微机原理及单片机课程设计

题　　目：电子音乐小钢琴

专　　业：电子信息工程

指导老师：葛玉荣、彭中怀

2019 年 6 月 18 日

一、团队介绍

姓名	学号	专业	分工
施林杰	17020022031	电子信息工程	代码编写，调试
唐梦蝶	17020022034	电子信息工程	写课程设计报告
田丰硕	17020022035	电子信息工程	程序调试，修改
刘志豪	17020022021	电子信息工程	资料收集曲谱制作

二、概述

2.1 单片机简介

单片机（Microcontrollers）是一种集成电路芯片，是采用超大规模集成电路技术把具有数据处理能力的中央处理器 CPU、随机存储器 RAM、只读存储器 ROM、多种 I/O 口和中断系统、定时器/计数器等功能（可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D 转换器等电路）集成到一块硅片上构成的一个小而完善的微型计算机系统，在工业控制领域广泛应用。从上世纪 80 年代，由当时的 4 位、8 位单片机，发展到现在的 300M 的高速单片机。

2.2 单片机特点及应用

单片机与微处理器有些不同， 微处理器的设计主要是考虑其计算机性能以及满足其 外接设备和网络接口的，而单片机则主要从工业控制方面出发，为了加强其控制能力， 从而提高工业环境下的可靠性、灵活性等。单片机有如下特点：

一、型号多样且种类繁多；

二、存储容量大；

三、频率和速度都高；

四、集成度高、可控性强；

五、功耗低；

六、配套软件多，易扩展。

正因为它有如此特点，使得其在许多领域都能得以应用：

一、在家用电器中的应用： 如今智能家居受到越来越多家庭的喜爱， 单片机控制的 智能家居让生活更加方便，更加安全。

二、在医疗设备中的运用：血糖仪，供氧设备，人体分析仪器，血养测试仪，等医学分析和生命科学仪器都与单片机有关，可用单片机控制。

三、在大型电器中的模块化运用：使用单片机控制大型设备的某些小模块，

并实现各个小模块之间协同控制，从而实现某一特定功能。

四、在工业控制方面:用单片机可以构成各种控制系统。例如可编程控制器，编码器，传动调速器，监控报警系统，与互联网组合构成多级控制系统等。

五、在汽车电子设备上的应用:如今人们买汽车除了其在机械上的性能之外最看重的就是汽车的电子设备,而单片机控制的电子设备不仅让汽车性能更加可靠，同时也使 人们有了更好的体验。

2.3 课题理解

这个课题非常新颖，以前从没想到能利用单片机来制作曲子，本质上这个课题就是利用单片机来控制蜂鸣器发出不同频率的声音来对应各个音阶,从而构成完整的曲子。其中既有最简单的一对一音符转换，也有复杂的中断操作等设计。同时还有完全陌生的红外通信以及键盘的映射原理,这些也都是在课堂上难以学到的知识。

三、硬件设计

本设计系统主要分为五个部分:红外接收模块、矩阵按键模块、蜂鸣器模块、数码管模块、点阵模块、AT89C516 单片机系统。整个设计以单片机为控制核心，处理接收到的红外信号,独立键盘按键信号并让蜂鸣器发声以及让数码管显示当前发声的音调。系统组成框如图 3.1。

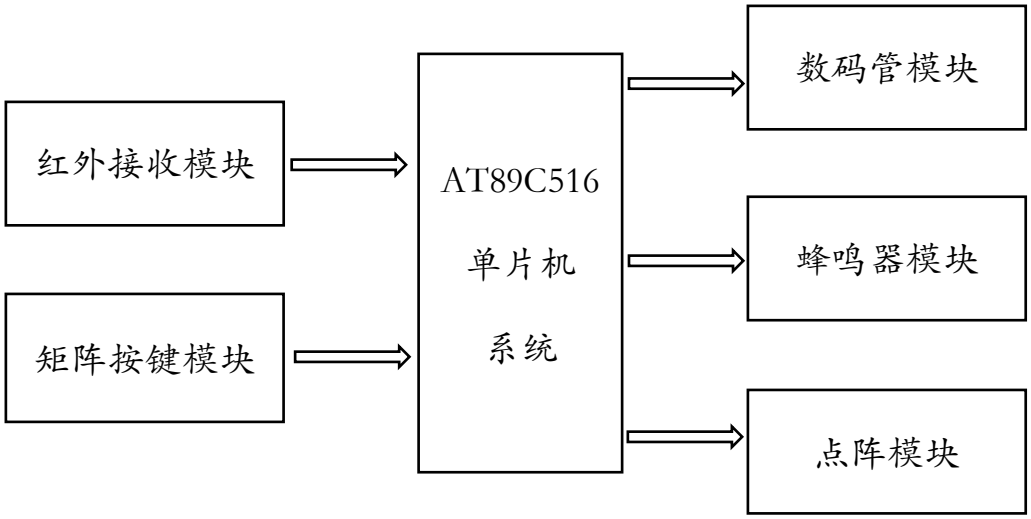


图 3.1 系统组成框图

3.1 STC89C51

此部分为单片机最小系统,单片机最小系统以 89C52 为核心 ,外加晶振电路、复位 电路、电源、 接地。电路结构构成简单，所以成本也比较低，但抗干扰能力还是很强的。为了让单片机有序运行，需要给其一个时钟模块作为参考，这就

