
# 湖南大學

HUNAN UNIVERSITY

课 程 名 称：电子系统设计与创新基础训练

实验项目名称：短距离聊天室

专 业 班 级：计科 1905

姓　　名：李平凡

学　　号：201907040102

指 导 教 师：况玲

完 成 时 间：2021 年 9 月 14 日

信息科学与工程学院

# 摘要

为实现变长信息的发送，根据上位机发送信息智能实现硬件操作，以单片机集成度高，存储容量大，外部扩展能力强，控制功能强等优点作为硬件基础进行 c 语言编程实现，通过按键控制模式选择，信息发送，信息接收，字节大小调整，并结合串口助手实现串口通信，红外通信，智能通信。

关键词：串口通信；红外通信；智能通信；变长信息

# 目 录

# 第 1 章　绪 论

## 1.1 主题和选题

利用红外通信和串口通信，实现单片机与上位机之间，与其他单片机之间的交流，并根据上位机传来的信息进行单片机特定的操作。

## 1.2 题目背景

单片机的使用领域已十分广泛，如智能仪表、实时工控、通讯设备、导航系统、家用电器等。单片机由于集成度高，存储容量大，外部扩展能力强，控制功能强，广泛应用于节能控制，智能语音设备，报警控制，医疗设备等仪器设备。此外，单片机可以应用在 CPU 开发，程序开发，存储器开发，计算机开发及 C 语言程序开发，使用 C 语言进行编程，然后采用软硬件仿真的工作方式。生活中随处可见单片机的身影，航空的导航装值，飞行仪上各种仪表盘控制，广泛使用的 IC 卡，手机，摄像机，全自动洗衣机，电动玩具，电子宠物，智能手表等。

## 1.3 解决问题

1，能够实现变长信息的发送，使得单片机与上位机，与其他单片机的通信不仅仅局限于固定的字节大小。

2，能够根据上位机发送的指定信息进行相应的操作，达到智能通信的效果，比如发送指令或者信息使蜂鸣器发出连串的声音等等。

3，可以监测自己的信息发送是否准确。

# 第 2 章 系统设计

## 2.1 设计目的

串口助手文本模式和 Hex 模式下发送数据使得必须按照按照代码中提前设定好的大小发送数据，否则发送不出去或者发送到板子上的数据存在错误，本次设计是为了能够在板子上自己设定接收信息数据的大小，从而和上位机进行通信，同时接收上位机信息可以做一些事情。并且和其他单片机进行红外通信，另增加功能监测自己的信息发送是否正确。并且增加 AI 功能可以自己输入信息，使得单片机做出相应操作。

## 2.2 系统总体设计

用 K2 键实现模式转变，通过一个全局变量 funcmode 表示模式名称，并用 tempmode 来记录是某一种状态，之后 K1 和 K3 分别根据不同的 tempmode 实现对应功能，K1 键主要负责信息发送，K3 主要负责是否进入模式状态，同时数码管和 Led 灯也根据这个标志位来进行相应的显示。数码管的显示包括频闪效果，新增字母显示，Led 灯包括数据大小的显示（从右往左位次增加），流水灯显示。

## 2.3 系统各部分详细设计

2.3.1 按键模块：

Key2 实现模式功能的转变，Key1 在不同模式下正确执行自己的功能，发送对应的正确信息，Key3 能够正确控制模式进入和退出，导航按键能够正确改变数码管上显示的数据大小。

