1 、课程设计要求

1.设计要求

设计一个简单的电子系统，使用 STM32f103 系列单片机，主要包括以下模块:

- 4 个 LED 灯
- 4 个独立按键 K1-K4
- 1 路模拟电压输入
- 串口转 USB 接口
- IIC 接口陀螺仪传感器
- IIC 接口 OLED 显示屏
- WIFI 通信模块
- 5V USB 供电

2.基本功能

基于 STM32f103 单片机实现如下功能:

- **功能 1：**系统上电启动，4 个 LED 灯闪烁 1s,OLED 屏显示课程名称、学号、姓名，保持 1s 后进入主界面，显示系统名称和功能菜单。通过 K1、K2 上下选择功能，K3 确定进入功能界面。在所有功能界面，默认 K4 返回主界面。

  附加：显示学号、姓名后，屏幕正中显示个人头像，个人头像需与真人相符，等待任意键后再进入主界面。
- **功能 2：**系统测试界面，在 OLED 屏显示 4 个 LED 灯状态、4 个按键状态、AD 采样数据、陀螺仪传感器原始数据。单页显示不下时通过 K1、K2 上下翻页。K3 按键可以开启/暂停一个简单的流水灯。

  附加：LED 灯状态、按键状态、AD 采样数据及陀螺仪传感器数据均使用图形绘制结合文字显示。
- **功能 3：**陀螺仪姿态解算界面，OLED 显示内容为陀螺仪当前 6 轴原始数据和解算出的姿态角数据，单页显示不下时使用 K1、K2 上下翻页。

  附加：使用 K3 按键进入陀螺仪控制状态，屏幕显示一个 8x8 大小的图形，该图形受陀螺仪数据控制，可实现上下左右移动。按任意键返回。
- **功能 4：**串口通信测试界面，在该界面，串口 1 将接收数据原样回送出去，OLED 屏上显示串口 1 当前接收数据（最后一行字符串且小于

40 字节）和累计接收数据字节数。

附加：K1 按键切换字符串/HEX 格式显示、显示数据不受屏幕大小限制，K2 按键下翻页，到底则返回第 1 页、K3 按键清空接收数据和累计数据。

- **功能 5：**WiFi 通信测试界面，进入该界面后检测 WiFi 模块是否正常 ，如正常则配置 WiFi 模块为 STA+AP 工作模式，OLED 屏上显示 AP 的 SSID 名称，WiFi 模块异常时 OLED 屏显示错误信息。配置完成后，手机或笔 记本连接 WiFi 模块，K1 按键设置模块连接手机或笔记本上的 TCP 服务器，并进入透传模式，屏幕显示接收到的最后一行字符串。K2 按键向服务器发送一个字符串、K3 按键退出透传模式。

  附加：不更改短按键功能，通过长按键和组合键检测，实现连接服务器 IP 地址和端口可设置功能。
- **功能 6：**设计 PC 端上位机软件，能接收单片机串口上传数据，并将数据帧中的有效数据提取出来显示在程序窗口中。显示内容包括：4 个 LED 灯状态、4 个按键状态、AD 采样数据或采样电压值、陀螺仪 6 轴原始数据及解算姿态角度，通信协议参考最后串口协议。

  附加： 除了接收串口数据，还能接收 TCP 客户端上传数据。

  上位机能根据 AD 值和陀螺仪姿态角控制 OLED 灯实现不同的报警效果。
- **拓展：[拓展 5]：**每隔一段时间 OLED 屏幕最上方随机生成一些*号并向下运动，屏幕最下方绘制一个倒 V 型小飞机，能通过陀螺仪来移动飞机躲避*号，撞到*号游戏结束，统计得分。

2 、系统方案设计（框图、原理图）

系统使用 STM32Cube 软件配置好引脚和所有具体需要的资源，FreeRTOS 是一个迷你的实时操作系统内核。作为一个轻量级的操作系统，功能包括：任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等，可基本满足较小系统的需要。

FreeRTOS 采用 C 语言编写，因此其结构简洁，可读性很强，因此整个系统可用 FreeRTOS 开启三个任务，实现数据采集、上传、按键动作和界面显示等综合功能，主任务负责 MPU6050 数据采集和数据上传、 按键任务负责按键扫描和按键动作处理、显示任务负责 OLED 屏幕显示刷新。

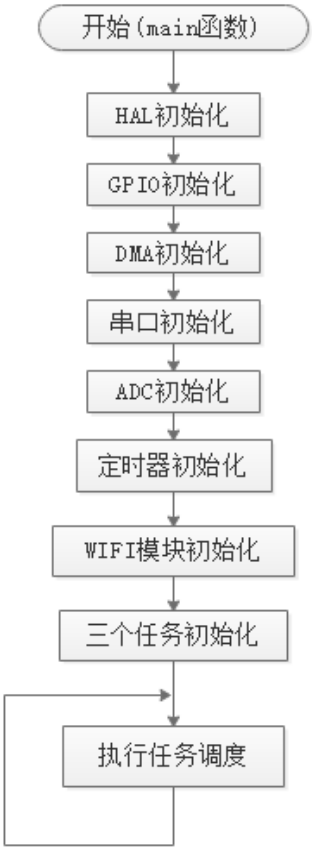
本系统的设计思路就是利用 FreeRTOS 的三个任务：**StartDefaultTask**、**StartKeyTask** 和 **StartDispTask** 管理整个系统。其中 **StartDefaultTask** 管理串口

1 和 WIFI 使用的串口 2。**StartKeyTask** 管理整个系统所有功能的按键操作，以保证能够将各个功能统一起来。**StartDispTask** 管理整个系统所有的 OLED 屏幕显示，包括各个功能的跳转和功能内部的按键操作。

整个系统的主框架为：main 函数为系统入口。

1. HAL 初始化；
2. 系统时钟初始化；
3. GPIO 初始化；
4. DMA 初始化；
5. 串口 1、串口 2 初始化；
6. 定时器 TIM3 初始化；
7. ADC 初始化；
8. WIFI 模块初始化；
9. 三个 FreeRTOS 任务初始化；
10. osKernelStart()函数运行[任务开始]；
11. while(1)死循环[里面无内容]；

则整个主系统的框图为：



相关核心代码如下：

Main（）函数

```
int main(void){
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART1_UART_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();
  MX_ADC1_Init();
/* USER CODE BEGIN 2 */
      rxit_ok = HAL_UART_Receive_IT(&huart1, pBuf, 1);   // 串口 1 开启第一次中断，每次接收 1 字节
      rxit_ok2 = HAL_UART_Receive_IT(&huart2, pBuf2, 1); // 串口 2 开启第一次中断，每次接收 1 字节
      printf("\nTengJiaLu\n16073212\n\n");              // 向串口 1 发送一行字符串
     Esp8266Err = InitESP8266();       //WIFI 初始化
  /* USER CODE END 2 */
  osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128);
  defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
  /* definition and creation of KeyTask */
  osThreadDef(KeyTask, StartKeyTask, osPriorityIdle, 0, 128);
  KeyTaskHandle = osThreadCreate(osThread(KeyTask), NULL);
  /* definition and creation of DispTask */
  osThreadDef(DispTask, StartDispTask, osPriorityIdle, 0, 128);
  DispTaskHandle = osThreadCreate(osThread(DispTask), NULL);
  osKernelStart();
      while(1);
}
```

