8, x-14, y+8.9,POINT_COLOR);
    Gui_DrawLine(x-14, y+8.9, x-5.1, y+6.4,POINT_COLOR);
    Gui_DrawLine(x-5.1, y+6.4, x-2.5, y+8.9,POINT_COLOR);
    Gui_DrawLine(x-2.5, y+8.9, x-5.1, y+11.5,POINT_COLOR);
    Gui_DrawLine(x-5.1, y+11.5, x-5.1, y+14,POINT_COLOR);
    Gui_DrawLine(x-3.8, y+1.3, x-2.5, y+8.9,POINT_COLOR);
    Gui_DrawLine(x, y-14, x, y+14,POINT_COLOR);
    Gui_DrawLine(x, y-14, x+3.8, y-1.3,POINT_COLOR);
    Gui_DrawLine(x+3.8, y-1.3, x+12.7,y+3.8,POINT_COLOR);
    Gui_DrawLine(x+12.7, y+3.8, x+14, y+8.9,POINT_COLOR);
    Gui_DrawLine(x+14, y+8.9, x+5.1, y+6.4,POINT_COLOR);
    Gui_DrawLine(x+5.1, y+6.4, x+2.5, y+8.9,POINT_COLOR);
    Gui_DrawLine(x+2.5, y+8.9, x+5.1, y+11.5,POINT_COLOR);
    Gui_DrawLine(x+5.1, y+11.5, x+5.1, y+14,POINT_COLOR);
    Gui_DrawLine(x+3.8, y+1.3, x+2.5, y+8.9,POINT_COLOR);
    Gui_DrawLine(x-5.1, y+14, x+5.1, y+14,POINT_COLOR);

}
```

```
void draw_enemy_plane(int x ,int y){
    Gui_DrawLine(x, y-4, x, y+8,POINT_COLOR);
    Gui_DrawLine(x, y-4, x+4, y-8,POINT_COLOR);
    Gui_DrawLine(x, y-4, x-4, y-8,POINT_COLOR);
    Gui_DrawLine(x+4, y-8, x+8, y,POINT_COLOR);
    Gui_DrawLine(x-4, y-8, x-8, y,POINT_COLOR);
    Gui_DrawLine(x+8, y, x, y+8,POINT_COLOR);
    Gui_DrawLine(x-8, y, x, y+8,POINT_COLOR);
    Gui_DrawLine(x+4, y-8, x+4, y+4,POINT_COLOR);
    Gui_DrawLine(x-4, y-8, x-4, y+4,POINT_COLOR);
}

void draw_his_plane(int x, int y){
    Gui_DrawLine(x, y+14, x-3.8, y+1.3,POINT_COLOR);
    Gui_DrawLine(x-3.8, y+1.3, x-12.7,y-3.8,POINT_COLOR);
    Gui_DrawLine(x-12.7, y-3.8, x-14, y-8.9,POINT_COLOR);
    Gui_DrawLine(x-14, y-8.9, x-5.1, y-6.4,POINT_COLOR);
    Gui_DrawLine(x-5.1, y-6.4, x-2.5, y-8.9,POINT_COLOR);
    Gui_DrawLine(x-2.5, y-8.9, x-5.1, y-11.5,POINT_COLOR);
    Gui_DrawLine(x-5.1, y-11.5, x-5.1, y-14,POINT_COLOR);
    Gui_DrawLine(x-3.8, y-1.3, x-2.5, y-8.9,POINT_COLOR);
    Gui_DrawLine(x, y+14, x, y-14,POINT_COLOR);
    Gui_DrawLine(x, y+14, x+3.8, y+1.3,POINT_COLOR);
    Gui_DrawLine(x+3.8, y+1.3, x+12.7,y-3.8,POINT_COLOR);
    Gui_DrawLine(x+12.7, y-3.8, x+14, y-8.9,POINT_COLOR);
    Gui_DrawLine(x+14, y-8.9, x+5.1, y-6.4,POINT_COLOR);
    Gui_DrawLine(x+5.1, y-6.4, x+2.5, y-8.9,POINT_COLOR);
    Gui_DrawLine(x+2.5, y-8.9, x+5.1, y-11.5,POINT_COLOR);
    Gui_DrawLine(x+5.1, y-11.5, x+5.1, y-14,POINT_COLOR);
    Gui_DrawLine(x+3.8, y-1.3, x+2.5, y-8.9,POINT_COLOR);
    Gui_DrawLine(x-5.1, y-14, x+5.1, y-14,POINT_COLOR);
}
```

```
void draw_boss(int x, int y){
    Gui_DrawLine(x-4.4,y-17.6,x-8.8,y-22,POINT_COLOR);
    Gui_DrawLine(x-8.8,y-22,x-22,y+17.6,POINT_COLOR);
    Gui_DrawLine(x-22,y+17.6,x-17.6,y+22,POINT_COLOR);
    Gui_DrawLine(x-17.6,y+22,x-10.3,y,POINT_COLOR);
    Gui_DrawLine(x-4.4,y+13.2,x-10.3,y,POINT_COLOR);
    Gui_DrawLine(x-8.8,y+4.4,x-4.4,y-8.8,POINT_COLOR);
    Gui_DrawLine(x-4.4,y-8.8,x-8.8,y-4.4,POINT_COLOR);
    Gui_DrawLine(x-8.8,y-4.4,x-4.4,y-17.6,POINT_COLOR);
    Gui_DrawLine(x-2.2,y,x-2.2,y+8.8,POINT_COLOR);
    Gui_DrawLine(x+4.4,y-17.6,x+8.8,y-22,POINT_COLOR);
    Gui_DrawLine(x+8.8,y-22,x+22,y+17.6,POINT_COLOR);
    Gui_DrawLine(x+22,y+17.6,x+17.6,y+22,POINT_COLOR);
    Gui_DrawLine(x+17.6,y+22,x+10.3,y,POINT_COLOR);
    Gui_DrawLine(x+4.4,y+13.2,x+10.3,y,POINT_COLOR);
    Gui_DrawLine(x+8.8,y+4.4,x+4.4,y-8.8,POINT_COLOR);
    Gui_DrawLine(x+4.4,y-8.8,x+8.8,y-4.4,POINT_COLOR);
    Gui_DrawLine(x+8.8,y-4.4,x+4.4,y-17.6,POINT_COLOR);
    Gui_DrawLine(x+2.2,y,x+2.2,y+8.8,POINT_COLOR);
    Gui_DrawLine(x-4.4,y-8.8,x+4.4,y-8.8,POINT_COLOR);
    Gui_DrawLine(x+2.2,y,x-2.2,y,POINT_COLOR);
    Gui_DrawLine(x-2.2,y+8.8,x+2.2,y+8.8,POINT_COLOR);
    Gui_DrawLine(x-4.4,y+13.2,x+4.4,y+13.2,POINT_COLOR);