代码：

```
void deal_Key()
{
    if (GetKeyAct(enumKey1) == enumKeyPress) {                                          //K1主要负责信息发送
        if (tempmode == 2) {
            IrPrint(&wordtxt, t);
        }
        if (tempmode == 3)
            Uart1Print(&wordtxt, t);
        if (tempmode == 4) {
            if (music_on == 1)
            {
                SetBeep(1000, 20);
                fmc = !fmc;                                                             //在AI模式下提供的音乐编写模式中K1负责谱子的播放与暂停（通过改变fmc标志位）
                if (fmc == 1)
                    Uart1Print( "Let's Play", sizeof("Let's Play"));
            }
            else {
                SetBeep(1000, 20);
                Uart1Print( "Have a good day", sizeof("Have a good day"));
            }
        }
    }
    if (GetKeyAct(enumKey2) == enumKeyPress) {                                          //K2控制按键状态改变
        SetBeep(1000, 20);
        if (++funcmode > AI_Response) funcmode = Rectxt;
        M24C02_Write(0x00, funcmode);
        if (funcmode == Checktxt) {
            tempmode = 3;
            SetUart1Rxd(wordtxt, t, rxdhead, 0);
        }
        if (funcmode == Rectxt) {
            tempmode = 1;
            SetIrRxd(wordtxt);
            SetUart1Rxd(wordtxt, GetIrRxNum(), rxdhead, 0);
        }
        if (funcmode == Sendtxt) {
            tempmode = 2;
        }
        if (funcmode == AI_Response) {
            tempmode = 4;
        }
    }
}
```

```
void deal_Nav() {
    int i = 0;
    int op;
    char temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8;
    if ((tempmode == 2) || (tempmode == 4)) {
        temp1 = d1;
        temp2 = d2;
        temp3 = d3;
        temp4 = d4;

        if (GetAdcNavAct(enumAdcNavKeyLeft) == enumKeyPress)  {        //控制位数增加
            if (flag) if (weishu < 4) {
                    weishu++;
                    switch (weishu) {
                    case 1: d1 = 0; break;
                    case 2: d2 = 0; break;
                    case 3: d3 = 0; break;
                    case 4: d4 = 0; break;
                    default: break;
                    }
                    SetBeep(1000, 20);
                }
        }
        if (GetAdcNavAct(enumAdcNavKeyRight)  == enumKeyPress)  {      //控制位数减少
            if (flag) if (weishu > 1) {
                    weishu--;
                    switch (weishu) {
                    case 0: d1 = 10; break;
                    case 1: d2 = 10; break;
                    case 2: d3 = 10; break;
                    case 3: d4 = 10; break;
                    default: break;
                    }
                    SetBeep(1000, 20);
                }
        }
        if (GetAdcNavAct(enumAdcNavKeyUp) == enumKeyPress)  {          //控制数字增加
            if (flag) {
                switch (weishu) {
                case 1: d1++; break;
                case 2: d2++; break;
                case 3: d3++; break;
                case 4: d4++; break;
                default: break;
                }
            }
        }
```

```
    }
    if (GetAdcNavAct(enumAdcNavKeyDown) == enumKeyPress)  {        //控制数字减少
        if (flag) {
            switch (weishu) {
            case 1: d1--; break;
            case 2: d2--; break;
            case 3: d3--; break;
            case 4: d4--; break;
            default: break;
            }
        }
    }
    if (GetAdcNavAct(enumAdcNavKey3) == enumKeyPress) {            //控制模式进入并对缓冲区数组进行初始化
        flag = !flag;
        if (tempmode == 4)
            for (op = 0; op < 20; op++) {
                wordtxt[op] = '\0';
            }
        SetBeep(1000, 20);
    }
    if (temp1 == 10)temp1 = 0;
    if (temp2 == 10)temp2 = 0;
    if (temp3 == 10)temp3 = 0;
    if (temp4 == 10)temp4 = 0;
    t = temp4 * pow(10, 3) + temp3 * pow(10, 2) + temp2 * pow(10, 1) + temp1;//计算接收到的字节数
    SetUart1Rxd(wordtxt, t, rxdhead, 0);
```

2.3.2 显示模块：

led 灯在不同模式下以不同方式闪烁，比如以字节大小作为显示的参数进行显示，以流水灯的方式显示。数码管左边四位显示工作模式，右边显示（接收或设定的字节大小）在设定字节大小时，采用蜂鸣器提示按键是否有效以及通过频闪的方式判断某一位的改变并对某一位进行更改。

代码：

```c
void deal_Led()
{
    static i;
    if ((tempmode == 2) || (tempmode == 3))
        LedPrint(t);
    if ((tempmode == 0) || (tempmode == 4)) {              //以流水灯方式显示Led灯
        i++;
        if (i == 10) {
            i = 0;
            LedPrint(Ledt);
            Ledt = Ledt * 2;
            if (Ledt > 512)Ledt = 1;
        }
    }
    if (tempmode == 1) {                                   //以红外接收到的字节大小的方式显示Led灯
        LedPrint(GetIrRxNum());
    }
}
```

```c
void deal_Seg()
{
    char temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8;
    static unsigned char ct100mS = 10;
    if (tempmode == 2) {                    //send模式显示数码管，前四位代表模式缩写，后四位表示可设置的字节数
        temp1 = d1;
        temp2 = d2;
        temp3 = d3;
        temp4 = d4;
        temp5 = 10;
        temp6 = 29;
        temp7 = 27;
        temp8 = 5;
        if (--ct100mS == 0) ct100mS = 10;                   //实现数码管频闪效果
        if (ct100mS >= 8) {
            if (flag) {
                switch (weishu) {
                case 1: temp1 = 10; break;
                case 2: temp2 = 10; break;
                case 3: temp3 = 10; break;
                case 4: temp4 = 10; break;
                default: break;
                }
            }
        }
    }
    if (tempmode == 4) {                        //前四位AI模式缩写，后四位表示字节数
        temp1 = d1;
        temp2 = d2;
        temp3 = d3;
        temp4 = d4;
        temp5 = 12;
        temp6 = 12;
        temp7 = 31;
        temp8 = 26;
        if (--ct100mS == 0) ct100mS = 10;
        if (ct100mS >= 8) {
            if (flag) {
                switch (weishu) {
                case 1: temp1 = 10; break;
                case 2: temp2 = 10; break;
                case 3: temp3 = 10; break;
                case 4: temp4 = 10; break;
                default: break;
                }
            }
        }
    }
    if (tempmode == 1) {                        //前四位表示接收模式的缩写，后四位表示接收到的信息字节数
        temp8 = 26;
        temp7 = 27;
        temp6 = 28;
        temp5 = 10;
        temp4 = GetIrRxNum() / 1000 % 10;
        temp3 = GetIrRxNum() / 100 % 10;
        temp2 = GetIrRxNum() / 10 % 10;
        temp1 = GetIrRxNum() % 10;
    }
    if (tempmode == 3) {                        //前四位表示查看模式缩写，后四位表示字节数
        temp8 = 28;
        temp7 = 30;
        temp6 = 27;
        temp1 = d1;
        temp2 = d2;
        temp3 = d3;
        temp4 = d4;
        temp5 = 10;
    }
    if (tempmode == 0) {
        temp1 = 12;
        temp2 = 12;
        temp3 = 12;
        temp4 = 12;
        temp5 = 12;
        temp6 = 12;
        temp7 = 12;
        temp8 = 12;
    }
    Seg7Print(temp8, temp7, temp6, temp5, temp4, temp3, temp2, temp1);
```