3 、软件设计（软件功能框图、各模块流程图）

整个系统实现了 6 个基本功能和 1 个拓展功能。

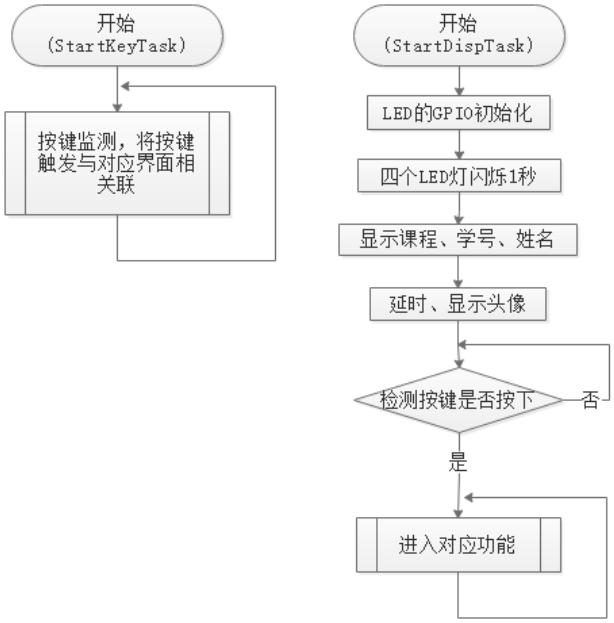
功能一：启动界面及按键控制菜单功能

此功能主要实现了开机启动界面和菜单界面。根据要求，系统上电启动，4 个 LED 灯闪烁 1s,OLED 屏显示课程名称、学号、姓名，保持 1s 后进入主界面，显示系统名称和功能菜单。通过 K1、K2 上下选择功能，K3 确定进入功能界面。在所有功能界面，默认 K4 返回主界面。**屏幕正中显示个人头像，个人头像需与真人相符，等待任意键后再进入主界面。**

因此在 StartDispTask()任务里首先执行开机启动时的界面，显示完成后等待按键按下，之后使用 while 循环根据 当前功能记录变量 显示对应的功能界面即

可。

程序流程图如下：



主要代码如下：

StartDispTask（）函数

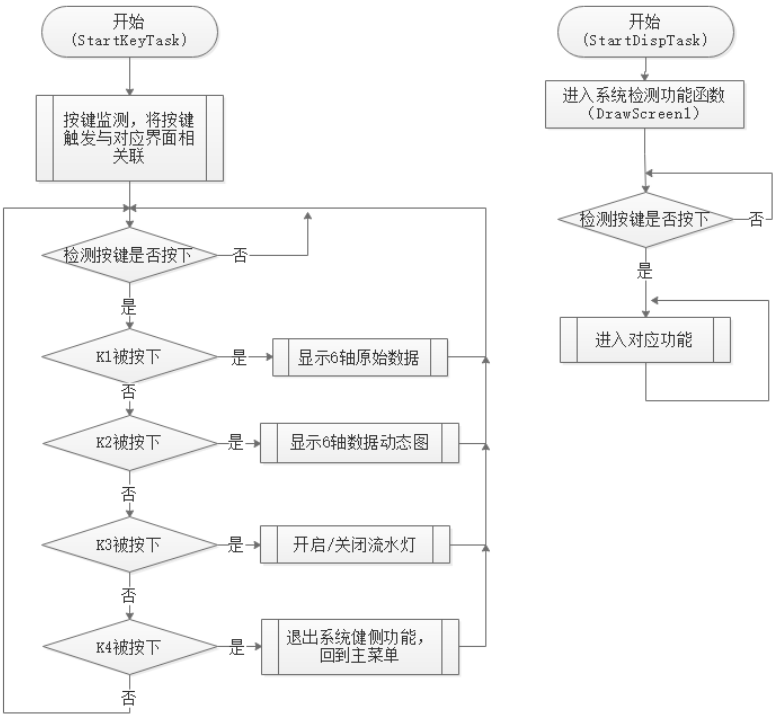
```
void StartDispTask(void const * argument){
    LED_GPIO_Init();
    for(int i=0;i<5;i++){   //LED 闪烁
        SetLEDS(0x0f);
        osDelay(160);
        SetLEDS(0x00);
        osDelay(160);
    }
    SetLEDS(0x00);
    GUI_Init();//显示课程姓名学号
    GUI_Clear();
    GUI_SetFont(&GUI_FontHZ_NewSimSun_16);
    GUI_DispStringAt("电子系统设计", 16, 0);
    GUI_SetFont(&GUI_FontHZ_KaiTi_20);
    GUI_DispStringAt("滕佳禄", 34, 20);        //128-48=80
    GUI_DispStringAt("16073212", 24, 40);    //128-64=64
    GUI_Update();
    osDelay(2000);          //延时 2 秒
    GUI_Clear();
   GUI_DrawBitmap(&bmbptjl, 22, 0);   //显示头像
   GUI_Update();
    while(!ScanKey());          //等待按键按下
    initf=0;
```

功能二：实现系统测试功能

系统测试界面，在 OLED 屏显示 4 个 LED 灯状态、4 个按键状态、AD 采样数据、陀螺仪传感器原始数据。单页显示不下时通过 K1、K2 上下翻页。K3 按键可以开启/暂停一个简单的流水灯。**LED 灯状态、按键状态、AD 采样数据及陀螺仪传感器数据均使用图形绘制结合文字显示。**

首先显示功能名称，在屏幕左侧显示系统测试。接着在中间和右边显示四个 LED 灯和按键状态，并且用图形表示出当前状态。下方用长条动态表示电压采样数据的，按键 K1 按下，显示陀螺仪的 6 轴原始数据。按键 K2 按下，显示 6 轴的动态图。按键 K3 开启/关闭流水灯效果。按键 K4 则退出系统测试。