是时钟振荡电路，电源所选用的是+5V 的电源，可直接由稳压电源提供，如图 3.2。

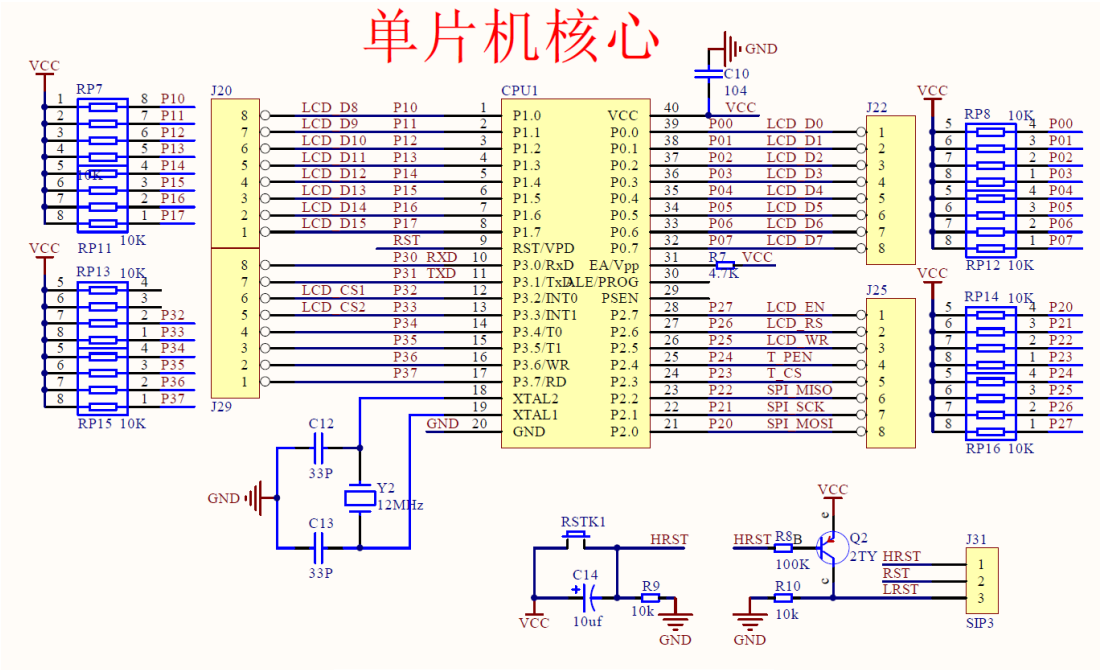


图 3.2 单片机核心

3.2 红外接收模块

红外遥控在现实生活中运用非常广泛，电视、空调、投影仪等各种电器都有用到,红外遥控距离长，抗干扰能力强，其次红外接收只占用到单片机一个 I/O 口，信号处理主要由程序编程决定，所以 I/O 占用率低。而红外接收模块主要是红外接收器 IRM 。接收器之所以可以接受红外线，是因为里面集成了一个红外信号收集放大电路。它仅仅只有三个管脚，电源正负极和信号输出端，如图 3.3。在两个引脚直接接上电源电压后她便是一个放大器，敏感度强且价格低廉，大小也合适，能方便的直接使用。在接收到已经被调制好后的信号后便能实现放大，选频和解调等功能， 并从输出端输出原始信号。 我们使用 STC89C51 单片机红外遥控来控制歌曲实现远程弹琴、歌曲切换功能，将 P3^2 管脚接在 J11 上。

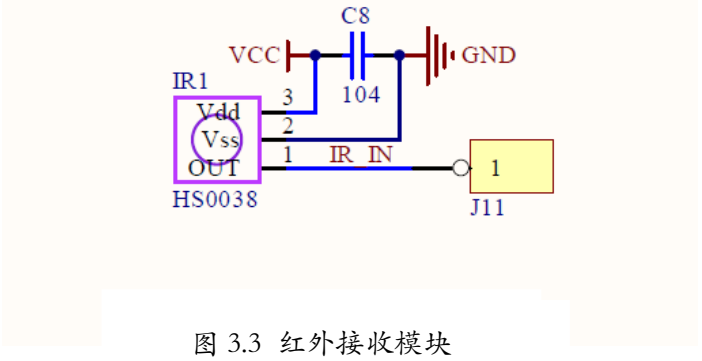


图 3.3 红外接收模块

这里对红外遥控原理做简单介绍： 当红外遥控的按键按下时，遥控首先发射一个完整的全码，一个完整的全码由引导码、用户码 、数据码和数据反码组成，如图 3.4。其中，引导码由 9ms 的起始码和 4.5ms 的结果码组成；系统码和数据码一共 32 位；其中前 16 位为用户识别码，用来识别不同型号的红外遥控，防止不同机种遥控红外信号互相串扰。后 16 位为各 8 位的操作码及其反码，该设置用于对接收数据进行比较核对以保证其数据的准确性。验证完成后单片机会根据接收到的红外数据码执行相应的动作。

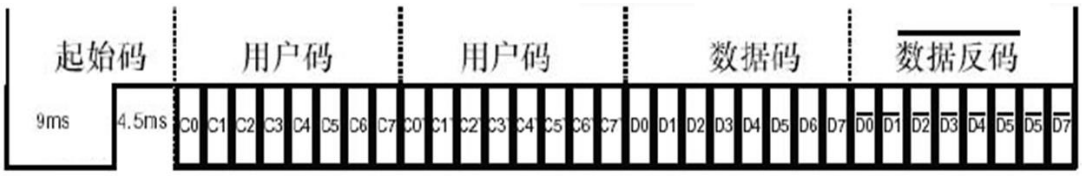


图 3.4 红外遥控发射全码

当按下一个按键不松时，系统会发射一个连发代码，连发代码是在持续按键时发 送的码 ,如果键按下超过 108ms 仍未松开，那么接下来发射的连发代码便仅由 2.5ms 的结束码和 9ms 的起始码组成 ,它主要是让接收端知道某键被按着而未松开。

红外接收头接收到红外信号后会传送到单片机，给单片机去进行红外解码。解码最关键的地方便是识别“0”和“1”。代码“0”的信号是由 0.56ms 的低电平和 0.565ms 的高电平组成、“1”代码则是由 0.56ms 的低电平和 1.69ms 的高电平组成，如图 3.5。所以我们主要是通过后面的高电平来区别0 和 1 的。

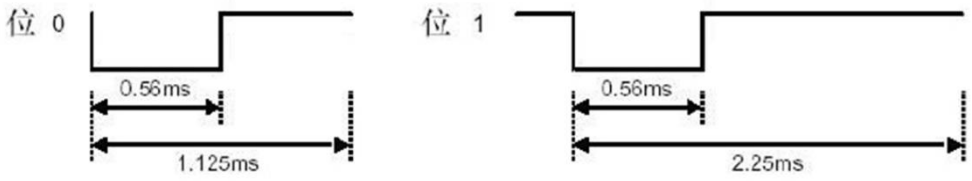


图 3.5 高低电平定义

3.3 矩阵按键模块

矩阵按键扫描原理

方法一：

逐行扫描：我们可以通过高四位轮流输出低电平来对矩阵键盘进行逐行扫描，当低四位接收到的数据不全为 1 的时候，说明有按键按下，然后通过接收到的数据是哪一位为 0 来判断是哪一个按键按下。

方法二：

行列扫描：我们可以通过高四位全部输出低电平，低四位输出高电平。当接

收到的数据，低四位不全为高电平时，说明有按键按下，然后通过接收的数据值，判断是哪一列有按键按下，然后再反过来，高四位输出高电平，低四位输出低电平，然后根据接收到的高四位的值判断是哪一行有按键按下，这样就能够确定是哪一个按键按下了。