2.3.3AI 模式：

会根据上位机发送的信息进行一个信息反馈，此时每次都需要对接收信息的缓冲区数组进行一个初始化，以便接收信息的准确性，然后和内部设定好的信息（指令）进行比较，从而更改标志位来实现单片机上的某些操作。

代码：

```c
void deal_AI()
{


    if (tempmode == 4) {
        if (strcmp(wordtxt, "music on") == 0) {           //对缓冲区字符串比较是否是music on如果是就进入该模式
            music_on = 1;                                 //音乐编写模式下的标志位打开
            Uart1Print("You are a creator", sizeof("You are a creator"));
        }
        /*if(music_on==1){
              Uart1Print("This is your music",sizeof("This is your music") );
        }*/
        if (strcmp(wordtxt, "music off") == 0) {        //退出音乐编写模式
            music_on = 0;
            Uart1Print("What do you want to do next", sizeof("What do you want to do next"));
        }
    }

}
```

```c
void deal_music()
{
    static ps = 0;
    if (music_on == 1) {
        if (fmc == 1) {
            switch (wordtxt[ps]) {
            case '1':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x21, 0x10);
                    ps++;
                }
                break;
            case '2':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x22, 0x10);
                    ps++;
                }
                break;
            case '3':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x23, 0x10);
                    ps++;
                }
            case '4':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x24, 0x10);
                    ps++;
                }
            case '5':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x25, 0x10);
                    ps++;
                }
            case '6':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x26, 0x10);
                    ps++;
                }
            case '7':
                if (GetBeepStatus() == 0) {
                    PlayTone(0xFA, 250 , 0x27, 0x10);
                    ps++;
                }
            }

        }
```

考虑到阻塞与非阻塞的问题，每 10ms 就会去检测蜂鸣器状态是否空闲，如果空闲就可以去读取下一个字符，并按照下一个字符对应的音调发音，否则就等到第二次 10ms 的事件触发，继续检查，当按下 K1 键便可以暂停音乐播放

2.3.4 通信模块：

通信模块包括三个方面，信息发送，信息监测，信息接收，信息发送需要自己在单片机上设定好固定的字节大小，然后上位机通过红外发送的方式发送自己设定字节大小的信息。信息监测是将这个信息发送给自己，来判断自己上位机的信息是否正确发送给单片机，信息接收是接收来自外部红外信息，同时将接收到的字节大小反馈在数码管上。

代码：

```c
void deal_Ir()
{
    if (tempmode == 1) {
        if (GetIrRxNum() != 0) {                //信息接收
            SetBeep(1000, 60);
            Uart1Print(&wordtxt, GetIrRxNum());
        }
    }

}
```

信息发送和信息检测代码与按键事件触发代码绑定

# 第 3 章 实验测试

## 3.1 实验过程

1，将程序下载到单片机后，首先进入待机状态，按下 K2 键更改模式，第一个 send 模式，发送信息，按下 K3 进入字节调整的模式，往左按下导航按键增加 1 位并可对该增加位进行修改，往上按数字加 1，往下按数字减 1，往右按位数减 1。调整好字节大小后上位机便发送自己设定好的字节大小信息（Led 灯也会显示该字节大小），按下 K1 键便可以通过红外发送的方式发送给其他处于接收状态的单片机。

2，继续按下 K2 键进入 check 模式，根据自己之前设定好的字节大小，check 模式下会默认将该字节大小显示在数码管右四位上（Led 灯也会显示），同时按下 K1 键便可将自己之前上位机发送给单片机的信息回馈给上位机，从而判断信息发送是否准确。

3，继续按下 K2 键进入 AI 模式，按下 K1 键会在上位机接收区显示“Have a good day”，由此表示进入到了 AI 模式，进入之后，设定 8 字节大小，输入 music on，上位机接受区显示“You are a creator”表示进入到了音乐编写的模式，接下来再设定字节大小输入 1234567 七位数字的组合信息，发送到单片机上，按下 K1 键，单片机便可按照这七位数字的组合信息发送相应的音调，并在上位机接收区返回信息“Let’s Play”表示正在演奏，再次按下 K1 键停止演奏，此时输入 9 字节大小的 music off 通过上位机接收区的信息“What do you want to do next”确定退出音乐编写模式。（由于代码资源限制不能再设定其他的 AI 功能）