程序流程图如下：



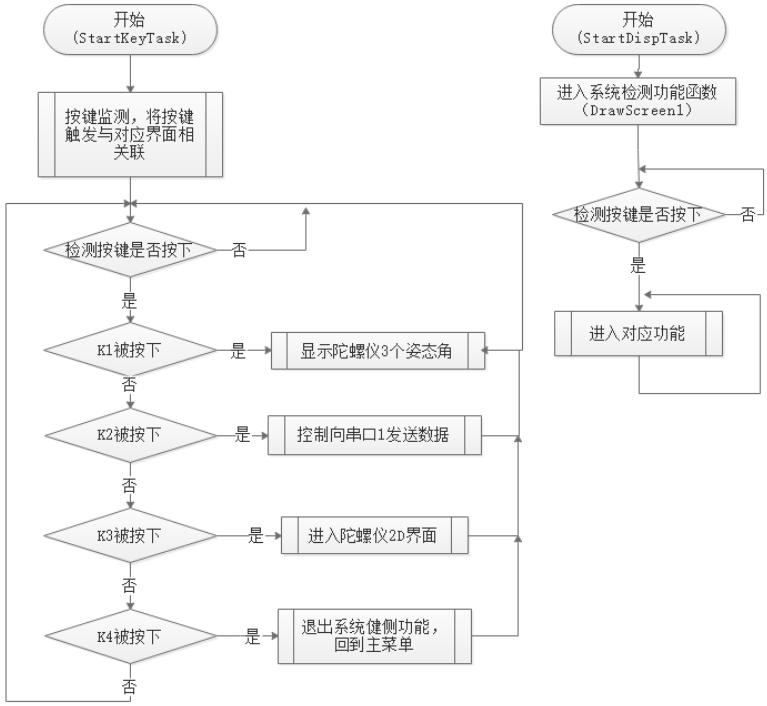
功能三：陀螺仪姿态解算

OLED 显示内容为陀螺仪当前 6 轴原始数据和解算出的姿态角数据，单页显示不下时使用 K1、K2 上下翻页。使用 K3 按键进入陀螺仪控制状态，屏幕显示一个 8x8 大小的图形，该图形受陀螺仪数据控制，可实现上下左右移动。按任意键返回。

首先显示功能名称，在屏幕左侧显示姿态解算。主要做法为利用四元数进行

姿态解算，显示在界面上，K1 按下显示解算出的三个姿态角，K2 按下控制发送到串口的开关，K3 按下进入陀螺仪的 2d 控制。K4 按下则退出姿态解算界面。

程序流程图如下：



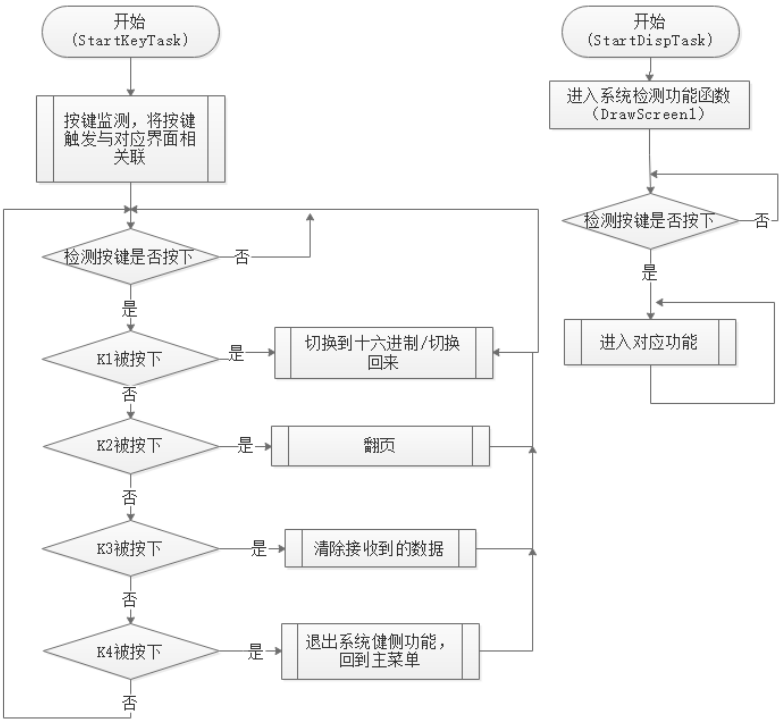
功能四：串口通信

串口通信测试界面，在该界面，串口 1 将接收数据原样回送出去，OLED 屏上显示串口 1 当前接收数据（最后一行字符串且小于 40 字节）和累计接收数据字节数。K1 按键切换字符串/HEX 格式显示、显示数据不受屏幕大小限制，K2 按键下翻页，到底则返回第 1 页、K3 按键清空接收数据和累计数据。

首先显示功能名称，在屏幕左侧显示串口通信。主要做法为利用串口 1 与电脑的串口助手进行通信。附加功能的实现，利用不同按键进入不同的功能，实现翻页时利用一个记录当前显示位置的变量来记录字符串的显示下标，每次翻页就将下标向后移动，直到下标移动到字符串末尾，显示时只要从下标记录的位置开始显示即可。则返回字符串头部，以此实现循环的功能。对于十六进制的显示，将每个字符转换为十六进制显示即可。

对于统计接收的总字数与清除数据功能只要在接收时对统计接收数据的变量进行累加，清除时对其置零即可。

程序流程图如下：



相关代码如下：

DrawScreen2（）函数[部分略]