本次设计中我们采用行列扫描的方法，我们将JP3跟单片机的P1端口连接。

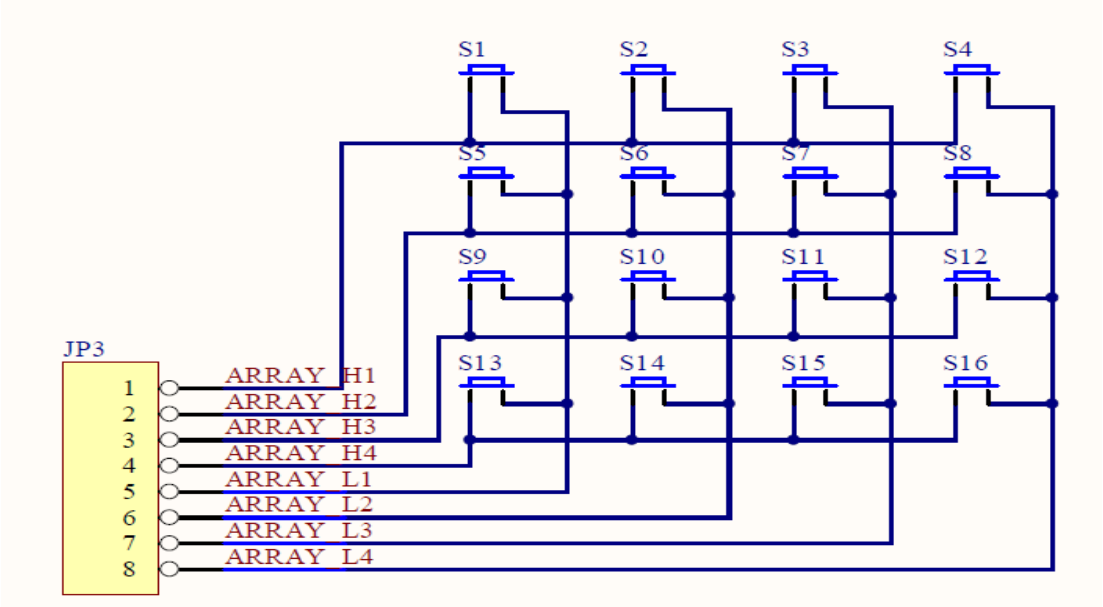


图 3.6 矩阵按键模块

3.4 蜂鸣器模块

压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配箱及共鸣箱、外壳等组成。多谐振荡器由晶体管或集成电路构成，当接通电源后（1.5~15V 直流工作电压），多谐振荡器，输出 1.5~2.5KHZ 的音频信号，阻抗匹配器推动压电蜂鸣片发声。

改变单片机引脚输出波形的频率，就可以调整控制蜂鸣器音调，产生各种不同音色、音调的声音。改变输出电平的高低电平占空比，就可以控制蜂鸣器的声音大小。我们将 P2^0 管脚跟 J7 连接。

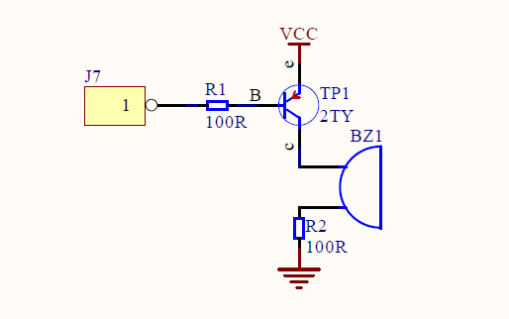


图 3.7 蜂鸣器模块

3.5 数码管模块

LED 数码管根据 LED 的不同接法可以分为 2 类：共阴和共阳。

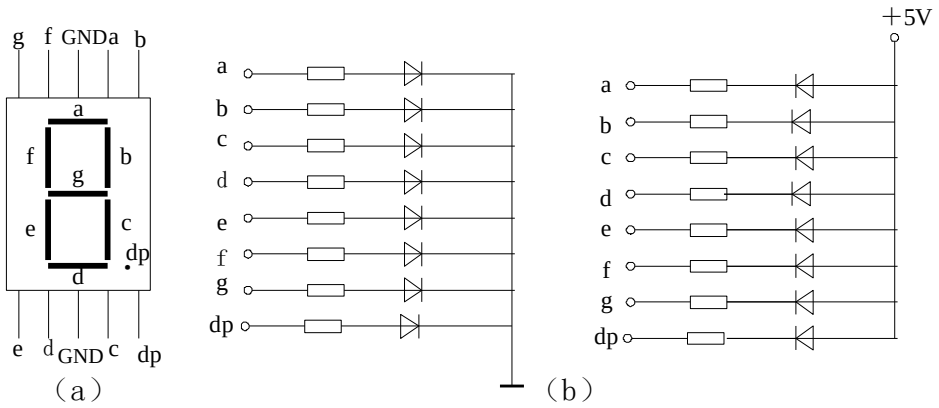


图 3.8 静态数码管共阴、共阳接法

共阴数码管表					
0x3f	0x06	0x5b	0x4f	0x66	0x6d
0	1	2	3	4	5
0x7d	0x07	0x7f	0x6f	0x77	0x7c
6	7	8	9	A	B
0x39	0x5e	0x79	0x71	0x00	
C	D	E	F	无显示	

LED 静态显示的特点就是每个数码管的段选必须接一个 8 位数据线来保持显示的字形码。当送入一次字形码后，显示字形可一直保持，直到送入新字形码为止。这种方法的优点是占用 CPU 时间少，显示便于监测和控制。缺点是硬件电路比较复杂，成本较高。我们将 J8 跟单片机的 P0 端口连接