4，继续按下 K2 键进入 receive 模式，接收来自外来的红外信息，当有红外信息发送被捕获，便会将该信息发送至上位机，并在数码管上显示接收到的信息字节大小。

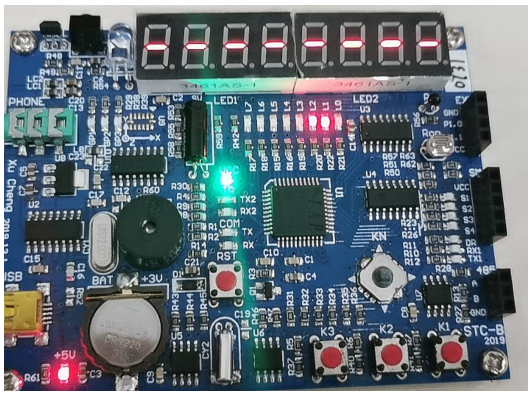
## 3.2 测试结果

1.软件测试

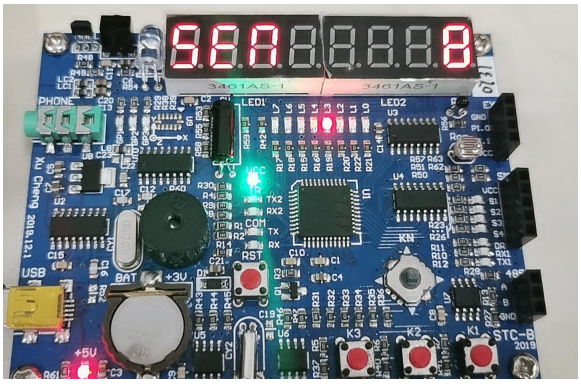
使用 keil4 软件完成工程后，调试无错误产生。

2.硬件测试

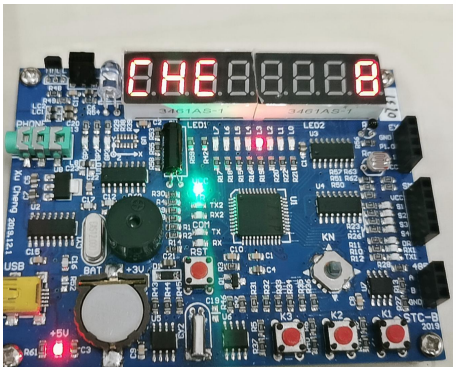
把生成的 hex 文件使用 STC-ISP 下载到学习板后，实现了对应的功能。Key2 实现模式功能的转变，Key1 在不同模式下正确执行自己的功能，发送对应的正确信息，Key3 能够正确控制模式进入和退出，导航按键能够正确改变数码管上显示的数据大小。