}
```

3.UI 页面设计

（1）主菜单页面设计及逻辑模式

初始界面包含三个模式，时钟、温湿度显示、单人游戏模式以及双人联机游戏模式，通过长按 SELECT 按键进行模式切换（mode=0,1,2 分别代表以上三种模式）、START 按键选择是否进入当前模式(start=0 退回主菜单，start=1 进入各模式)，同样可以通过按 START 键退出所选模式。UI 设计本着简洁明了的理念，界面仅包含三种模式以及框选图标，同时在选中某一模式时候设计了流彩显示，兼顾了美观。

```
0   时钟模式
1   单人游戏
2   双人游戏
*/
uint8_t Mode=0;
/*
0   不开始
1   开始
*/
uint8_t Start=0;
```

```
while(1)
{
    /*<UserCodeStart>*//*<SinOne-Tag><5>*/
    /*****User program*****/
    draw_menu();                //绘制开始页面
    while(Mode==0&&Start==1)
    {
        Lcd_Clear(BACK_COLOR);
        Gui_DrawFont_GBK24(5,10,RED,BACK_COLOR,"CLOCK INIT");
        ESP8266_Cmd_Init();     //esp8266初始化
        ESP_get_data();         //世界时间获取
        Parse_BJ_Time();        //转成年月日，时分秒格式
        Clock_start_draw();
        Lcd_Clear(BACK_COLOR);
        while(Start==1)
        {
            Clock_run();
            if(KEY_Scan(1)==KEY_Start_PRES) Start=0; //判断是否返回主菜单
        }
    }
    while(Mode==1&&Start==1)
    {
        PWM_CmdEX(PWM0_Type,ENABLE);
        music_set(1);
        init_Game1();
        run_games();
        games_reset();
    }
    while(Mode==2&&Start==1)
    {
        PWM_CmdEX(PWM0_Type,ENABLE);
        music_set(1);
        init_Game2();
        run_games2();
        games_reset2();
    }
```

```
120  void draw_menu(void)
121 {
122     PWM_CmdEX(PWM0_Type,DISABLE);
123     POINT_COLOR=BLACK;      //设置字体颜色为黑色
124     BACK_COLOR=GRAY2;       //设置背景颜色为灰色
125     Lcd_Clear(BACK_COLOR);  //将背景颜色置为灰色
126     while(!Start)
127     {
128         if(KEY_Scan(1)==KEY_Start_PRES) Start=1;
129         if(KEY_Scan(1)==KEY_Select_PRES) Mode++;
130         Mode%=3;
131         Gui_DrawFont_GBK24(5,10,RED,BACK_COLOR,"Menu");
132         Gui_DrawFont_GBK24(10,45,GRAY1,BACK_COLOR,"Clock       ");
133         Gui_DrawFont_GBK24(10,61,GRAY1,BACK_COLOR,"Game for one");
134         Gui_DrawFont_GBK24(10,77,GRAY1,BACK_COLOR,"Game for two");
135         if(Mode==0)
136         {
137             Gui_DrawFont_GBK24(10,45,GRAY1,BLUE,"Clock       ");
138         }
139         else if(Mode==1)
140         {
141             Gui_DrawFont_GBK24(10,61,GRAY1,BLUE,"Game for one");
142         }
143         else if(Mode==2)
144         {
145             Gui_DrawFont_GBK24(10,77,GRAY1,BLUE,"Game for two");
146         }
147     }
148 }
```

（2）时钟及温湿度模式 UI 设计

```
//显示年月日，时分秒
sprintf(numbuf,"%d,",(int)year);
Gui_DrawFont_GBK24(0,0,POINT_COLOR,BACK_COLOR,numbuf);
sprintf(numbuf,"%d,",(int)month);
Gui_DrawFont_GBK24(25,0,POINT_COLOR,BACK_COLOR,numbuf);
sprintf(numbuf,"%d",(int)day);
Gui_DrawFont_GBK24(50,0,POINT_COLOR,BACK_COLOR,numbuf);
sprintf(numbuf,"%d:",(int)hour);
Gui_DrawFont_GBK24(0,20,POINT_COLOR,BACK_COLOR,numbuf);
sprintf(numbuf,"%d:",(int)minute);
Gui_DrawFont_GBK24(25,20,POINT_COLOR,BACK_COLOR,numbuf);
sprintf(numbuf,"%d",(int)second);
Gui_DrawFont_GBK24(50,20,POINT_COLOR,BACK_COLOR,numbuf);

SHT30_Read();           //温湿度传感器信息获取
SHT30_Show();           //温湿度传感器信息显示
```

```
232 void SHT30_Show(void)
233 {
234     sprintf(hANDt_buff,"%6.2f*C %6.2f",temperature.fdata,humidity.fdata); //111.01*C 100.01%（保留2位小数）
235     Gui_DrawFont_GBK24(0,50,RED,BACK_COLOR,hANDt_buff);
236 }
```

（3）飞机游戏 UI 设计

飞机游戏起始页面包含游戏名、按键提示以及游戏背景的渲染。

```
12 void draw_start_page(void){
13     Gui_DrawFont_GBK24(30,20,GRAY0,BACK_COLOR,"SPACE");
14     Gui_DrawFont_GBK24(72,40,GRAY0,BACK_COLOR,"WAR");
15     Gui_DrawFont_GBK24(20,70,GRAY1,BACK_COLOR,"Press Key");
16     DisplayButtonUp(98,70,120,86);
17
18     draw_my_plane(14,113);
19     draw_my_plane(64,113);
20     draw_my_plane(114,113);
21 }
```

飞机游戏标头页面包含两个标题：SCROE 表示玩家得分。每击落一架飞机可得一分；LIFE 表示玩家所剩生命值，子弹每次碰撞都会使生命值减一，直至清零时进入结束页面。

```
23 void init_play_interface(void){
24     //设置基础界面
25     Lcd_Clear(BACK_COLOR);
26     Gui_DrawFont_GBK16(0,0,GRAY0,BACK_COLOR,"SCROE");
27     Gui_DrawFont_GBK16(90,0,GRAY0,BACK_COLOR,"LIFE");
28 }
```

```c
void draw_play_title(int8_t score, int8_t life){

    int i=0;
    int start_x = 95;
    Lcd_Clearsome(0,113,128,128,BACK_COLOR);
    Lcd_Clearsome(start_x,20-3,128,20+4,BACK_COLOR);
    sprintf(s, "%d", (int)score);
    Gui_DrawFont_GBK24(50,0,POINT_COLOR,BACK_COLOR,s);
//  Gui_DrawFont_Num32(40,0,POINT_COLOR,BACK_COLOR,score);      //显示分数

    for(i=0;i<life;i++)
    {
        Gui_Circle(start_x+10*i,20,3,RED);
    }
}

```

飞机游戏结束页面包含结束标语与玩家的最终得分