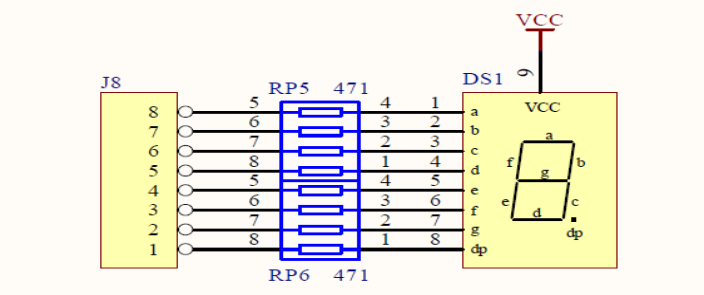


图 3.9 静态数码管模块

四、软件设计

4.1 整个项目的总体流程图

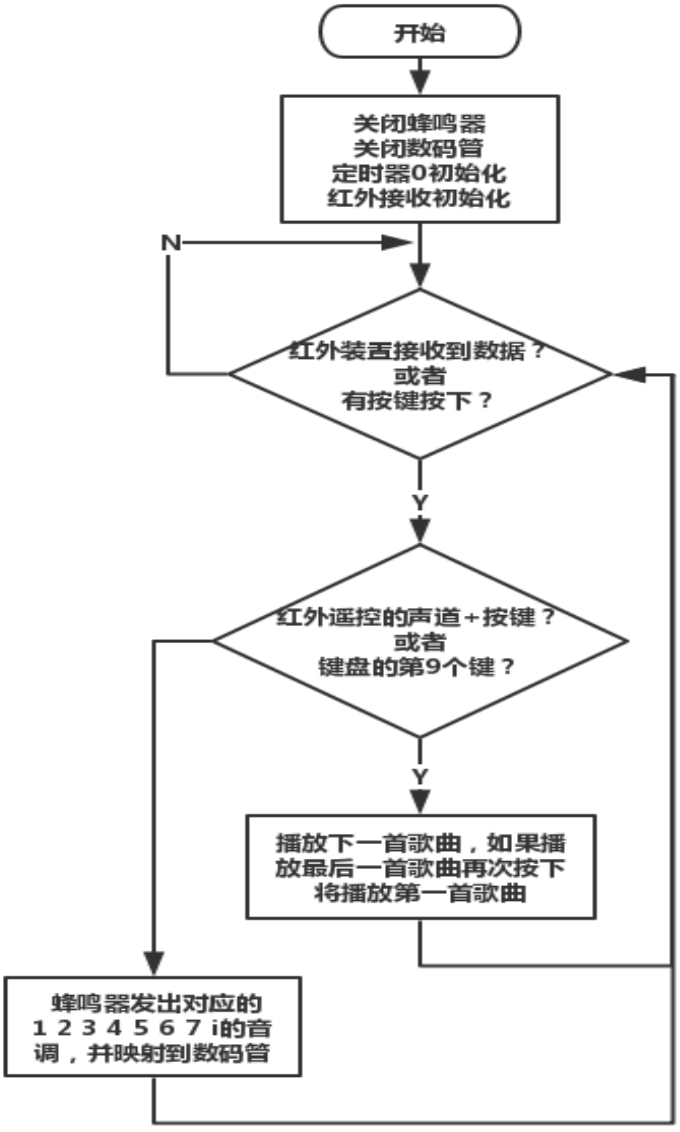


图 4.1 总体流程图

4.2 键盘控制模块

4.2.1 思路

我们通过行列扫描函数（KeyDown）来获取按键值以及红外接收转置来获取用户码，通过判断键值和用户码是否为我们定义的音符按键来实现按键控制音符输出效果，如果键值或用户码为我们定义的音符键值我们就从音符存放数组里取出每一个音符对应的音调，然后给定时器赋值，打开定时器中断控制定时器的定时时间，并从数码管断码数组里取出当前应付对应的值送到数码管显示。如果按

键为矩阵键盘的第九个按键或者红外遥控的声道+按键 5ms 则自动播放音乐，如果我们不断按这两个按键则实现歌曲切换，如果当前歌曲为最后一首则下一次播放第一首歌曲，实现歌曲循环播放。歌曲控制中我们用 music 存放播放哪一首歌，通过歌曲播放函数（song）来播放选中的歌曲

4.2.2 部分代码

```
/*********************************************************************************
* 函 数 名         : key_play
* 函数功能         : 按键控制音符以及歌曲切换
* 输    入         : 无
* 输    出         : 无
*********************************************************************************/
void key_play()
{
    KeyDown();
    //按键映射音符
    if (KeyValue == 0 || IrValue[2] == 0X0C)     //按下矩阵键盘的0键或者红外遥控的1键
    {
        temp1 = table[0];                        //为赋初值做准备
        temp2 = table[1];
        TH0 = temp1;
        TL0 = temp2;
        TR0 = 1;                                 //开始计时
        delay(100);                              //默认节拍是100ms
        key = 1;
        P0 = ~smgduan[key];                      //送数码管显示
    }
    //其余按键检测跟上述一致此处省略

    //歌曲切换控制
    if (KeyValue == 8 || IrValue[2] == 0X09)     //按下矩阵键盘的8键或者红外遥控的声道+键
    {
        delay(5);                                 //延迟5ms
        if (KeyValue == 8 || IrValue[2] == 0X09)//如果按键还按着
        {
            music++;                             //切换音乐
            if (music == 5)                      //四首歌循环播放
            {
                music = 0;
            }
            key = 9;
```

```
            P0 = ~smgduan[key];                 //送数码管显示
        }
    }
}
```

4.3 自动播放歌曲模块

4.3.1 思路

通过判断 music 的值从而确定播放哪一首歌，然后从不同歌曲的曲谱存放数组中取出音调和节拍给定时器 0 赋值，当音调为 0xff 时则结束播放，播放时依然要检测按键值和红外接收值判断是否结束播放执行其他功能。

4.3.2 部分代码

```
/*******************************************************************************
* 函 数 名       : song
* 函数功能       : 自动播放歌曲
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void song()
{
    if (music > 0)
    {
        key = 0;
        if (music == 1)                                 //判断按键是否按下
        {
            while (1)
            {
                KeyDown();
                key_play();
                if (key != 0)
                {
                    if (key != 9)
                    {
                        music = 0;
                    }
                    break;
                }
                TH0 = table1[music_dat[n][0]] / 256;    //赋初值
                TL0 = table1[music_dat[n][0]] % 256;
                TR0 = 1;                                //音乐开始
                delay100ms(music_dat[n][1]);        //调用延时，用做节拍的发生
                n++;                                //下个音调开始
```

```
                if (music_dat[n][0] == 0xff)
                {
                    n = 0;                              //判断是否到最后一个音调
                    TR0 = 0;                            //一个调放完，即将进行下一个
调
                }
            }
        }
        //其余歌曲播放跟上述代码一致，此处省略
    else
    {
        TR0 = 0;
        TR1 = 0;
        beep = 1;
    }
}
```

五、记录

5.1 问题

1、如何实现按键控制歌曲切换。歌曲播放的时候我们需要一个循环如果没有外部控制结束就要到歌曲播放结束才能退出，在播放过程中若有按键按下就要执行其他功能，但是在做的时候无法退出这个播放过程。
2、按下歌曲播放按键后，继续按这个按键无法切歌，但是按下其他按键可以执行其他功能。

5.2 解决方法

1、在 key_play 函数中定义一个变量 key 联合 music 变量来判断是否符合播放条件，只有 music!=0 和 key!=0 的时候才进行歌曲播放，具体实现为如下代码：

```
void song()
{
    if (music > 0)
    {
        key = 0;
        if (music == 1)                                 //判断按键是否按下
        {
            while (1)
            {
                KeyDown();
                key_play();
```

```
                if (key != 0)
                {
                    if (key != 9)
                    {
                        music = 0;
                    }
                    break;
                }
                TH0 = table1[music_dat[n][0]] / 256;    //赋初值
                TL0 = table1[music_dat[n][0]] % 256;
                TR0 = 1;                                //音乐开始
                delay100ms(music_dat[n][1]);            //调用延时，用做节拍的发生
                n++;                                    //下个音调开始
                if (music_dat[n][0] == 0xff)
                {
                    n = 0;                              //判断是否到最后一个音调
                    TR0 = 0;                            //一个调放完，即将进行下一个
调
                }
            }
        }
    }
```