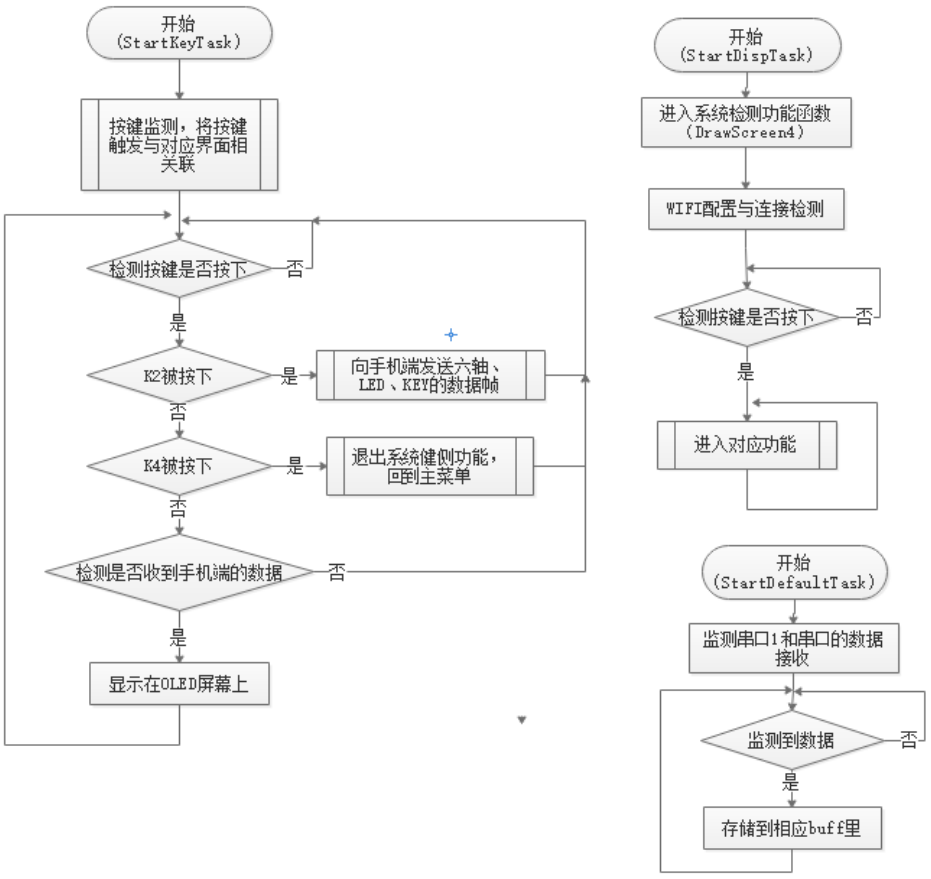
```
void DrawScreen3(void){
    px=24;py=16;
    if(rxfk1){    //切换显示[十六进制]
        if(yj>=strlen(rx1_show))
            yj=0;
        s=rx1_show+yj;      //换页核心代码
        ty=yj;
        while(*s!='\0'&&(ty-yi)<15){//一页显示 15 个字符的 HEX
            if(px>=116){      //换行
                px=24;
                py+=16;
            }
            sprintf(c,"%0x",*s);
            GUI_DispStringAt(c, px,py);
            px+=20;s++;ty++;
        }
    }
    else{    //普通显示字符
        if(yi>=strlen(rx1_show))
            yi=0;
        s=rx1_show+yi;
        tx=yi;
        while(*s!='\0'&&(tx-yi)<39){      //一页显示 39 个字符
```

```
                if(px>=128){        //换行
                    px=24;
                    py+=16;
                }
                c[0]=*s;c[1]='\0';
                GUI_DispStringAt(c, px,py);
                px+=8;s++;tx++;
            }
        }
```

功能五：WIFI 通信

WiFi 通信测试界面，进入该界面后检测 WiFi 模块是否正常 ，如正常则配置 WiFi 模块为 STA+AP 工作模式，OLED 屏上显示 AP 的 SSID 名称，WiFi 模块异常时 OLED 屏显示错误信息。配置完成后，K1 按键设置模块连接手机或笔记本上的 TCP 服务器，并进入透传模式，屏幕显示接收到的最后一行字符串。K2 按键向服务器发送一个字符串、K3 按键退出透传模式。

此部分首先初始化 WIFI 模块，配置相应的模式，与手机上的网络助手通信。程序流程图如下：



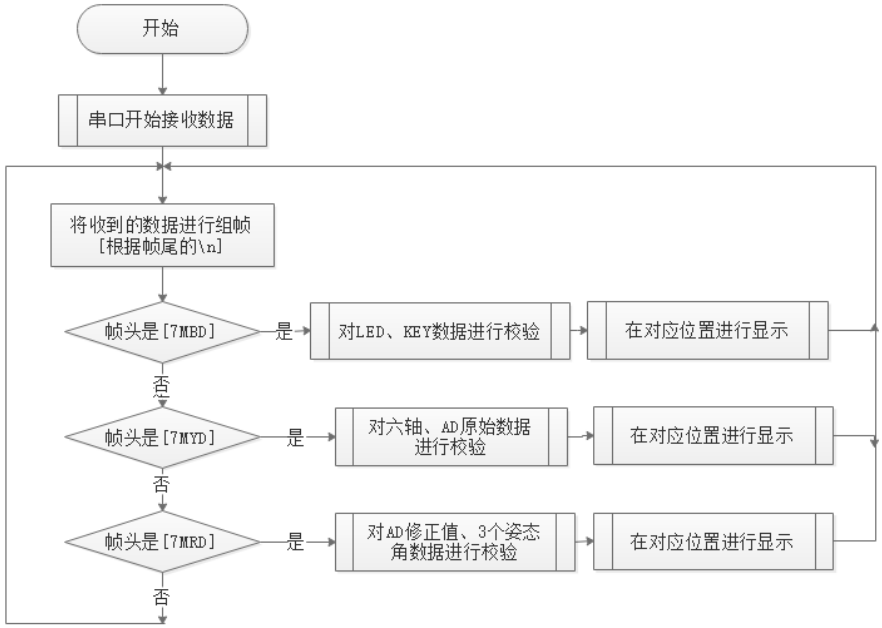
功能六：串口助手上位机

设计 PC 端上位机软件，能接收单片机串口上传数据，并将数据帧中的有效数据提取出来显示在程序窗口中。显示内容包括：4 个 LED 灯状态、4 个按键状态、AD 采样数据或采样电压值、陀螺仪 6 轴原始数据及解算姿态角度，通信协议参考最后串口协议。

使用 Qt 进行上位机开发。在单片机中将六轴数据、AD 采样、KEY、LED 根据给定的通信协议进行编码，上位机接收到后，进行解包，将对应的数据放在相应的控件上。

此功能与功能二的 K2 键发送数据到串口的功能整合在了一起，共用一个模块，因此流程图不在赘述。不过值得注意的是校验码的生成，需要进行处理。

针对上位机，对应的流程图如下：



相关代码如下：

单片机程序[只列出核心代码]

```
//对 buf 进行校验，末尾加上【校验和】后两个字节的十六进制和'\n'。
void AddJiaoYan(char *buf){
    char tmp[20],*p=buf;
    uint16_t tcnt=0,tm;
    while(*p!='\0'){
        tcnt+=*p;
        ++p;
    }
    tm=tcnt%16;
    tcnt>>=4;
```

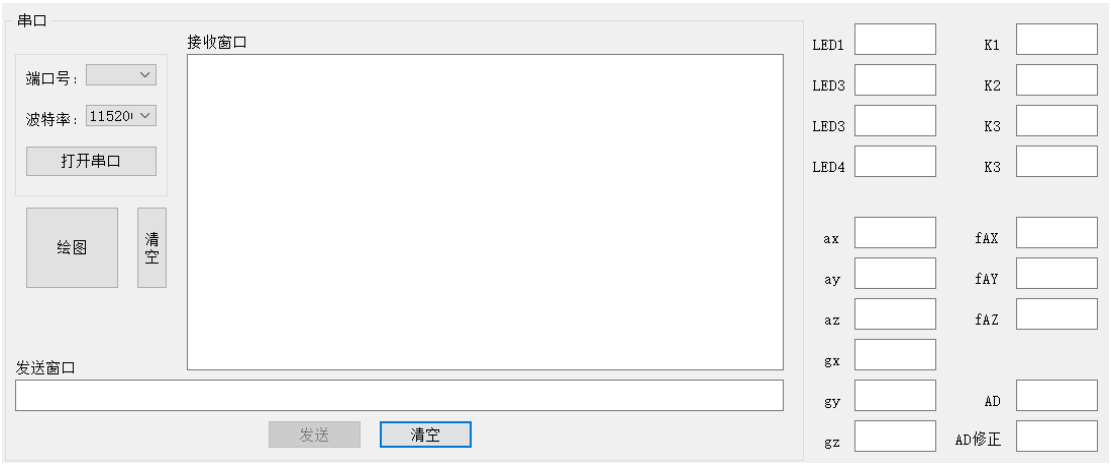
```
        sprintf(tmp,"%0x%0x\n",tcnt%16,tm);
        strcat(buf,tmp);
}
```