```c
void draw_end(int8_t *score,int8_t *score_2, int8_t *score_3){
    Lcd_Clear(BACK_COLOR);

    Gui_DrawFont_GBK16(30,20,GRAY0,BACK_COLOR,"GAME OVER");
    Gui_DrawFont_GBK16(10,40,GRAY0,BACK_COLOR,"My SCORE:");
    sprintf(s, "%d", (int)*score);
    Gui_DrawFont_GBK24(90,40,POINT_COLOR,BACK_COLOR,s);
    //LCD_ShowxNum(100,170,*score_2,4,24,"0");
    if(*score_2 == 0 && *score_3 == 0){
        *score_2 = *score;
    }else if(*score_2 != 0 && *score_3 == 0){
        *score_3 = *score_2;
        *score_2 = *score;
        Gui_DrawFont_GBK16(10,40,GRAY0,BACK_COLOR,"P2 SCORE:");
//      LCD_ShowxNum(100,170,*score_2,4,24,0);
    }
    *score = 0;

    draw_my_plane(14,113);
    draw_my_plane(64,113);
    draw_my_plane(114,113);

}
```

4.单人游戏逻辑设计（展示部分代码 run.h）

包含以下重要函数

```
#ifndef __RUN_H
#define __RUN_H

#include "SC_Init.h"    // MCU initialization header file, including all firmware library header files

void generate_enemy_planes(int8_t level, int8_t enemy_planes[10][2]);   //生成敌人飞机
void switch_my_act(int8_t my_plane[2],int8_t my_buttles[20][2]);        //更新图像
void update_my_buttles(int8_t my_buttles[20][2]);                       //更新我的子弹
void update_enemy_buttles(int8_t enemy_buttles[20][2], int8_t level);   //更新敌机子弹
void update_enemy_planes(int8_t enemy_planes[10][2], int8_t level);     //更新敌机位置
//生成子弹
void generate_enemy_buttles(int8_t level, int8_t enemy_plane[10][2],  int8_t enemy_buttles[20][2]);
void generate_boss_buttles(int8_t boss[2],  int8_t enemy_buttles[20][2]);
//碰撞检测
void check_my_plane_to_enemy_plane_collide(int8_t *score,int8_t *life,  int8_t my_plane[2], int8_t enemy_planes[10][2]);
void check_my_buttles_to_enemy_plane_collide(int8_t *score, int8_t my_buttles[20][2], int8_t enemy_planes[10][2]);
void check_my_plane_to_enemy_buttles_collide(int8_t *life,  int8_t enemy_buttles[20][2], int8_t my_plane[2]);
void check_my_buttles_to_boss_collide(int8_t *score, int8_t *boss_life, int8_t my_buttles[20][2], int8_t boss[2]);

//以下函数仅用于boss关卡
//清除所有敌机
void clear_all_enemy_plane(int8_t my_plane[2],  int8_t enemy_plane[10][2], int8_t my_buttles[20][2], int8_t enemy_buttles[20][2]);
//移动boss
void move_boss(int8_t boss[2], int8_t *pace);
#endif
```

（1）飞机与 boss 数据生成与更新机制

函数 generate_enemy_planes(int8_tlevel,int8_t enemy_planes[10][2])，初步设定最大值为 10 个敌机,数组的第二维存储它们的 x,y 坐标,函数功能为随机生成敌机。

```
void generate_enemy_planes(int8_t level, int8_t enemy_planes[10][2]){
    int i = 0;

    // 随机生成地敌方飞机
    for(i=0; i<10; i++){
        if(enemy_planes[i][0] == 0){
            if(i < level*3){
                if(rand()%100 < level + 3){
                    enemy_planes[i][0] = rand()%100 + 15;  //随机飞机的横坐标
                    enemy_planes[i][1] = 30;               //随机飞机的纵坐标
                }
            }
            break;
        }
    }
}
```

函数 void switch_my_act（），根据按键的信息，执行对应的操作。如按压 R 按键，屏幕先清除之前的画面，在原来位置向右十个像素点位置生成新的飞机图像。

```c
void switch_my_act(int8_t my_plane[2],int8_t my_buttles[20][2]){
  int i=0;
  int key = KEY_Scan(1);
  //清除之前所画
  Lcd_Clearsome(my_plane[0]-14,my_plane[1]-14,my_plane[0]+14,my_plane[1]+14,BACK_COLOR);


  switch(key)
    {
      // right
      case R_PRES:
        if(my_plane[0]+10 < right_limit - 10){
        my_plane[0] = my_plane[0]+10;
      }

        break;

      //left
      case L_PRES:
        if(my_plane[0]-10 > left_limit + 10){
        my_plane[0] = my_plane[0]-10;
      }
        break;

      // attack
      case KEY_X_PRES:
        for(i=0; i<20; i++){
        if(my_buttles[i][0] == 0){
          my_buttles[i][0] = my_plane[0];
          my_buttles[i][1] = my_plane[1]-15;
          break;
        }
      }
```

函数 void update_my_buttles(int8_t my_buttles[20][2])，采用二维数组的方式保存 20 个子弹的 x，y 位置信息，遍历 20 个子弹，每次刷新先删除之前的图像，在初始位置向上三个像素的地方生成新的子弹图像，碰到边界则清除子弹。（敌方子弹、飞机的位置刷新均采用这种方式）

```c
void update_my_buttles(int8_t my_buttles[20][2]){
  int i=0;

  for(i=0; i<20; i++){
    if(my_buttles[i][0] != 0){
      //清除之前所画
      Lcd_Clearsome(my_buttles[i][0]-3,my_buttles[i][1]-3,my_buttles[i][0]+3,my_buttles[i][1]+3,BACK_COLOR);

      my_buttles[i][1] = my_buttles[i][1] - 3;

      //当到达边界时销毁
      if(my_buttles[i][1] < 42){
        my_buttles[i][0] = 0;
        my_buttles[i][1] = 0;
      }
    }
  }

}
```

（2）碰撞检测、积分以及殒命机制

碰撞（我方飞机与敌方子弹、敌方飞机与我防子弹）检测的实质是判断子弹中心与飞机中心的相对位置，如果两者在 x、y 方向的相对位置均小于 9 个像素点，则判断为一次碰撞，根据具体情况的得分或者殒命。