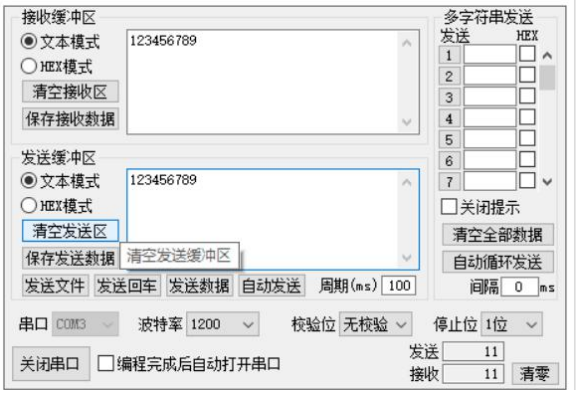


待机状态

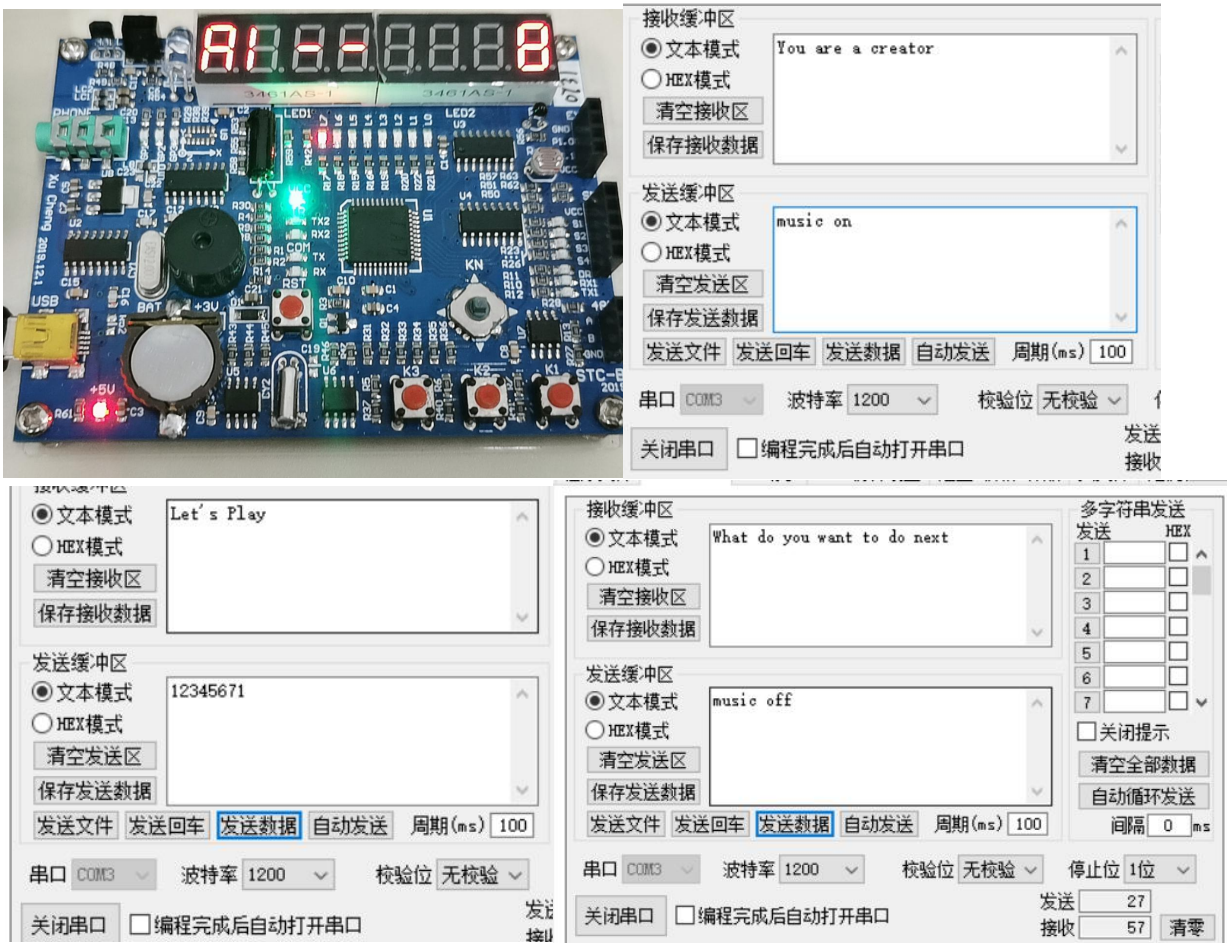


红外发送信息模式，此时可以发送 8 字节的数据






信息监测模式，在接收区查看信息发送是否正确




8 字节指令 music on 使得进入音乐编辑模式，根据七位数字进行乐谱的编写，并通过 K1 键使得蜂鸣器发音，输入 music off 退出音乐编辑模式。

# 第 4 章 实现难点说明

## 4.1 硬件原理

1，明白 LED 数码管如何编码进行相应的显示，基于 bsp 编程中 decode_table[]数组里面存储的是对应数字的编码方式，数码管从最上方开始顺时针转圈分别对应着从右往左的 8 个 bit 位，由此可以自己在 decode_table[]数组中加入自己想要的数码管显示方式（比如各种字母的显示，来表示进入到了某一种模式）

2，明白阻塞与非阻塞原理：当程序要求单片机的某一个设备进行操作的时候，如果不能获取到设备资源，就会将该程序休眠，处于挂起状态。但基于 bsp 编程，当遇到阻塞状态时会将相应的函数返回值设置为返回失败，失败的原因就是因为该设备正在被占用，不能去处理下一条程序。所以我们可