2、由于我们用了一个变量（KeyValue）来存放键值和数组（IrValue）来存放红外接收的全码，当我们按下矩阵键盘的一个按键或者红外遥控的一个按键时这个值会被保存，直到有新的按键按下，所以我们在每次结束的时候给这两个变量赋一个新的值 0xff，来解决这个问题。

六、成果展示

本次设计实现的功能：

1、蜂鸣器叫 8 个音符的输出效果；

2、键盘按下对不同音符的映射功能；

3、单片机自动播放《Kiss The Rain》《Big Big World》《铃儿响叮当》《小小姑娘》等曲子；

4、显示当前音符对应的 LED。

未实现的功能：

在点阵或液晶屏动态更新当前演奏或播放的动态频谱效果。

实现思路：采用傅立叶转换，将采样到的数据使用 8*8 的 LED 排灯输出，那么可以使用 74HC595 来做驱动，采样 128 个点。那么，第一个点开启第一排的 LED，然后送数据显示，然后第二点送第二排的 LED，然后送数据显示，依次类

推。

七、后记

通过此次课程设计，我学到了单片机的相关编译，烧录程序的使用，关于键盘的使用和红外通信原理，动手能力得到了提升。

——刘志豪

通过这次的课程设计，我知道了如何根据芯片手册来写代码，或者通过芯片手册理解这个模块的工作原理，这次课程设计过程中查找了很多资料，如果要做成一件事要花很多时间去做，这次课程设计虽然花了很多时间但是收获还是挺大的。

——施林杰

这次的程序设计大大提高了我的动手实践能力，提高对于代码的理解和运用。

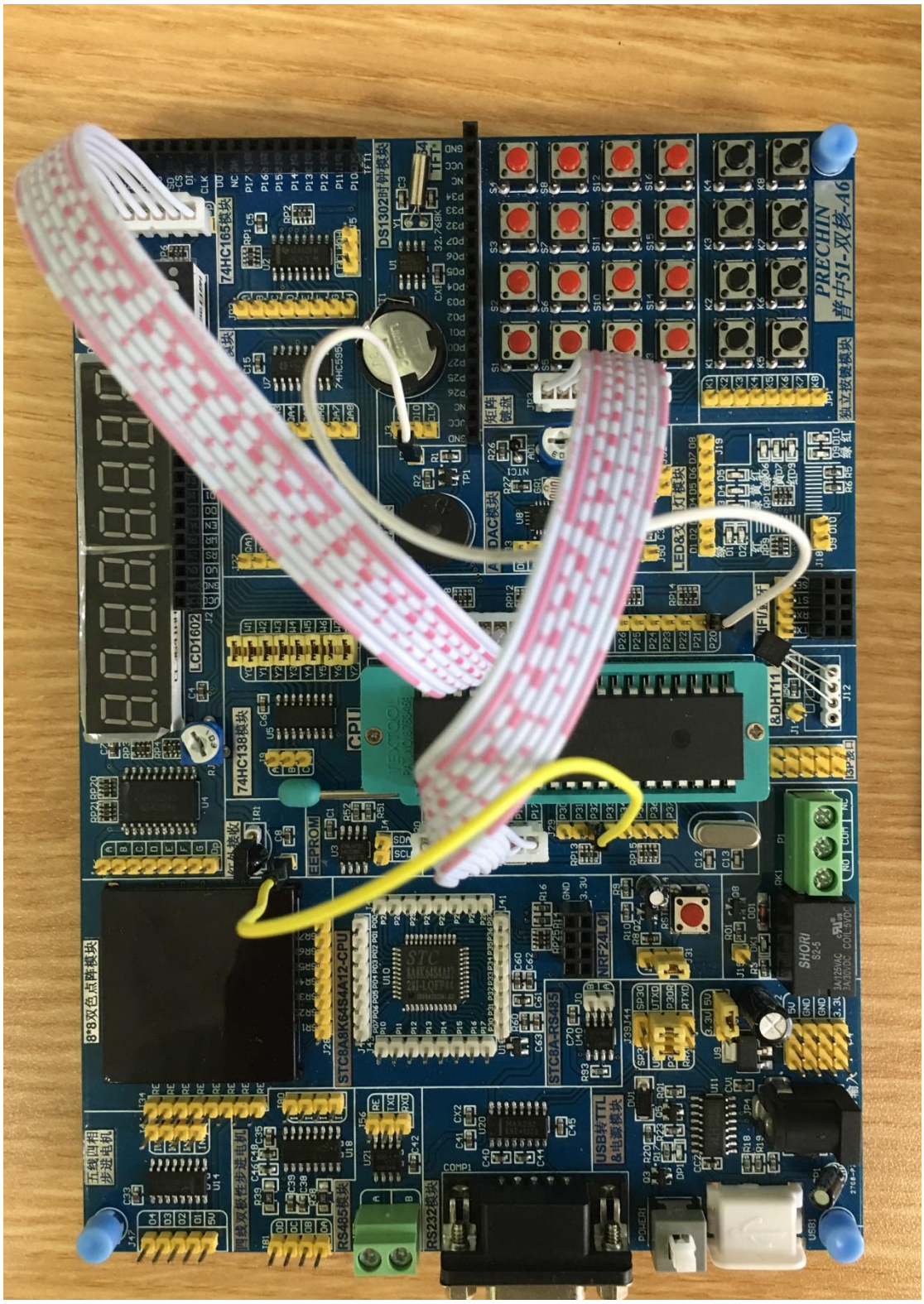
——田丰硕

在做这个课程设计之前需要下很多功夫去了解 51 单片机的相关操作对应的代码的书写，也需要一步一步去实现它的初级功能，最后才能形成思路去做我们的设计，让我对单片机有了更深刻的理解，不止局限于理论知识。

——唐梦蝶

八、附录

8.1 连接图


PRECHIN
普中51-双核-A6
DS1302时钟模块
74HC165模块
LCD1602
74HC138模块
EEPROM
CPU
8*8双色点阵模块
STC8A8K64S4A12-CPU
STC8A-RS485
NRF24L01
USB转TTL
&电源模块
RS232模块
RS485模块
DAC模块
LED&灯模块
8DHT11
独立按键模块
矩阵键盘
五线四相
步进电机
四线双极性步进电机