```
175 void check_my_buttles_to_enemy_plane_collide(int8_t *score,int8_t my_buttles[20][2], int8_t enemy_planes[10][2]){
176   int i=0;
177   int j=0;
178   int temp=0;
179 
180   for(i=0; i<10; i++){
181     if(enemy_planes[i][0] != 0){
182       for(j=0; j<20; j++){
183         //碰撞检测
184         if(abs(my_buttles[j][0] - enemy_planes[i][0]) <= 9){
185           if(abs(my_buttles[j][1] - enemy_planes[i][1]) <= 9){
186             Lcd_Clearsome(enemy_planes[i][0]-8,enemy_planes[i][1]-8,enemy_planes[i][0]+8,enemy_planes[i][1]+8,BACK_COLOR);
187 
188             my_buttles[j][0] = 0;
189             my_buttles[j][1] = 0;
190 
191             enemy_planes[i][0] = 0;
192             enemy_planes[i][1] = 0;
193 
194             temp = (*score);
195             (*score) = temp+1;
196 
197 //            LED1 = 0;
198               Delay_ms(100);
199 //              LED1 = 1;
200 
201             break;
202           }
203         }
204       }
```

5.双人联机逻辑设计

双人联机的思路是双方玩家共用一套程序，通过 UART 串口，玩家一方实时将按钮的按压信息以八位字符的形式通过串口传送给另一方，另一方一旦接受到信息，进入信息接受的中断程序，保存信息，并通过实时位置更新的程序将位置信息变化反馈在 tft 屏幕上，动态刷新还有积分和殒命的逻辑和之前的程序无异，唯一的区别时由于双方对战敌机的显示位置是镜像的，所以像素位置（敌机以及敌机子弹）动态刷新结果和敌方按钮扫描的结果的逻辑关系与单人模式下相反。包含以下函数

以下为处理单片机通过串口接收到数据得代码部分

```
void USCI2Interrupt()       interrupt 16
{
    /*USCI2_it write here begin*/
    /*USCI2_it write here*/
//  if(USCI2_GetFlagStatus(USCI2_UART_FLAG_RI)==SET) //得到接受区缓存数据并置入Recbuff里
//    {
//        Recbuff[i++]=USCI2_UART_ReceiveData8();
//        if(i>49) i=0;
//    }
    if(USCI2_GetFlagStatus(USCI2_UART_FLAG_RI)==SET) //得到接受区缓存数据并置入Recbuff里
    {
        abuf = USCI2_UART_ReceiveData8();
    }

    /*<Generated by EasyCodeCube begin>*/
    /*<Generated by EasyCodeCube end>*/
    /*USCI2Interrupt Flag Clear begin*/
    USCI2_ClearFlag(USCI2_UART_FLAG_RI);
    USCI2_ClearFlag(USCI2_UART_FLAG_TI);
    /*USCI2Interrupt Flag Clear end*/
}
```

下图函数是实现我方和敌方飞机位置根据摇杆和按钮信息动态更新的代码。

```
void switch_both_act(int8_t my_plane[2],int8_t my_buttles[10][2],int8_t his_plane[2],int8_t his_buttles[10][2]){
  int i=0;
  int key2 = KEY_Scan(1);
  //清除之前所画
  Lcd_Clearsome(my_plane[0]-16,my_plane[1]-16,my_plane[0]+16,my_plane[1]+16,BACK_COLOR);
  Lcd_Clearsome(his_plane[0]-16,his_plane[1]-16,his_plane[0]+16,his_plane[1]+16,BACK_COLOR);
  switch(key2)
    {
      // right
      case R_PRES:
        if(my_plane[0]+10 < right_limit2 - 15)
        {
          my_plane[0] = my_plane[0]+10;
        }
        UART0_SendData8(R_PRES);
        break;

      //left
      case L_PRES:
        if(my_plane[0]-10 > left_limit2 + 15)
        {
          my_plane[0] = my_plane[0]-10;
        }
        UART0_SendData8(L_PRES);
        break;

      // attack
      case KEY_X_PRES:
        for(i=0; i<10; i++)
        {
          if(my_buttles[i][0] == 0)
          {
            my_buttles[i][0] = my_plane[0];
            my_buttles[i][1] = my_plane[1]-15;
            break;
          }
        }
        UART0_SendData8(KEY_X_PRES);
        break;
      default:
        UART0_SendData8(8); //不是设定的任何一种按键
        break;
    }
  switch(rev_key)
    {
      // right
      case R_PRES:
        if(his_plane[0]+10 < right_limit2 - 15)
        {
          his_plane[0] = his_plane[0]+10;
        }
        break;

      //left
      case L_PRES:
        if(his_plane[0]-10 > left_limit2 + 15)
        {
          his_plane[0] = his_plane[0]-10;
        }
        break;

      // attack
      case KEY_X_PRES:
        for(i=0; i<10; i++)
        {
          if(his_buttles[i][0] == 0)
          {
            his_buttles[i][0] = his_plane[0];
            his_buttles[i][1] = his_plane[1]+15;
            break;
          }
        }
        break;
      default:
        break;
    }
}
```

6.时钟、温湿度显示以及闹钟设计

（1）时钟功能是通过 ESP8266 模块要获取网络时间，ESP8266 的工作模式设置为：STA。传输协议选择 TCP,配置为 Client（客户端）。工作的方式是 ESP8266 连接到一个可以连上 Internet 的路由器，通过这个路由器连接到 Internet，通过访问 http://api.k780.com 以获取网络时间，并通过定时器（由 TIMER4 中断触发）将时间递增进而实现时钟功能。（下图展示接受数据部分以及 esp8266 配置部分代码）

```
void USCI2_SendByte(uint8_t Byte)
{
  USCI2_UART_SendData8(Byte);
}

void USCI2_SendString(char* p)
{
  while(*p !='\0')
  {
    USCI2_SendByte(*p);
    p++;
    Delay_ms(10);
  }
}

char ESP8266_Cmd_Init() //esp8266 配置信息
{
  USCI2_SendString("AT+RST\r\n");  ////esp8266重启指令
  Delay_ms(1000);

  USCI2_SendString("ATE0\r\n");  //关闭回显
  Delay_ms(2000);

  USCI2_SendString("AT+CWMODE=1\r\n");  // 设置工作模式 1 station模式  2 ap模式  3 station+ap模式
  Delay_ms(1000);

  USCI2_SendString("AT+CWJAP=\"WWW\",\"wyn62399\"\r\n");//输入路由器的账号密码
  Delay_ms(5000);

  USCI2_SendString("AT+CIPSTART=\"TCP\",\"api.k780.com\",88\r\n");
  Delay_ms(1000);

  USCI2_SendString("AT+CIPMODE=1\r\n");
  Delay_ms(1000);

  USCI2_SendString("AT+CIPSEND\r\n");
  Delay_ms(1000);

//  USCI2_SendString("AT+RST\r\n");  ////esp8266重启指令
//  Delay_ms(1000);

  return 0;
}

char ESP_get_data(void)
{
  i=0;
  USCI2_SendString("GET http://api.k780.com:88/?app=life.time&appkey=10003&sign=b59bc3ef6191eb9f747dd4e83c99f2a4&format=json\r\n");
  Delay_ms(1000);
  return 0;
}

void Parse_BJ_Time(void)
{
  year=Recbuff[1];
  month=Recbuff[2];
  day=Recbuff[3];
  hour=Recbuff[4];
  minute=Recbuff[5];
  second=Recbuff[6];
}
```

（2）闹钟功能是利用 TIMER4 中断触发作为定时器，在时钟页面可以对闹钟时间进行设置，在任何模式中，一旦达到预定时间，变会自动弹出提示页面（TIME OUT!!!），弹窗在 5s 后自动消失，游戏继续进行。