以设置 10ms 的回调函数每次去检测该设备是否空闲，如果空闲那么就可以去执行下一条我们设定好的程序。

3，基于 bsp 编程虽然方便，但是有的时候事件触发如果设定不当会导致硬件内部进行冲突，所以需要恰当使用事件触发函数，并且保证硬件资源不会被同时占用。

## 4.2 未解决问题

1，还没有实现前端页面制作，做出一个真正的前端聊天页面，目前仍然是使用串口助手来进行通信。

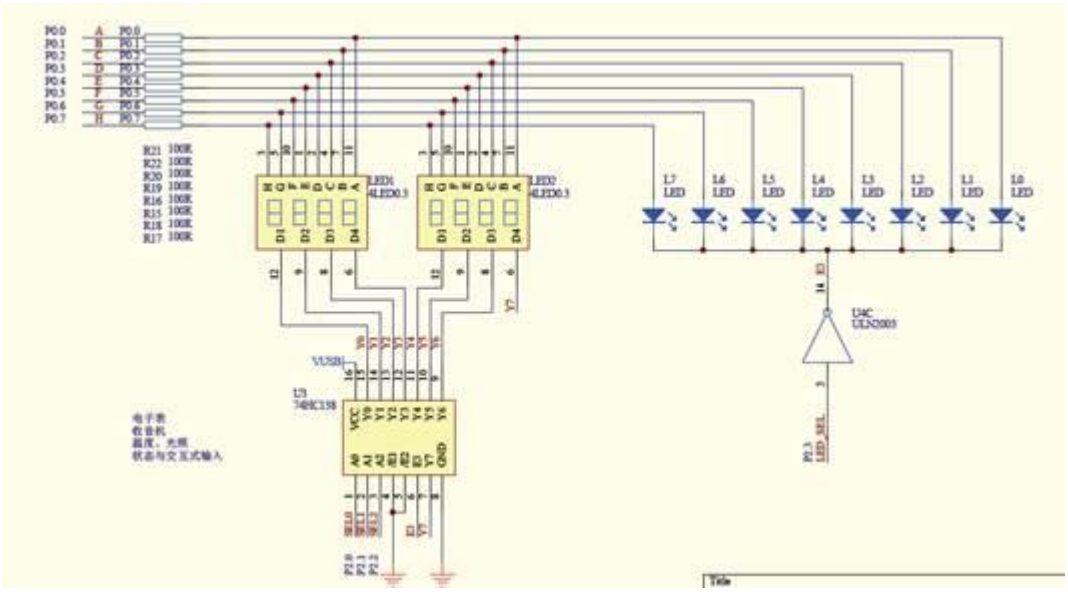
2，还不能解决多人通信红外相互干扰的问题，想过解决方法但还未实现，就是对每块单片机进行一个区别，加一个特定的编码，类似于 IP 地址，然后通过其中一块单片机（类似于服务器，对所有接受到的信息进行一个排队，依次向各个单片机发送信息，可以除了自己也可以包括自己）这样就可以实现私发和群发的功能。

# 第 5 章 总 结

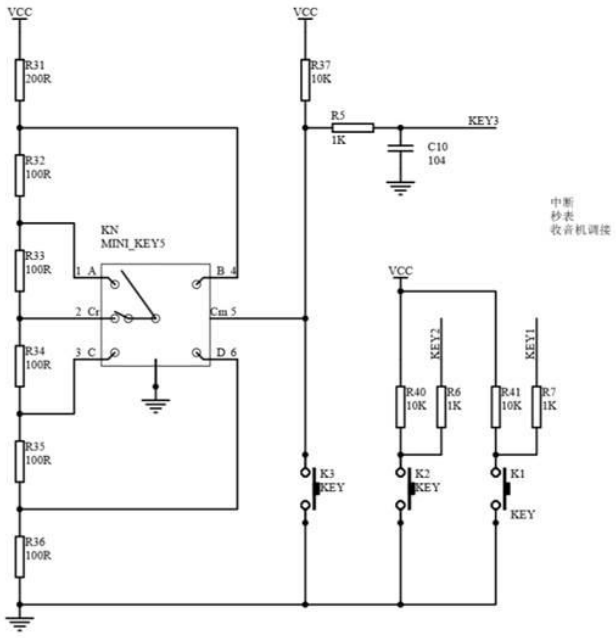
此项目期望值还未达满，但基本功能实现，由于单片机限制，不能继续扩充 AI 功能，在做此项目的过程中明白团队协作的重要性，以及要多向老师请教问题，在实现过程中，基于 bsp 的编程使得代码的编写方便许多，但是带来的底层冲突问题也经常让人摸不着头脑，从逻辑上来看可能没问题，但硬件上会带来许多冲突，典型的就是阻塞与非阻塞，同时许多初始化的函数需要相应的参数，否则可能带来引脚的冲突（我猜测（因为我没有初始化 adc 导致我没有编写导航按键中心键的功能，但却实现了导航按键其他按键的功能）），未达到的期望是自己还没有用其他语言制作一个聊天室的前端页面，也并没有通过单片机做到类似于服务器的功能达到真正的多人聊天，但自己的收获也很多，充分意识到单片机的集成度高，存储容量大，外部扩展能力强，控制功能强，等各项优点，在做项目的过程中也明白了团队帮助，老师指导的重要性。一个人自己挣扎那些 bug，倒不如请教别人来帮忙解决，能够省去很多无用功。如果还有机会，我会带着更多专业知识和技能去实现我想要实现的项目。