上位机程序[Qt,只列出核心代码]

```
    buf = serial->readAll();

    if(!buf.isEmpty())
    {
        data+=QString(buf);
        int len=data.indexOf("\n");
        while(len>=0){
            Data=data.left(len);
            data=data.mid(len+1);
            len=data.indexOf("\n");
            ui->textRecv->append(Data);
        if(Data.startsWith("7MBD"))//流水灯
ui->textEdit_12->clear();ui->textEdit_12->append(Data.mid(4,1)[0]=='1'?"●":"○");
ui->textEdit_13->clear();ui->textEdit_13->append(Data.mid(5,1)[0]=='1'?"●":"○");
     //下略 }
        else if(Data.startsWith("7MYD")){//略}
        else if(Data.startsWith("7MRD")){//略}
```

上位机界面如下：



拓展五：飞机游戏

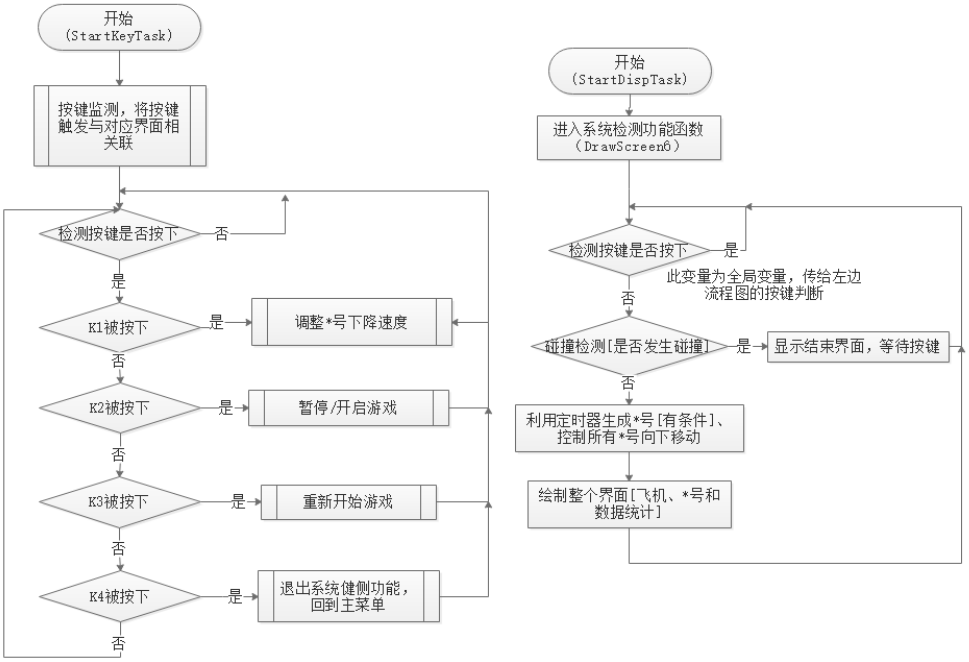
每隔一段时间 OLED 屏幕最上方随机生成一些*号并向下运动，屏幕最下方绘制一个倒 V 型小飞机，能通过陀螺仪来移动飞机躲避*号，撞到*号游戏结束，统计得分。

此游戏的关键点在于*的随机生成和碰撞检测。由于*的个数不定，因此采用二维数组对*的坐标点和状态进行存储，这样就可以动态刷新所有*的位置。

利用滴答计时器每隔一定时间采集一次 6 轴传感器中变化幅度较大的 ax 值，然后乘以一个质小数，再映射到屏幕的 0-111 的位置，这样便随机生成了一个点，[放弃使用库函数 rand()]例如这样：randx[randi]=(int)(ax*1.7)%101;由于定时器还控制*号的下降，这样生成的点太密集，因此对生成的点进行阈值判断，大于 92 的值再进行生成，这样相当于只有 8%的机会会生成这个点，这样判断会将点限制在 92-101 的屏幕范围内，因此我们**不取这次生成的点，判断这次的点大于 92 后，选取下一个生成的点，**这样完美的解决了这个问题。

本功能不仅实现了基本的要求，还有分数统计、改变速度、暂停游戏、重新开始等多个功能，游戏体验得到了极大的提升。通过 K1 按键改变速度，速度到达 7 级以后非常快，需要考验一定的水平。K2 键暂和继续游戏，K3 键重新开始游戏，K4 键退出游戏。

程序的流程图如下：



相关代码如下：

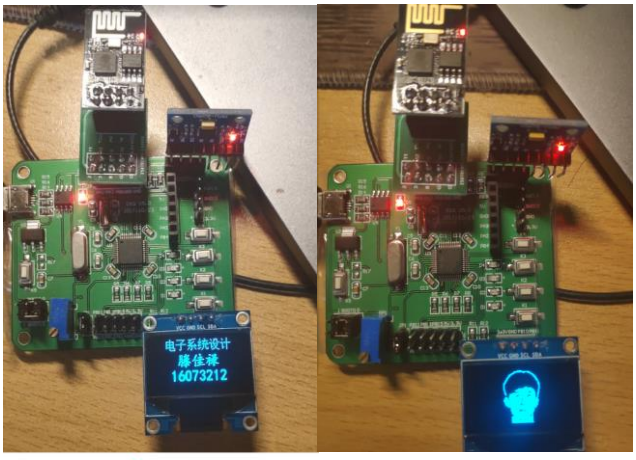
DrawScreen6（）函数[只展示核心代码]