```
227  void Timer4Interrupt()      interrupt 14
228 {
229      /*Timer4_it write here begin*/
230      /*Timer4_it write here*/
231      if(count<100)
232      {
233          count++;
234      }
235      else
236      {
237          count=0;
238          second+=1;
239          if(second>=60)
240          {
241              second=0;
242              minute++;
243          }
244          if(minute>=60)
245          {
246              minute=0;
247              hour++;
248          }
249          if(hour>=24)
250          {
251              hour=0;
252              day++;
253          }
```

判断是否满足当前时间与设定时间相同，如果满足，屏幕上即显示弹窗“TIME OUT”

```
62  void judge_clock(void)
63 {
64      if(set_hour==hour && set_minute==minute && set_second==second)
65      {
66          Lcd_Clear(0xFFFF);
67          Gui_DrawFont_GBK24(20,50,GRAY1,BACK_COLOR,"TIME OUT!!!");
68          Delay_ms(5000);
69          Lcd_Clear(BACK_COLOR);
70      }
71 }
```

（3）温湿度显示功能是通过 SHT30 温湿度传感器模块获取温湿度数据，通过 I2C 协议实现与单片机得相互通信，并在 TFT 屏幕上实时显示。（下图展示封装的 SHT30 模块相关函数，包括配置、获取以及解析数据，显示数据）

```c
#ifndef SHT30_H
#define SHT30_H

#include "SC_Init.h"
#include "SC_it.h"

#define SHT30_ADDR      0x44

typedef union
{
    float fdata;
    unsigned char cdata[4];
}float_data;//定义联合体存储float数据，float类型的存储符合IEEE标准，可用于传输数据

void SHT30_Init(void);
void SHT30_General_RESET(void);
void SHT30_Periodic_Config(void);

void SHT30_Read(void);
void SHT30_SendBytes(u16 cmd,u8 stop);
void SHT30_Single_Shot(u8 *buffer);
void SHT30_Periodic(u8 *buffer);

void SHT30_Show(void);

 #endif
```

7.背景音乐设计

通过 PWM 输出捕获的方式控制无源蜂鸣器发声，由定时器控制音乐的节拍、PWM 控制音调，所以只需要动态修改装载值即可实现 PWM 输出频率，进而控制音调变化，定时器的中断则负责控制节拍。由于游戏机的属性，我们选择了类似两只老虎得音乐作为 BGM。

C调音符与频率对照表

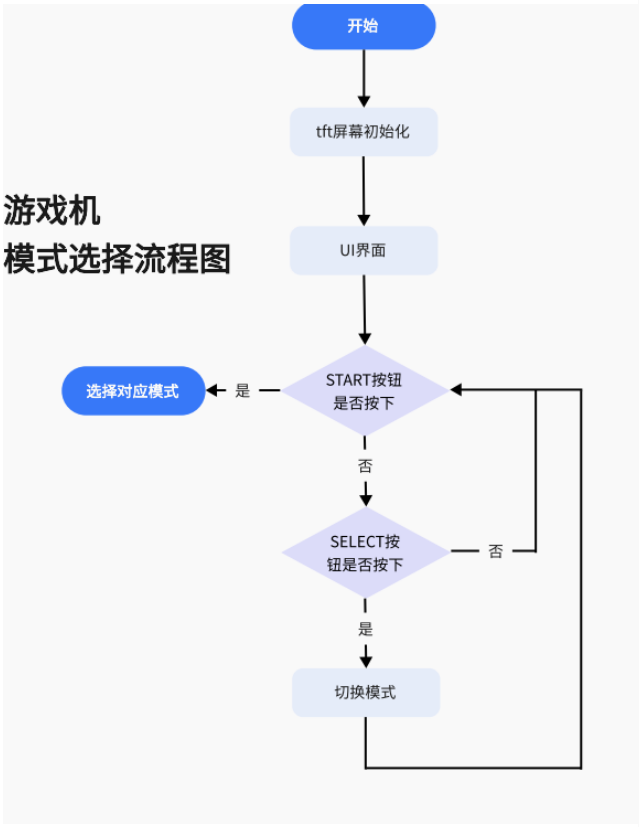
音符	频率/Hz	音符	频率/Hz	音符	频率/Hz
低音1	262	中音1	523	高音1	1046
低音1#	277	中音1#	554	高音1#	1109
低音2	294	中音2	587	高音2	1175
低音2#	311	中音2#	622	高音2#	1245
低音3	330	中音3	659	高音3	1318
低音4	349	中音4	698	高音4	1397
低音4#	370	中音4#	740	高音4#	1480
低音5	392	中音5	784	高音5	1568
低音5#	415	中音5#	831	高音5#	1661
低音6	440	中音6	880	高音6	1760
低音6#	466	中音6#	932	高音6#	1865
低音7	494	中音7	988	高音7	1976

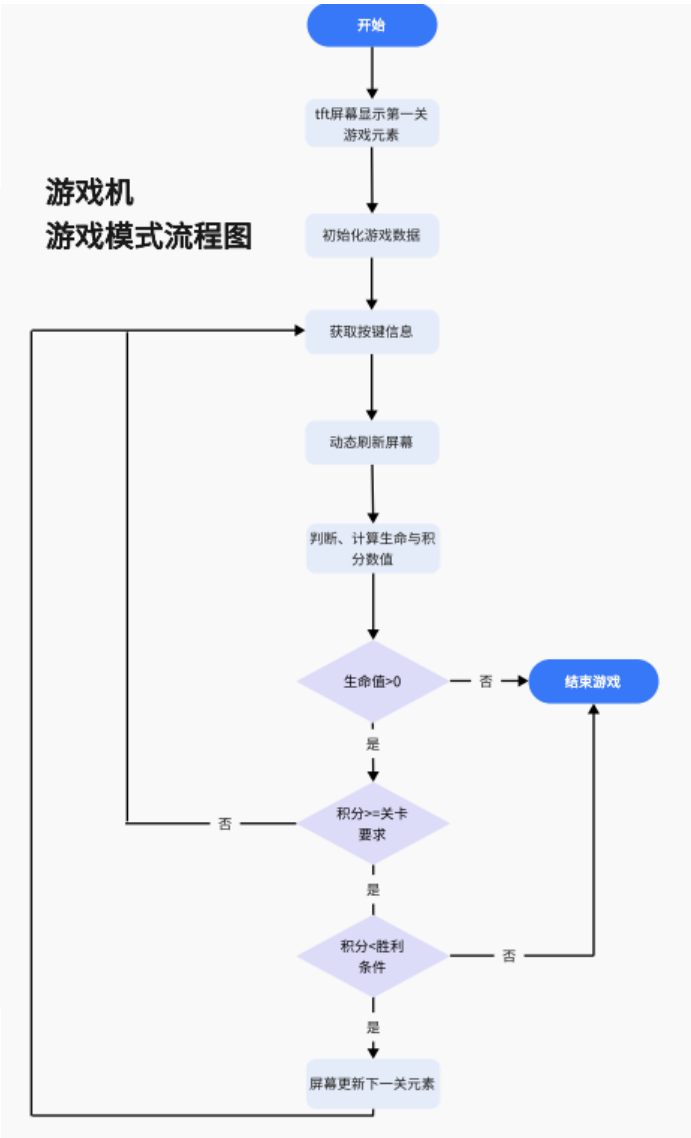
```c
#include "music.h"
#include "stdlib.h"