# 第 6 章 附录

## 6.1 设计硬件原理图



数码管与发光二极管硬件电路图




导航按键电路及工作原理说明

## 6.2 源程序

```
#include "ir.h"
#include "Key.H"
#include "sys.h"
#include "uart1.h"
#include "STC15F2K60S2.H"
#include "displayer.h"
#include "adc.h"
#include "stdio.H"
#include "math.h"
#include "beep.h"
#include "string.h"
#include "music.h"
#include "m24c02.h"
#include "stdlib.h"
code unsigned long SysClock=11059200;
#ifdef _displayer_H_
code char decode_table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00,0x08,0x40,0x01, 0x41,
0x48,
0x3f|0x80,0x06|0x80,0x5b|0x80,0x4f|0x80,0x66|0x80,0x6d|0x80,0x7d|0x80,0x07|0x80,0x7f|0x80,0x6f|0x80,0
x77,0x79,0x39,0x37,0x76,0x30};

#endif

unsigned char funcmode;
enum funcname {
        Rectxt,
        Sendtxt,
                                                Checktxt,
                                                AI_Response};
unsigned char tempmode=0;
```

```
unsigned char rxd[8];
unsigned char rxdhead[2]={0xaa,0x55};
unsigned char wordtxt[20];
unsigned char d1,d2,d3,d4,d5,d6,d7,d8;
static t;
static flag;
static fmc;
int weishu;
int Ledt=1;
int music_on=0;


void deal_Seg(){
                char temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8;
                static unsigned char ct100mS=10;
                if(tempmode==2){
                temp1=d1;
                temp2=d2;
                temp3=d3;
                temp4=d4;
                temp5=10;
                temp6=29;
                temp7=27;
                temp8=5;
                if(--ct100mS == 0) ct100mS=10;
                if(ct100mS >= 8){
                        if(flag){
                        switch(weishu){
                                case 1: temp1=10;break;
                                case 2: temp2=10;break;
                                case 3: temp3=10;break;
```

```
                    case 4: temp4=10;break;
                    default:break;
            }
            }
}
}
if(tempmode==4){
            temp1=d1;
            temp2=d2;
            temp3=d3;
            temp4=d4;
            temp5=12;
            temp6=12;
            temp7=31;
            temp8=26;
            if(--ct100mS == 0) ct100mS=10;
            if(ct100mS >= 8){
                    if(flag){
                    switch(weishu){
                    case 1: temp1=10;break;
                    case 2: temp2=10;break;
                    case 3: temp3=10;break;
                    case 4: temp4=10;break;
                    default:break;
                    }
            }
}

}
if(tempmode==1){
            temp8=26;
            temp7=27;
```

```
                temp6=28;
                temp5=10;
                temp4=GetIrRxNum()/1000%10;
                temp3=GetIrRxNum()/100%10;
                temp2=GetIrRxNum()/10%10;
                temp1=GetIrRxNum()%10;
        }
        if(tempmode==3){
                temp8=28;
                temp7=30;
                temp6=27;
                temp1=d1;
                temp2=d2;
                temp3=d3;
                temp4=d4;
                temp5=10;
        }
        if(tempmode==0){
                temp1=12;
                temp2=12;
                temp3=12;
                temp4=12;
                temp5=12;
                temp6=12;
                temp7=12;
                temp8=12;
        }
        Seg7Print(temp8,temp7,temp6,temp5,temp4,temp3,temp2,temp1);
}

void deal_Nav(){
        int i=0;
```

```
int op;
char temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8;
if((tempmode==2)||(tempmode==4)){
                                        temp1=d1;
                                        temp2=d2;
                                        temp3=d3;
                                        temp4=d4;

        if(GetAdcNavAct(enumAdcNavKeyLeft) == enumKeyPress)  {if(flag) if(weishu<4){
                weishu++;
                switch(weishu){
                                        case 1:d1=0;break;
                                        case 2:d2=0;break;
                                        case 3:d3=0;break;
                                        case 4:d4=0;break;
                                        default:break;
                }
                SetBeep(1000,20);
        }
}
if(GetAdcNavAct(enumAdcNavKeyRight)  == enumKeyPress)  {if(flag) if(weishu>1){
        weishu--;
        switch(weishu){
                case 0:d1=10;break;
                case 1:d2=10;break;
                case 2:d3=10;break;
                case 3:d4=10;break;
                default:break;
        }
        SetBeep(1000,20);
        }
}
```

```
if(GetAdcNavAct(enumAdcNavKeyUp)== enumKeyPress)  {
        if(flag){
                switch(weishu){
                        case 1:d1++;break;
                        case 2:d2++;break;
                        case 3:d3++;break;
                        case 4:d4++;break;
                        default:break;
                }
        }
}
if(GetAdcNavAct(enumAdcNavKeyDown)== enumKeyPress)  {
                if(flag){
                        switch(weishu){
                                case 1:d1--;break;
                                case 2:d2--;break;
                                case 3:d3--;break;
                                case 4:d4--;break;
                                default:break;
                        }
                }
        }
        if(GetAdcNavAct(enumAdcNavKey3)== enumKeyPress){
                flag=!flag;
                if(tempmode==4)
                        for(op=0;op<20;op++){
                                        wordtxt[op]='\0';
                        }
                SetBeep(1000,20);
        }
        if(temp1==10)temp1=0;
        if(temp2==10)temp2=0;
```

```
                if(temp3==10)temp3=0;
                if(temp4==10)temp4=0;
                t=temp4*pow(10,3)+temp3*pow(10,2)+temp2*pow(10,1)+temp1;
                SetUart1Rxd(wordtxt, t, rxdhead, 0);
            }
}


void deal_music(){
            static ps=0;
            if(music_on==1){
                if(fmc==1){