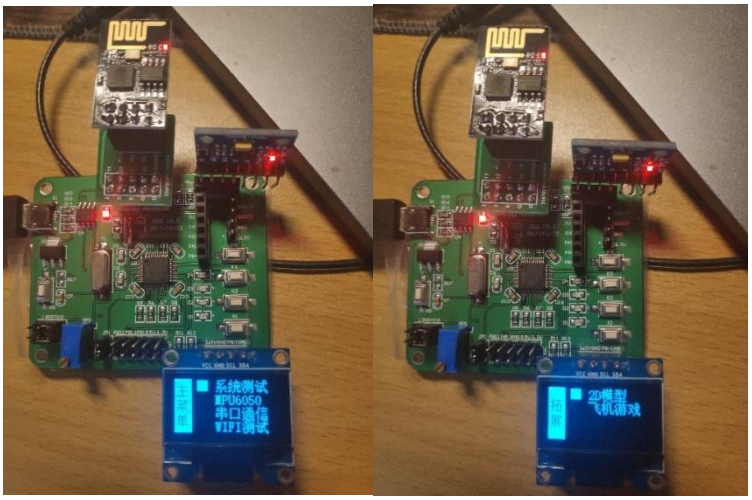
```
void DrawScreen6(void){
    //判断是否碰撞
    if(over==0){      //游戏未结束
        for(i=0;i<randi;i++){ //依次判断所有点
            if(randx[i]>=0&&randy[i]>=50){
                if(abs(randx[i]-sssx)<=6){   //判断是否碰撞
                    over=1;
                    break;}
                else{
```

```
                        if(randy[i]>=60){
                            randx[i]=-1;    //消掉这个点
                            randy[i]=-1;
                            ++flycnt;      //记录得分
                        }}}}}
    //生成点、控制点向下移动
    if (HAL_GetTick() > clockt + speed){    //speed 控制速度
        if(j==1&&(zantin==0)&&(over==0)){   //多重约束
            if(randi>=100)
                randi=0;
            randx[randi]=(int)(ax*1.7)%101;   //生成新的点
            randy[randi]=0;
            randi++;
            j=0;}
        if((int)(ax*1.7)%101>92)
            j=1;
        clockt = HAL_GetTick();
        if(zantin==0){
            for(i=0;i<randi;i++)   //下移所有点、受到暂停限制
                ++randy[i];
        }}
    //显示图形界面
    sssx=fAY;   //取 fAY 的值
    if(sssx>170)sssx=170;
    else if(sssx<10)sssx=10;
    sssx= (int)((sssx-10)*120.0/160); //归一化，将 10-160 映射到 0-120
    GUI_SetFont(&GUI_FontHZ_NewSimSun_16);
    if(over){   //游戏结束[略]   }
    else{ //游戏正常运行，显示界面[略]}
```