#define P_1 1
#define P_2 2
#define P_4_3 0.75
#define P_4_1 0.25

uint16_t C_FREQ[]={0,262,294,330,349,392,440,494,523,587,659,698,784,880,988,1046,1175,1318,1397,1568,1760,1976};

/*--------------------两只老虎--------------------*/
/*频率*/
uint8_t music_two_tiger_f[]={
    1,2,3,1,
    1,2,3,1,
    3,4,5,
    3,4,5,
    5,6,5,6,3,1,
    5,6,5,4,3,1,
    1,5,1,
    1,5,1,
};

/*1/4拍长*/
u16 music_two_tiger_t_echo=250;

/*长度*/
u16 music_two_tiger_len=(sizeof(music_two_tiger_f)/sizeof(uint8_t));

/*拍长*/
uint8_t music_two_tiger_t[]={
    P_1,P_1,P_1,P_1,
    P_1,P_1,P_1,P_1,
    P_1,P_1,P_2,
    P_1,P_1,P_2,
    P_4_3,P_4_1,P_4_3,P_4_1,P_1,P_1,
    P_4_3,P_4_1,P_4_3,P_4_1,P_1,P_1,
    P_1,P_1,P_2,
    P_1,P_1,P_2,
};

u8 B0[]="STOP";
u8 B1[]="two tigers";

struct MUSIC_T music_t;

void music_set(u8 n)
{
  switch(n){
    case 0:{
          music_t.f=NULL;
          music_t.t=NULL;
          music_t.len=0;
          music_t.t_each=0;
          music_t.name=B0;
          };break;
    case 1:{
          music_t.f=music_two_tiger_f;
          music_t.t=music_two_tiger_t;
          music_t.len=music_two_tiger_len;
          music_t.t_each=music_two_tiger_t_echo;
          music_t.name=B1;
          }; break;
  }
}
```

将 music_set()函数放置于 Timer_0 的定时器中断中完成 PWM 输出占空比的动态更新。