                    switch(wordtxt[ps]){
                        case '1':
                            if(GetBeepStatus()==0){
                                PlayTone(0xFA, 250 ,0x21, 0x10);
                                ps++;
                            }
                            break;
                        case '2':
                            if(GetBeepStatus()==0){
                                PlayTone(0xFA, 250 ,0x22, 0x10);
                                ps++;
                            }
                            break;
                        case '3':
                            if(GetBeepStatus()==0){
                                PlayTone(0xFA, 250 ,0x23, 0x10);
                                ps++;
                            }
                        case '4':
                            if(GetBeepStatus()==0){
```

```
                                        PlayTone(0xFA, 250 ,0x24, 0x10);
                                        ps++;
                                }
                        case '5':
                          if(GetBeepStatus()==0){
                                        PlayTone(0xFA, 250 ,0x25, 0x10);
                                        ps++;
                                }
                        case '6':
                                if(GetBeepStatus()==0){
                                        PlayTone(0xFA, 250 ,0x26, 0x10);
                                        ps++;
                                }
                        case '7':
                                if(GetBeepStatus()==0){
                                        PlayTone(0xFA, 250 ,0x27, 0x10);
                                        ps++;
                                }
                        }

                }

        }
        if(ps>=t){
                ps=0;
        }
}

void deal_AI(){
        if(tempmode==4){
                if(strcmp(wordtxt,"music on")==0){
                        music_on=1;
```

```
                                        Uart1Print("You are a creator",sizeof("You are a creator"));
                                }
                                if(strcmp(wordtxt,"music off")==0){
                                        music_on=0;
                                        Uart1Print("What do you want to do next",sizeof("What do
you want to do next"));
                                }
                        }

}
void deal_Ir(){
        if(tempmode==1){
                        if(GetIrRxNum()!=0){
                                SetBeep(1000,60);
                                Uart1Print(&wordtxt,GetIrRxNum());
                        }
        }

}

void deal_Key(){
                        if(GetKeyAct(enumKey1)==enumKeyPress){
                                if(tempmode==2){
                                        IrPrint(&wordtxt,t);
                                }
                                if(tempmode==3)
                                Uart1Print(&wordtxt,t);
                                if(tempmode==4){
                                        if(music_on==1)
                                        {
                                                SetBeep(1000,20);
                                                fmc=!fmc;
```

```
                                                    if(fmc==1)
                                                    Uart1Print( "Let's Play",sizeof("Let's
Play"));
                                            }
                                            else{
                                                    SetBeep(1000,20);
                                                    Uart1Print( "Have a good
day",sizeof("Have a good day"));
                                            }
                                    }
                            }
                            if (GetKeyAct(enumKey2)== enumKeyPress){
                                    SetBeep(1000,20);
                                    if(++funcmode > AI_Response) funcmode= Rectxt;
                                    M24C02_Write(0x00,funcmode);
                                    if (funcmode == Checktxt){
                                            tempmode=3;
                                            SetUart1Rxd(wordtxt, t, rxdhead, 0);
                                    }
                                    if(funcmode == Rectxt){
                                            tempmode=1;
                                            SetIrRxd(wordtxt);
                                            SetUart1Rxd(wordtxt,GetIrRxNum(), rxdhead, 0);
                                    }
                                    if(funcmode == Sendtxt){
                                            tempmode=2;
                                    }
                                    if (funcmode == AI_Response){
                                            tempmode=4;
                                    }
                            }
}
```

```
void deal_Led()
{
	static i;
	if((tempmode==2)||(tempmode==3))
			LedPrint(t);
	if((tempmode==0)||(tempmode==4)){
			i++;
			if(i==10){
					i=0;
					LedPrint(Ledt);
					Ledt=Ledt*2;
					if(Ledt>512)Ledt=1;
			}
	}
	if(tempmode==1){
			LedPrint(GetIrRxNum());
	}
}

void main(){
		MySTC_Init();
		IrInit(NEC_R05d);
		DisplayerInit();
		BeepInit();
		AdcInit(ADCexpEXT);
		Key_Init();
		MusicPlayerInit();
		Uart1Init(1200);
		d1=0;
		d2=10;
		d3=10;
```

```
d4=10;
d5=10;
d6=10;
d7=10;
d8=10;
weishu=0;
SetDisplayerArea(0,7);
SetEventCallBack(enumEventSys1mS , deal_Led);
SetEventCallBack(enumEventSys100mS , deal_Seg);
SetEventCallBack(enumEventSys10mS , deal_music);
SetEventCallBack(enumEventKey, deal_Key);
SetEventCallBack(enumEventIrRxd,deal_Ir);
SetEventCallBack(enumEventUart1Rxd,deal_AI);
SetEventCallBack(enumEventNav, deal_Nav);
while(1)
{ MySTC_OS();
}

}
```

# 第 7 章 参考文献

（4）你板子冒烟了，阻塞与非阻塞，52_阻塞与非阻塞_engineer0 的博客-CSDN 博客，2021-04-25

（4）嵌入式@hxydj，单片机串口实现字符串命令解析，单片机串口实现字符串命令解析_HXYDJ 的博客-CSDN 博客_串口字符串的解析，2020-10-30