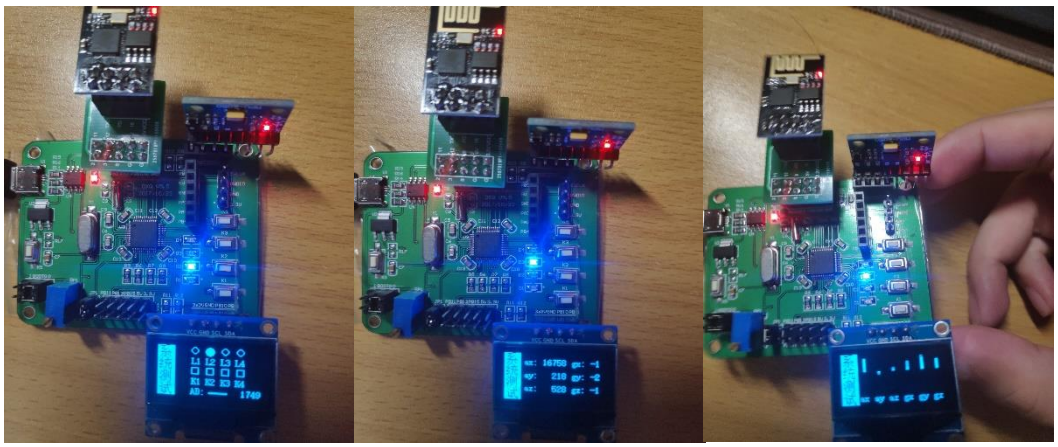
4 、功能测试

功能一：启动界面及按键控制菜单功能



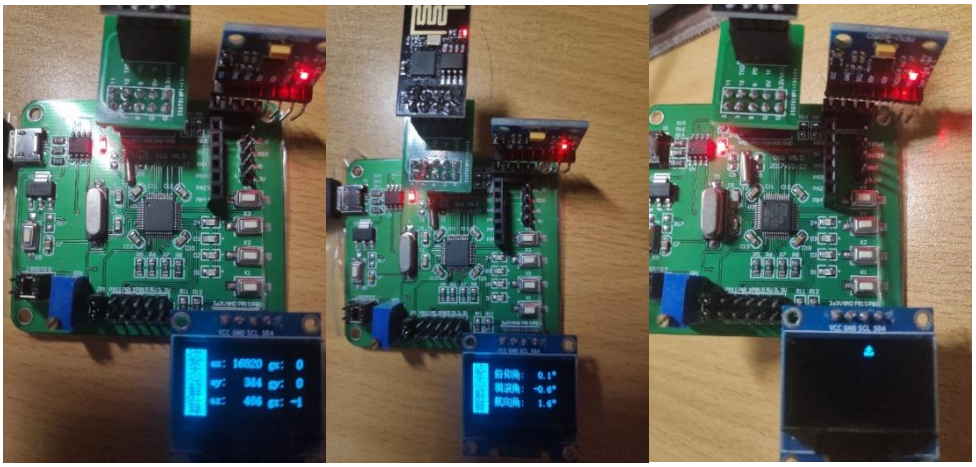


功能二：实现系统测试功能



其中，图一为 AD 值、KEY 和 LED 的显示和图形化，图二为陀螺仪 6 轴原始数据，图三为 6 轴数据的动态显示。

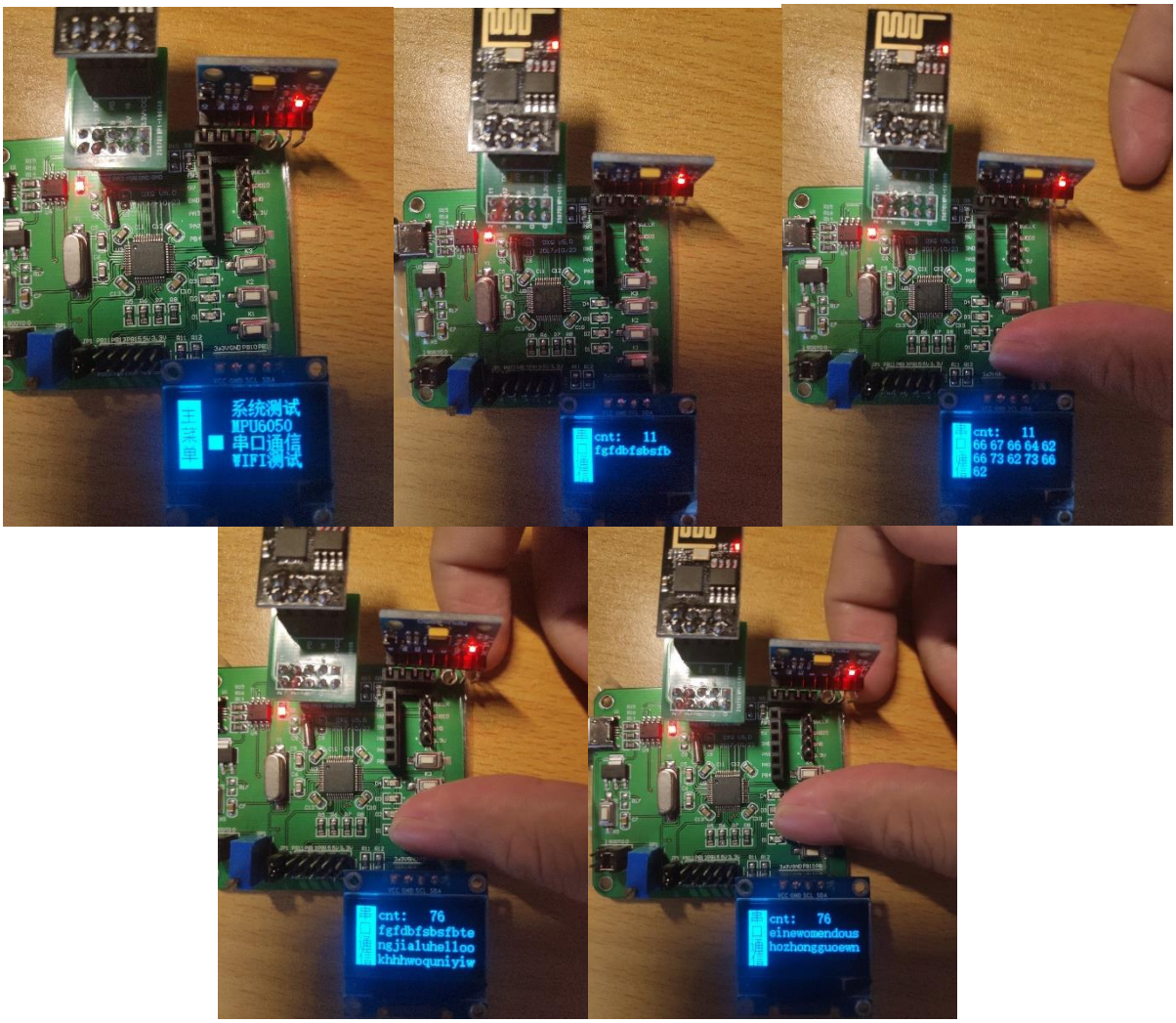
功能三：陀螺仪姿态解算



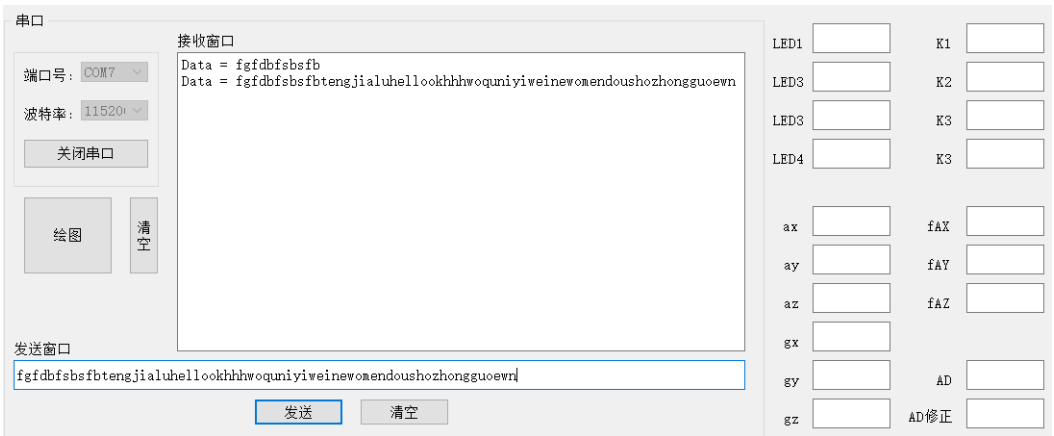
如图，图一为陀螺仪 6 轴原始数据，按键 K1 跳转到图二，3 个姿态角数据，K3 跳

转到陀螺仪的 2D 图形化。按键 K2 开启/关闭串口功能，向串口发送信息。K4 退出姿态解算功能。

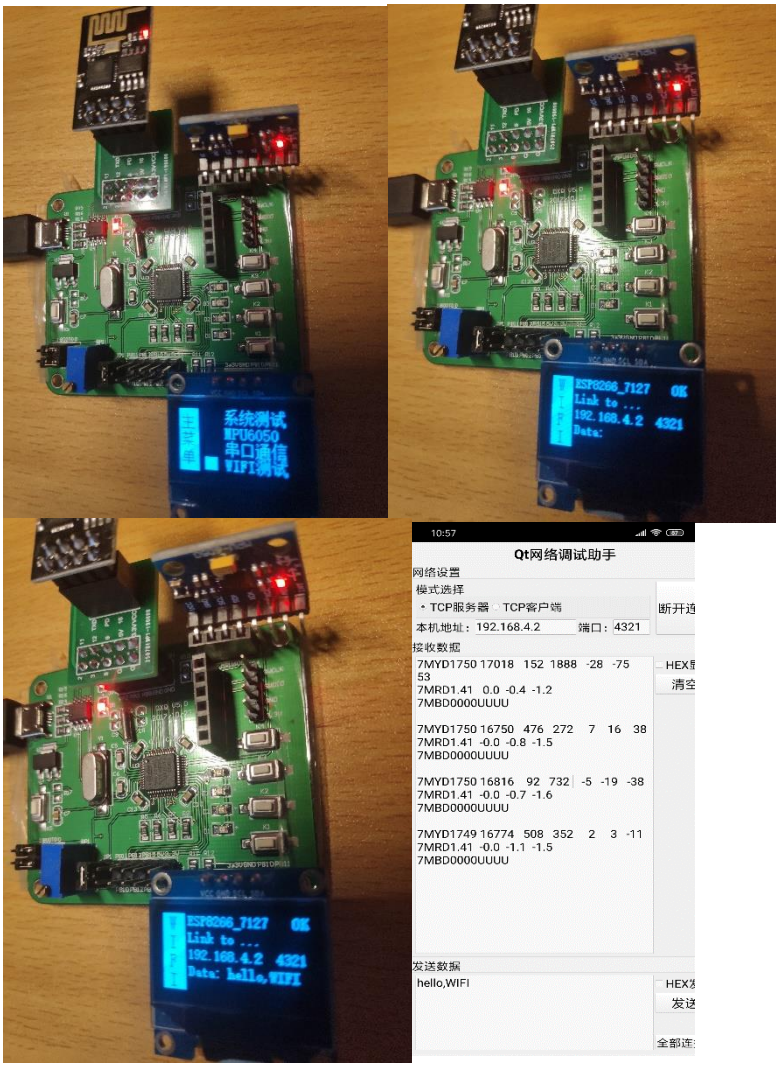
功能四：串口通信



如图所示：图一为串口通信菜单界面，图二为收到串口发送的数据时的显示情况，按下 K1 进入图三，数据的十六进制显示，图四为继续接收数据时的情况，此时一页已经显示不下，按下 K2 进行翻页，值得一提，只要数据不超过缓存区的大小，可以无限翻页。下图为对应的串口助手上位机所显示的从单片机返回的数据。