```c
void Timer0Interrupt()      interrupt 1
{
    /*TIM0_it write here begin*/
    my_buzzer_play();
    TIM0_Mode1SetReloadCounter(33536);
    /*TIM0_it write here*/
    /*<Generated by EasyCodeCube begin>*/
    /*<UserCodeStart>*//*<SinOne-Tag><6>*/
    //Timer0Interrupt
    {
        /*<UserCodeStart>*//*<SinOne-Tag><7>*/
        /*****User program*****/
        /*<UserCodeEnd>*//*<SinOne-Tag><7>*/
        /*<Begin-Inserted by EasyCodeCube for Condition>*/
    }
    /*<UserCodeEnd>*//*<SinOne-Tag><6>*/
    /*<Generated by EasyCodeCube end>*/
    /*Timer0Interrupt Flag Clear begin*/
    /*Timer0Interrupt Flag Clear end*/
}
```

四、流程框图展示

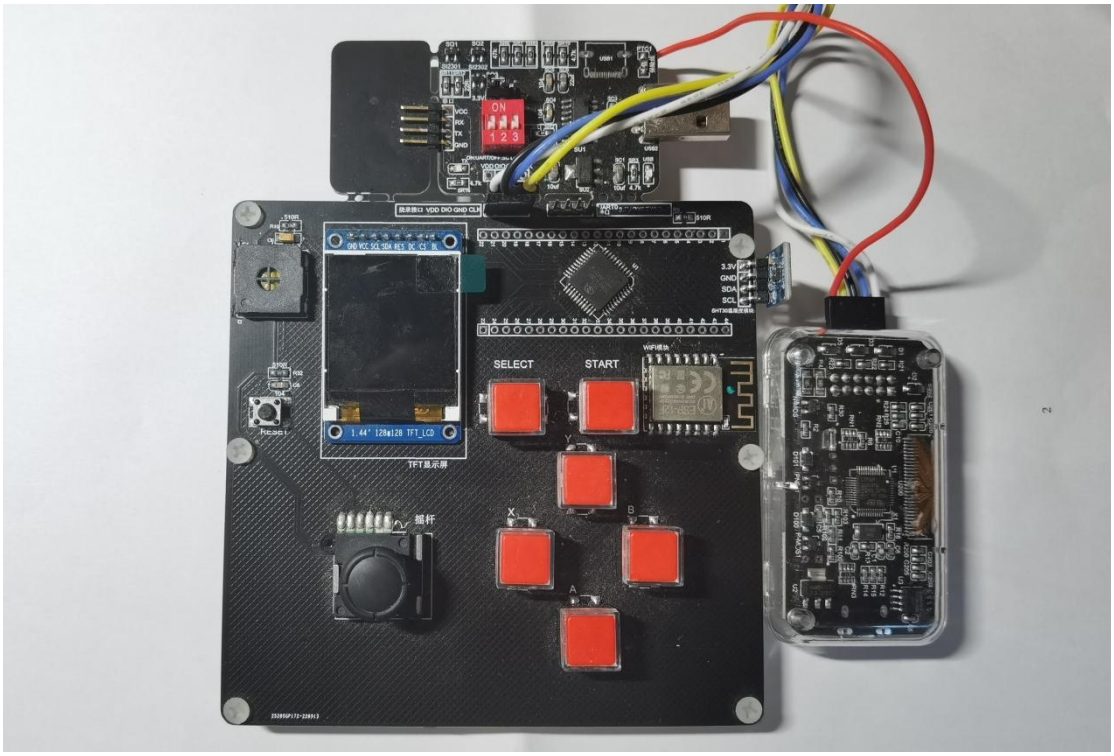

游戏机
模式选择流程图
开始
tft屏幕初始化
UI界面
START按钮
是否按下
是
选择对应模式
否
SELECT按
钮是否按下
否
是
切换模式


游戏机
游戏模式流程图
开始
tft屏幕显示第一关
游戏元素
初始化游戏数据
获取按键信息
动态刷新屏幕
判断、计算生命与积
分数值
生命值>0
否
结束游戏
是
积分>=关卡
要求
否
是
积分<胜利
条件
否
是
屏幕更新下一关元素

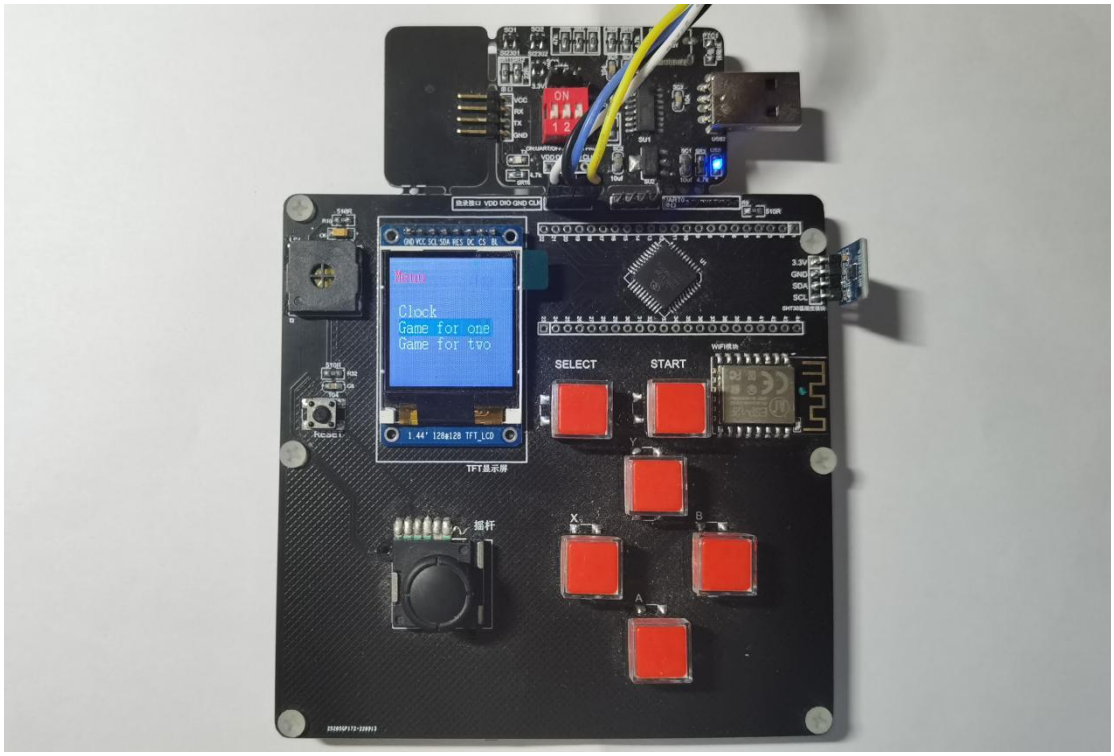
五、实物展示