8.2 源码

```c
//main.c
#include "reg52.h"
#include "public.h"
#include "key.h"
#include "song.h"
#include "readir.h"
#include "time0.h"
u8 music = 0;              //音乐播放曲目
u8 n = 0;                  //读取第一首歌的节拍
u8 n1 = 0;                 //读取第二首歌的节拍
u8 n2 = 0;                 //读取第三首歌的节拍
u8 n3 = 0;                 //读取第四首歌的节拍
u8 key = 0;                //取数码管断码
u8 KeyValue = 0xff;        //矩阵键盘键值赋值
u8 IrValue[6] = 0;         //存放红外接收数据
u8 temp1 = 0;              //临时保存定时器0的TH0
u8 temp2 = 0;              //临时保存定时器0的TL0
u8 Time = 0;               //红外解码时判断高电平时间
/*******************************************************************************
* 函 数 名       : main
* 函数功能       : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
    beep = 0;
    sumaguan = 0;
    systimer0_init();      //定时器0初始化
    IrInit();              //红外控制初始化
    while (1)
    {
        beep = 1;          //打开蜂鸣器
        key_play();        //按键控制函数
        song();            //自动播放歌曲函数
    }
}

//key.c
#include "reg52.h"
#include "public.h"
#include "key.h"
/*******************************************************************************
* 函 数 名       : KeyDown
```

```
* 函数功能        : 读取矩阵键盘键值
* 输    入        : 无
* 输    出        : 无
*******************************************************************************/
void KeyDown()
{
    char a = 0;
    GPIO_KEY = 0x0f;
    if (GPIO_KEY != 0x0f)
    {
        delay1(1000);
        if (GPIO_KEY != 0x0f)
        {
            switch (GPIO_KEY)
            {
            case 0x07: KeyValue = 0; break;
            case 0x0b: KeyValue = 1; break;
            case 0x0d: KeyValue = 2; break;
            case 0x0e: KeyValue = 3; break;
            }
            GPIO_KEY = 0xf0;
            switch (GPIO_KEY)
            {
            case 0x70: KeyValue = KeyValue; break;
            case 0xb0: KeyValue = KeyValue + 4; break;
            case 0xd0: KeyValue = KeyValue + 8; break;
            case 0xe0: KeyValue = KeyValue + 12; break;
            }
            while ((a < 50) && (GPIO_KEY != 0xf0))//强制退出
            {
                delay1(1000);
                a++;
            }
        }
    }
}
/*******************************************************************************
* 函 数 名        : key_play
* 函数功能        : 按键控制音符以及歌曲切换
* 输    入        : 无
* 输    出        : 无
*******************************************************************************/
void key_play()
{
```

```
KeyDown();
if (KeyValue == 0 || IrValue[2] == 0X0C)     //按下矩阵键盘的0键或者红外遥控的1
{
    temp1 = table[0];                        //为赋初值做准备
    temp2 = table[1];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 1;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 1 || IrValue[2] == 0X18)     //按下矩阵键盘的1键或者红外遥控的2键
{
    temp1 = table[2];                        //为赋初值做准备
    temp2 = table[3];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;
    delay(100);                              //默认节拍是100ms
    key = 2;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 2 || IrValue[2] == 0X5E)     //按下矩阵键盘的2键或者红外遥控的3键
{
    temp1 = table[4];                        //为赋初值做准备
    temp2 = table[5];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 3;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 3 || IrValue[2] == 0X08)     //按下矩阵键盘的3键或者红外遥控的4键
{
    temp1 = table[6];                        //为赋初值做准备
    temp2 = table[7];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 4;
    P0 = ~smgduan[key];                      //送数码管显示
```

```
}
if (KeyValue == 4 || IrValue[2] == 0X1C)     //按下矩阵键盘的4键或者红外遥控的5键
{
    temp1 = table[8];                        //为赋初值做准备
    temp2 = table[9];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 5;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 5 || IrValue[2] == 0X5A)     //按下矩阵键盘的5键或者红外遥控的6键
{
    temp1 = table[10];                       //为赋初值做准备
    temp2 = table[11];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 6;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 6 || IrValue[2] == 0X42)     //按下矩阵键盘的6键或者红外遥控的7键
{
    temp1 = table[12];                       //为赋初值做准备
    temp2 = table[13];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 7;
    P0 = ~smgduan[key];                      //送数码管显示
}
if (KeyValue == 7 || IrValue[2] == 0X52)     //按下矩阵键盘的7键或者红外遥控的8键
{
    temp1 = table[14];                       //为赋初值做准备
    temp2 = table[15];
    TH0 = temp1;
    TL0 = temp2;
    TR0 = 1;                                 //开始计时
    delay(100);                              //默认节拍是100ms
    key = 8;
    P0 = ~smgduan[key];                      //送数码管显示
```

```
    }
    if (KeyValue == 8 || IrValue[2] == 0X09)     //按下矩阵键盘的8键或者红外遥控的声道+
键
    {
        delay(5);                                //延迟5ms
        if (KeyValue == 8 || IrValue[2] == 0X09)//如果按键还按着
        {
            music++;                             //切换音乐
            if (music == 5)                      //四首歌循环播放
            {
                music = 0;
            }
            key = 9;
            P0 = ~smgduan[key];                  //送数码管显示
        }
    }
    TR0 = 0;                                     //计时器停止，也就是停止放音
    IrValue[2] = 0XFF;                           //由于解码之后临时保存了按下按键的码
值，
                                                 //会导致没有其他按键按下时一直在重复上
一次的功能
    KeyValue = 0xff;                             //同上
}

//public.c
#include "reg52.h"
#include "public.h"

u16 code smgduan[17] = { 0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
                         0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71
};//显示0~F的值

/************************** Kiss The Rain **************************/
//下面每一组是【音调】【节拍】
//比如说：1，4。其中1是音调，在函数中取得1的值然后在table[]中取得相应的音调
//其中4是节拍，通过这个值来确定延时的长短
u8 code music_dat[][2] = {
    5,2, 1,2, 2,2, 2,4, 3,4, 3,16,
    1,2, 2,2, 3,2, 2,4, 5,4, 5,16,
    5,2, 6,2, 7,2, 7,4, 1,4, 1,16,
    2,4, 3,4, 2,4, 1,4, 7,8,
    1,4, 7,4, 5,4, 5,4, 6,4, 6,8,
    5,2, 4,2, 4,4, 5,4, 5,8,
    1,4, 2,4, 3,4, 4,4, 4,16,
```

```
    5,4,  4,4,  3,16,  2,4,  5,4,  1,4,  2,4,  2,4,  3,4,  3,16,
    1,2,  2,2,  3,2,  2,4,  5,4,  5,16,
    5,2,  6,2,  7,2,  7,4,  1,4,  1,16,
    2,4,  3,4,  2,4,  1,4,  7,16,
    1,4,  7,4,  5,4,  5,4,  6,4,  6,8,
    5,2,  4,2,  4,4,  5,4,  5,8,  5,8,
    1,2,  2,2,
0XFF                                        //结束
};

/*************************小小姑娘*****************************/