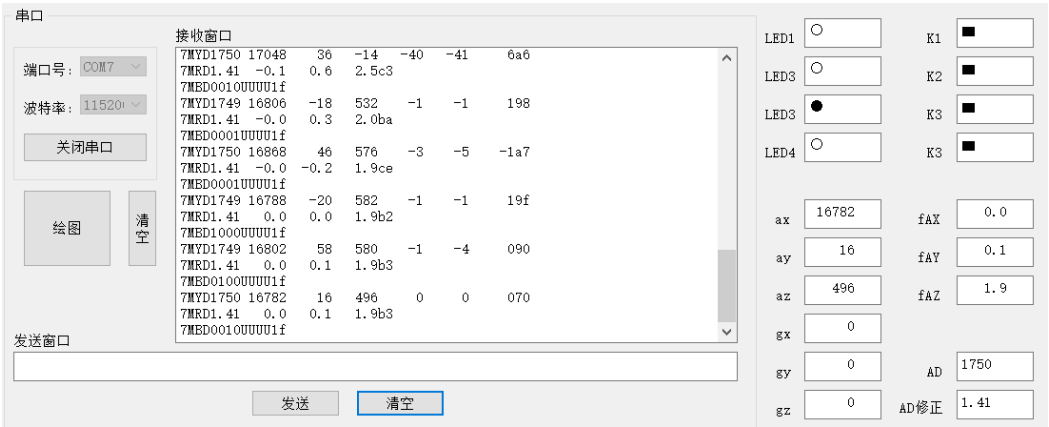


功能五：WIFI 通信

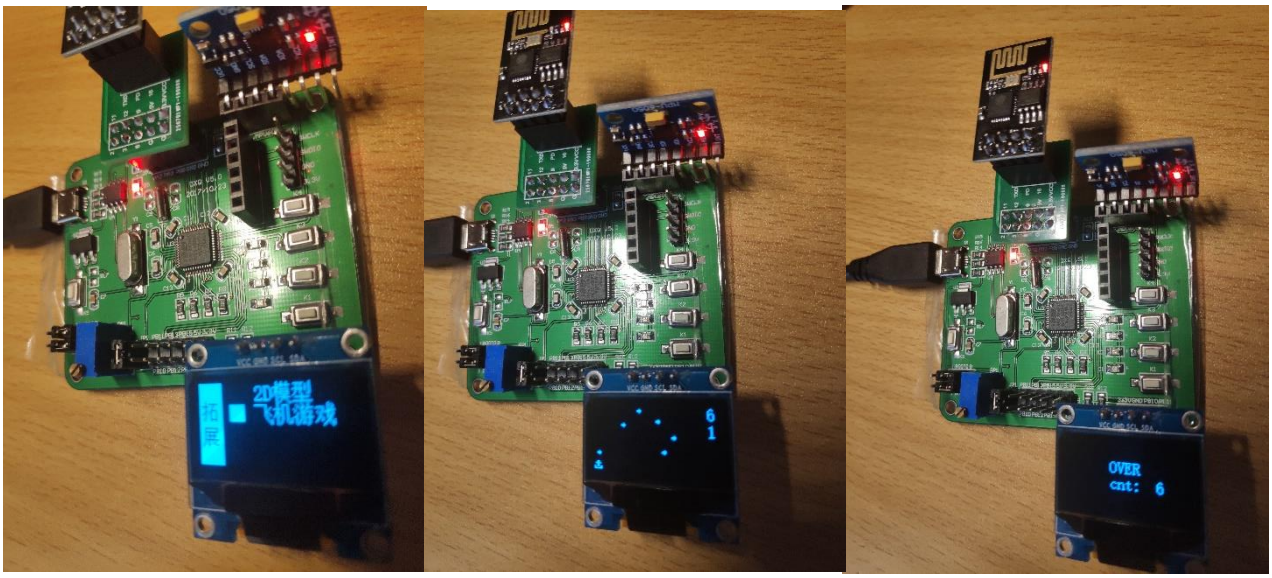


如图所示：图一为 WIFI 通信菜单界面，图二为 WIFI 连接成功，按下 K2，向手机端发送数据。图三为收到手机端发送来的数据。图四为手机界面。

功能六：串口助手上位机



拓展五：飞机游戏



如图所示：图一为飞机游戏的进入菜单，图二为正在进行的游戏，可以看到 x 屏幕顶部随机下落的*号，左下方为飞机，马上被*号击中，右上角 6 为当前的得分，1 为速度等级，当前为 1 级。游戏过程中可以按下 K2 键暂停游戏，按下 K1 键提高下落速度。图三为飞机被*号击中，显示游戏结束，得分为 6 分。此时，按下 K3 键游戏重新开始。按下 K4 退出游戏。

5 、课程反馈

通过这几天的设计经历，让我对 STM32 单片机开发程序又有了新的认识，我接触 STM32 也有两年多了，以前开发一个单片机程序就是使用裸机，利用中断和各种定时器经行任务调度，在上了这次课程实践后，给我最大的收获有两点：

第一点是 FreeRTOS 的微系统的任务调度策略带来编程的丝滑快感，原本比较难以实现的按键侦听和串口侦听，使用任务调度直接单独拎出来，独立实现，使得各个菜单之间的交互和变换变得简单而且还是模块化，使得后期菜单的拓展与修改变得及其容易；

第二点是 Qt 开发上位机、串口助手等其他软件的便捷与快速，这些比起 MFC、C#开发起来还要方便，上手及其快，实现功能时各类控件操作简便，虽然最后没有实现数据的动态显示，但整体思路却很清晰。

通过本次课程，在单片机、嵌入式开发的知识储备上又加固了一层，但也有一些不足的地方，在实现 WIFI 的时候出了很多问题，起初连个初始化都搞不定，最后也没有实现修改 IP 地址的重连的功能，个人感觉是我使用的程序框架与 WIFI 库里的不符合，连接服务器函数老是卡死。其他地方没遇到什么困难。

总之，本次课程收获颇丰。