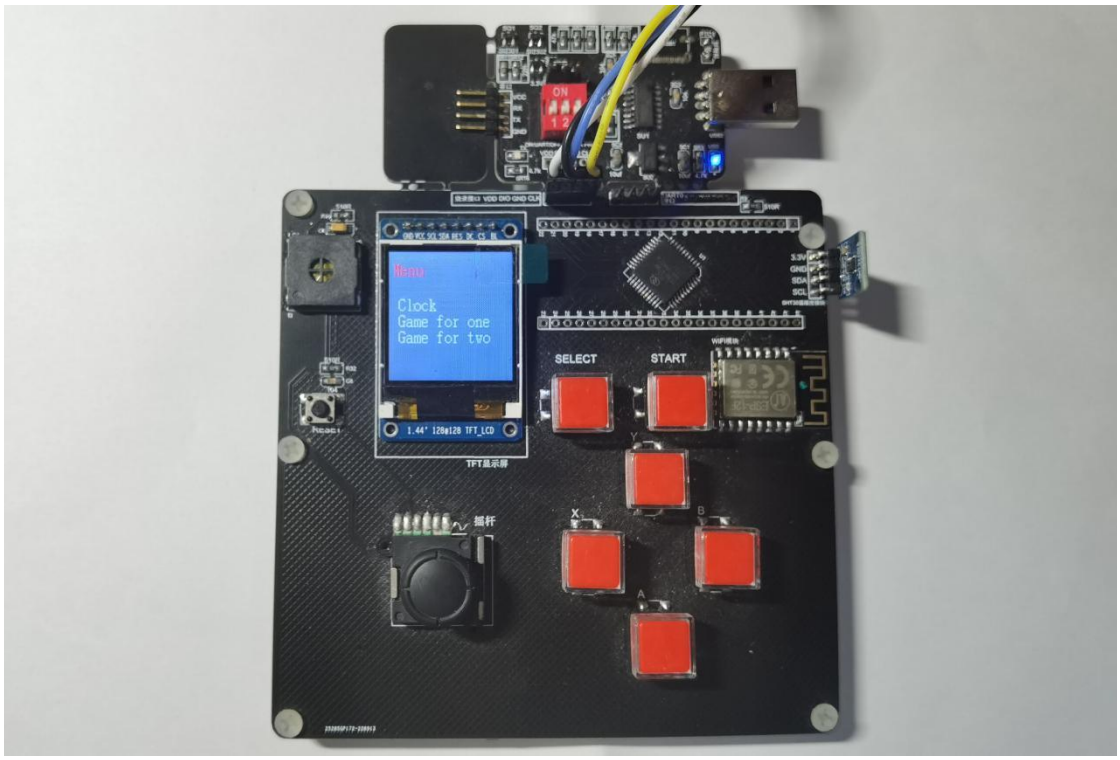
下图展示了游戏机的 UI 界面，飞机游戏的两个难度关卡，以及游戏机整体外观

1.游戏机整体硬件展示

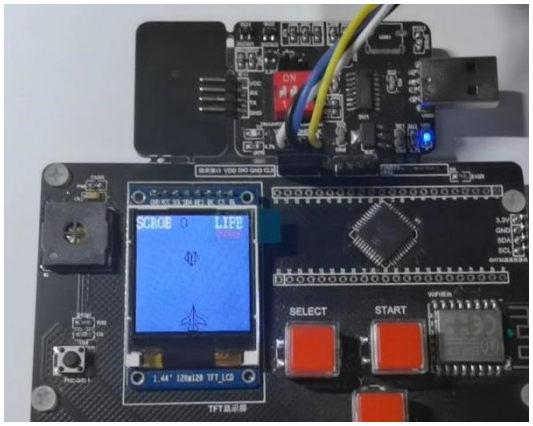


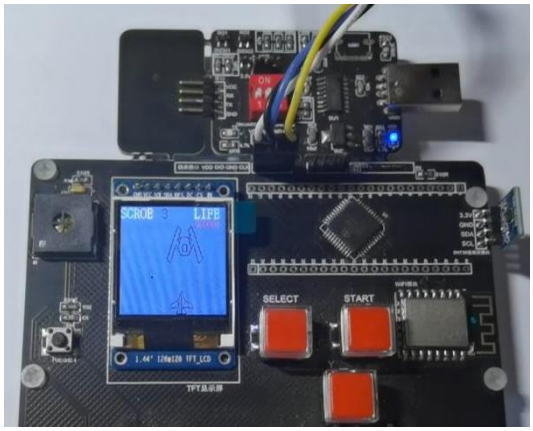
2.游戏机 UI 界面展示

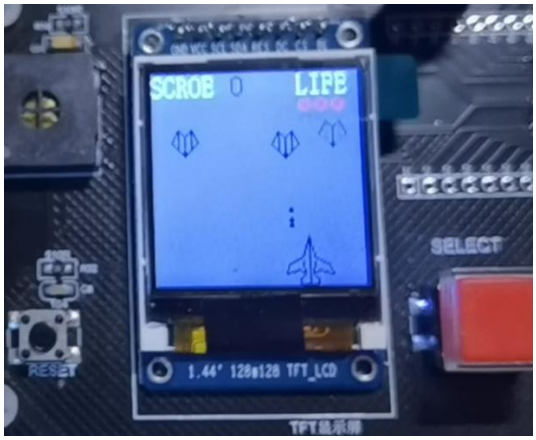




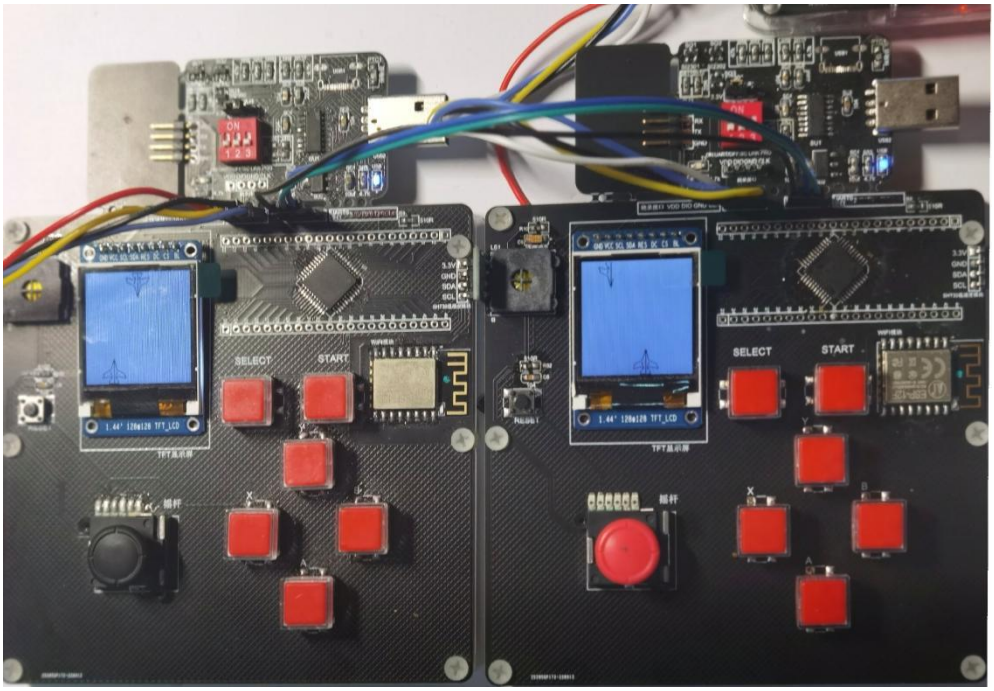
3.游戏机单人游戏界面展示（包含第一、第二关卡，普通敌机及 BOSS）



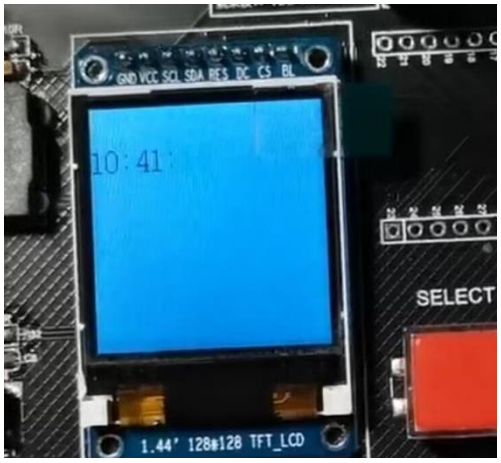


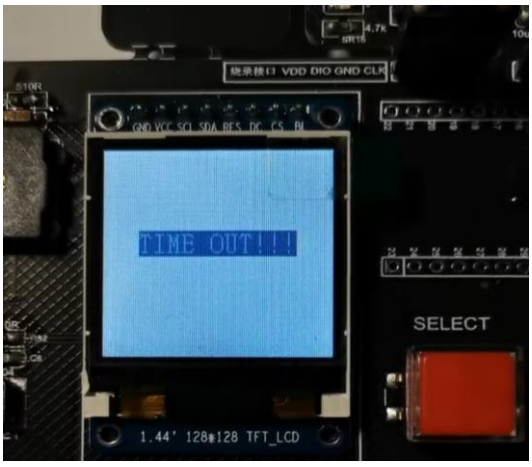


4.游戏机双人联机游戏画面



5.闹钟模式展示





六 设计总结

关于这次赛元杯比赛，我们花费了比较多的心思，既是对课程理论内容的一次复习和巩固，还让我们丰富了更多与该专业相关的其他知识，比如软件应用等，在摸索中学习，在摸索中成长，在学习的过程中带着问题去学我发现效率很高，这是做这次设计的又一收获。

在真正设计之前我们做了相当丰富的准备，首先巩固一下课程理论，再一遍熟悉课程知识的构架，然后结合加以理论分析、总结，有了一个清晰的思路和一个完整的的软件流程图之后才着手设计。在设计程序时，也不能妄想一次就将整个程序设计好，反复修改、不断改进是程序设计的必经之路；养成注释程序的好习惯也是非常必要的，一个程序的完美与否不仅仅是实现功能，也需要一目了然，为资料的保存和交流提供了方便；在设计过程中遇到问题是很正常，应该将每次遇到的问题记录下来，并分析整理，提高自己的编程能力。

本次比赛我们也有不足之处，由于疫情的缘故比赛延期，假期里四个组员没有办法及时高效沟通，游戏机的设计存在不足，其一是 esp8266 配网获取时间的部分，返回的数据总是错误，由于技术和时间原因，暂时没有解决。其二是温湿度传感器获取温湿度数据的部分，同样是难以获得正确的数据，暂待解决。在未来，我们也会聚焦实践中出现的问题，加强理论学习，加强实践。