u8 code music_dat1[][2] = {
    1,3,  1,3,  1,6,  5,6,  3,3,  3,3,
    3,6,  1,6,  1,3,  3,3,  5,6,  5,3,
    4,3,  3,3,  2,12,  2,3,  3,3,  4,6,  4,6,
    3,3,  2,3,  3,6,  1,6,  1,3,  3,3,  2,6,  5,6,
    7,3,  2,3,  1,12,
    0XFF                                    //结束
};

/*************************铃儿响叮当*****************************/
u8 code music_dat2[][2] = {
    6,2,  6,2,  6,4,  6,2,    6,2,  6,4,                    //第一节
    6,2,  8,2,  4,3,  5,1,  6,8,                          //第二节
    7,2,  7,2,  7,3,  7,1,  7,2,  6,2,  6,2,  6,1,  6,1,//第三节
    6,2,  5,2,  5,2,  4,2,  5,4,  8,4,                    //第四节
    6,2,  6,2,  6,4,  6,2,  6,2,  6,4,                    //第五节
    6,2,  8,2,  4,3,  5,1,  6,8,                          //第六节
    7,2,  7,2,  7,3,  7,1,  7,2,  6,2,  6,2,  6,1,  6,1,//第七节
    8,2,  8,2,  7,2,  5,2,  4,6,                          //第八节
    0XFF                                                  //结束
};

/*******************big big world*************************/
u8 code music_dat3[][2] = {
    1,4,  2,4,  3,8,  3,8,  3,8,  3,4,  4,4,  2,8,  2,8,  2,4,
    2,4,  2,4,  3,4,  1,8,  1,8,  1,8,  1,4,  2,4,  3,16,  2,8,
    1,4,  2,4,  3,8,  3,8,  3,8,  3,4,  4,4,  2,8,  2,8,  2,8,
    2,4,  3,4,  3,2,  2,2,  1,8,  1,16,  5,4,  3,4,  2,4,  2,4,
    3,16,  3,16,  2,8,  5,8,  1,32,  5,8,  3,8,  3,4,  3,2,  2,2,
    2,16,  2,8,  3,8,  2,8,  1,8,  0,8,  1,4,  1,4,  1,4,  6,4,
    6,4,  5,4,  3,4,  2,4,  1,4,  2,4,  2,4,  3,8,  0,8,  2,4,
    2,4,  1,16,  2,4,  2,4,  3,4,  2,4,  1,4,  2,2,  3,2,  2,2,
    3,4,  0,8,  2,4,  2,4,  1,4,  1,2,  6,2,  5,8,  3,4,  2,4,
```

```
    1,4, 2,4, 3,8, 1,4, 2,4, 4,8, 4,4, 4,4, 4,8, 3,4,
    3,4, 2,8, 2,4, 1,4, 1,4, 2,8, 3,32, 2,4, 2,4, 0,8,
    2,4, 2,8, 3,4, 4,8, 4,4, 5,4, 5,4, 4,4, 3,2, 2,2,
    3,4,
    0XFF                                        //结束
};
u16 code table[] = {                            //数组存放的数据是各个音调的初始值
    0Xfc,0X8e,0Xfc,0Xed,0Xfd,0X43,0Xfd,0X6a,
    0Xfd,0Xb3,0Xfd,0Xf3,0Xfe,0X2d,0xfe,0x4e
};

unsigned int code table1[9] = {                 //这个数组里存放的是1-7的音调的初始值
    0xfc5b,0xfc8e,0xfced,0xfd43,0xfd6a,
    0xfdb3,0xfdf3,0xfe2d,0xfb68
};

/*******************************************************************************
* 函 数 名       : delay1
* 函数功能       : 延时函数，i=1时大约延时10us
* 输    入       : i
* 输    出       : 无
*******************************************************************************/
void delay1(u16 i)
{
    while (i--);
}

/*******************************************************************************
* 函 数 名       : delay
* 函数功能       : 延时函数,z=1时延时1ms
* 输    入       : z
* 输    出       : 无
*******************************************************************************/
void delay(u16 z)
{
    u16 x, y;
    for (x = z; x > 0; x--)
        for (y = 110; y > 0; y--);
}
/*******************************************************************************
* 函 数 名       : delay100ms
* 函数功能       : 延时函数作为1/4节拍
* 输    入       : z
* 输    出       : 无
```

```
*******************************************************************************/
void delay100ms(unsigned char z)
{
    unsigned int i;
    z++;
    while (--z)
    {
        for (i = 11502; i; i--);
    }
}
//readir.c
#include "reg52.h"
#include "public.h"
#include "readir.h"
/*******************************************************************************
* 函 数 名       : IrInit
* 函数功能       : 红外初始化
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void IrInit()
{
    IT0 = 1;//下降沿触发
    EX0 = 1;//打开中断0允许
    EA = 1;  //打开总中断

    IRIN = 1;//初始化端口
}
/*******************************************************************************
* 函 数 名       : ReadIr
* 函数功能       : 解码
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void ReadIr() interrupt 0
{
    u8 j, k;
    u16 err;
    Time = 0;
    delay1(700); //7ms
    if (IRIN == 0)         //确认是否真的接收到正确的信号
    {
        err = 1000;                 //1000*10us=10ms,超过说明接收到错误的信号
                                    /*当两个条件都为真是循环，如果有一个条件为假的
```

```
                                    时候跳出循环，免得程序出错的时
                                    候，程序死在这里*/
        while ((IRIN == 0) && (err > 0))    //等待前面9ms的低电平过去
        {
            delay1(1);
            err--;
        }
        if (IRIN == 1)                      //如果正确等到9ms低电平
        {
            err = 500;
            while ((IRIN == 1) && (err > 0))         //等待4.5ms的起始高电平过去
            {
                delay1(1);
                err--;
            }
            for (k = 0; k < 4; k++)         //共有4组数据
            {
                for (j = 0; j < 8; j++)     //接收一组数据
                {
                    err = 60;
                    while ((IRIN == 0) && (err > 0))//等待信号前面的560us低电平过去
                    {
                        delay1(1);
                        err--;
                    }
                    err = 500;
                    while ((IRIN == 1) && (err > 0))     //计算高电平的时间长度。
                    {
                        delay1(10);                      //0.1ms
                        Time++;
                        err--;
                        if (Time > 30)
                        {
                            return;
                        }
                    }
                    IrValue[k] >>= 1;                    //k表示第几组数据
                    if (Time >= 8)                       //如果高电平出现大于565us，那
么是1
                    {
                        IrValue[k] |= 0x80;
                    }
                    Time = 0;                            //用完时间要重新赋值
```

```
                }
            }
        }
        if (IrValue[2] != ~IrValue[3])              //判断接收的数据是否正确
        {
            return;
        }
    }
}

//song.c
#include "reg52.h"
#include "public.h"
#include "key.h"
/*******************************************************************************
* 函 数 名       : song
* 函数功能       : 自动播放歌曲
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void song()
{
    if (music > 0)
    {
        key = 0;
        if (music == 1)                             //判断按键是否按下
        {
            while (1)
            {
                KeyDown();
                key_play();
                if (key != 0)
                {
                    if (key != 9)
                    {
                        music = 0;
                    }
                    break;
                }
                TH0 = table1[music_dat[n][0]] / 256;    //赋初值
                TL0 = table1[music_dat[n][0]] % 256;
                TR0 = 1;                                //音乐开始
                delay100ms(music_dat[n][1]);            //调用延时，用做节拍的发生
                n++;                                    //下个音调开始
```

```
                if (music_dat[n][0] == 0xff)
                {
                    n = 0;                                  //判断是否到最后一个音调
                    TR0 = 0;                                //一个调放完，即将进行下一个
调
                }
            }
        }
        if (music == 4)                                     //判断按键是否按下
        {
            while (1)
            {
                KeyDown();
                key_play();
                if (key != 0)
                {
                    if (key != 9)
                    {
                        music = 0;
                    }
                    break;
                }
                TH0 = table1[music_dat1[n1][0]] / 256;      //赋初值
                TL0 = table1[music_dat1[n1][0]] % 256;
                TR0 = 1;                                    //音乐开始
                delay100ms(music_dat1[n1][1]);              //调用延时，用做节拍的发生
                n1++;                                       //下个音调开始
                if (music_dat1[n1][0] == 0xff)
                {
                    n1 = 0;                                 //判断是否到最后一个音调
                    TR0 = 0;                                //一个调放完，即将进行下
一个调
                }
            }
        }
        if (music == 3)                                     //判断按键是否按下
        {
            while (1)
            {
                KeyDown();
                key_play();
                if (key != 0)
                {
                    if (key != 9)
```

```
                {
                    music = 0;
                }
                break;
            }
            TH0 = table1[music_dat2[n2][0]] / 256;      //赋初值
            TL0 = table1[music_dat2[n2][0]] % 256;
            TR0 = 1;                                    //音乐开始
            delay100ms(music_dat2[n2][1]);             //调用延时，用做节拍的发生
            n2++;                                       //下个音调开始
            if (music_dat2[n2][0] == 0xff)
            {
                n2 = 0;                                 //判断是否到最后一个音调
                TR0 = 0;                                //一个调放完，即将进行下
一个调
            }
        }
    }
    if (music == 2)                                     //判断按键是否按下
    {
        while (1)
        {
            KeyDown();
            key_play();
            if (key != 0)
            {
                if (key != 9)
                {
                    music = 0;
                }
                break;
            }
            TH0 = table1[music_dat3[n3][0]] / 256;      //赋初值
            TL0 = table1[music_dat3[n3][0]] % 256;
            TR0 = 1;                                    //音乐开始
            delay100ms(music_dat3[n3][1]);             //调用延时，用做节拍的发生
            n3++;                                       //下个音调开始
            if (music_dat3[n3][0] == 0xff)
            {
                n3 = 0;                                 //判断是否到最后一个音调
                TR0 = 0;                                //一个调放完，即将进行下
一个调
            }
        }
```

```
        }
    }
    else
    {
        TR0 = 0;
        TR1 = 0;
        beep = 1;
    }
}

//time0.c
#include "reg52.h"
#include "public.h"
#include "time0.h"
/*******************************************************************************
* 函 数 名       : systimer0_init
* 函数功能       : 定时器0初始化
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void systimer0_init(void)
{
    TMOD |= 0x01;//设置为1时用或（|）
    TMOD &= 0xfd;//设置为0时用与（&）
    EA = 1;
    ET0 = 1;
    TR0 = 1;
}
/*******************************************************************************
* 函 数 名       : time0
* 函数功能       : 定时器0中断1程序
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void time0() interrupt 1
{
    if (music == 1)
    {
        TH0 = table1[music_dat[n][0]] / 256;        //赋初值
        TL0 = table1[music_dat[n][0]] % 256;
    }
    else if (music == 4)
    {
        TH0 = table1[music_dat1[n1][0]] / 256;      //赋初值
```

```
        TL0 = table1[music_dat1[n1][0]] % 256;
    }
    else if (music == 3)
    {
        TH0 = table1[music_dat2[n2][0]] / 256;      //赋初值
        TL0 = table1[music_dat2[n2][0]] % 256;
    }
    else if (music == 2)
    {
        TH0 = table1[music_dat3[n3][0]] / 256;      //赋初值
        TL0 = table1[music_dat3[n3][0]] % 256;
    }
    else
    {
        TH0 = temp1;
        TL0 = temp2;
    }
    beep = ~beep;                                   //不断取反得到相应的音调
}

//public.h
#ifndef          __PUBLIC_H__
#define          __PUBLIC_H__

#include<reg52.h>
//---重定义关键词---//
#ifndef u8
#define u8 unsigned char
#endif

#ifndef u16
#define u16 unsigned int
#endif

sbit beep = P2 ^ 0;           //蜂鸣器管脚
sbit sumaguan = P0;           //数码管显示
sbit IRIN = P3 ^ 2;           //红外管脚
#define GPIO_KEY P1           //矩阵键盘管脚

extern u8 IrValue[6];         //保存红外接收的数据
extern u8 Time;               //红外解码时间判定
extern u8 temp1, temp2;       //保存定时器0的定时值
extern u8 music;              //音乐播放曲目
extern u8 n;                  //读取第一首歌的节拍
```

```
extern u8 n1;                   //读取第二首歌的节拍
extern u8 n2;                   //读取第三首歌的节拍
extern u8 n3;                   //读取第四首歌的节拍
extern u8 key;                  //取数码管断码
extern u8 KeyValue;             //矩阵键盘的键值
extern u16 code smgduan[17];    //存放数码管断码0-f
extern u8 code music_dat[][2]; //存放第一首歌节拍、音调
extern u8 code music_dat1[][2]; //存放第二首歌节拍、音调
extern u8 code music_dat2[][2]; //存放第三首歌节拍、音调
extern u8 code music_dat3[][2]; //存放第四首歌节拍、音调
extern u16 code table[];        //存放1 2 3 4 5 6 7 i
extern u16 code table1[9];      ////这个数组里存放的是1-7的音调的初始值

void delay1(u16 i);
void delay(u16 z);
void delay100ms(unsigned char z);
#endif

//key.h
#ifndef          __KEY_H__
#define          __KEY_H__
void key_play();
void KeyDown();
#endif

//readir.h
#ifndef          __READIR_H__
#define          __READIR_H__
void IrInit();
void ReadIr();
#endif

//song.h
#ifndef          __SONG_H__
#define          __SONG_H__
void song();
#endif

//time0.h
#ifndef          __TIME0_H__
#define          __TIME0_H__
void systimer0_init(void);
void time0();
#endif
```