福建師範大學
FUJIAN NORMAL UNIVERSITY

二〇一八届本科毕业论文（设计）报告

题目：桌面天气预报系统设计

学生姓名：李正念

所在学院：物理与能源学院

学　　号：135032014207

专　　业：新能源科学与工程

指导教师：何志杰

目录

桌面天气预报系统设计

物理与能源学院　新能源科学与工程专业
135032014207　李正念　指导教师：何志杰

摘要：对天气进行预报具有很大的现实意义。本系统采用 STM32 单片机为主控芯片，通过 WiFi 模块 GET 天气 API 接口，获取天气预报数据。控制器把获取到的天气数据通过串口定时发送至串口屏，结合优美的 GUI 界面，就构成了一套炫丽的桌面天气预报系统。本系统具有触摸屏搜索与语音搜索天气的功能，可以搜索各个城市的天气数据并更新至串口屏显示。除此之外，本系统还具有如下几个功能：WiFi 配置功能、空气质量检测、收音机功能、语音识别功能、实时显示时间。

关键词：天气预报；WiFi；串口屏；收音机；语音识别

1 引言

1.1 课题研究背景

1.1.1 对天气进行预报的必要性

对于天气的预测在现实生活中具有重大意义，涉及的领域很多。智能硬件方面，无论是智能家居产品还是可穿戴设备，都能接入天气数据，贴近用户生活场景、优化使用体验。能源方面，长期对气象进行预测可为传统的能源提供一定的需求预测和生产指导。移动物联网方面，移动端用户对于天气资讯有普遍需求，引入天气可以贴近生活场景，提升用户体验[1]。

1.1.2 智能家居的趋势

随着智能技术的快速发展，智能家居已经成为必然趋势，并且国人的生活的水平不断的提高，对于智能家居的追求也越来越强烈[2]。桌面天气预报系统也属于智能家居中的一种，可以实时显示当地的天气信息。例如现在有一种智能浴镜，集成了逐时天气预报数据，用户在早晨洗漱时可以随时浏览各地新闻，了解当日天气的变化，为出行做好准备。

1.2 课题任务

本系统采用 STM32F103RET6 为主控芯片控制所需的外围功能模块来完成特定的功能，因此需要编写相应的驱动程序与应用程序。具体的任务为：（1）编写 ESP8266 WIFI 模块的基本驱动，使其能连上 WiFi 网络并连接得上国内提供部分数据供个人用户免费使用的的天气服务器——心知天气，并从中获取天气数据；（2）编写串口屏所需驱动以及制定一些自主的传输数据的协议使得串口屏能与 STM32 单片机进行准确地通信，使得串口屏能正常显示；（3）编写 TEA5767 收音机模块驱动，使其能采集得到所设置频率的音频信号并通过耳机或者扬声器播报出来，并且能通过液晶触摸屏调节频率；（4）编写 LD3320 语音识别模块驱动，使其能够正确识别所设定的语音关键词，并通过识别得到的关键词搜索天

气信息；（5）通过 DHT11 温湿度传感器采集当地温湿度信息并更新至 HMI 智能串口触摸屏进行显示；（6）通过粉尘传感器测量当地的 PM2.5 的值并发送至 HMI 智能串口触摸屏进行显示；（7）编写系统的整体应用程序以及做好系统各模块之间的协调工作，使得系统各模块之间都能稳定的运行。

2 系统总设计

2.1 系统框图

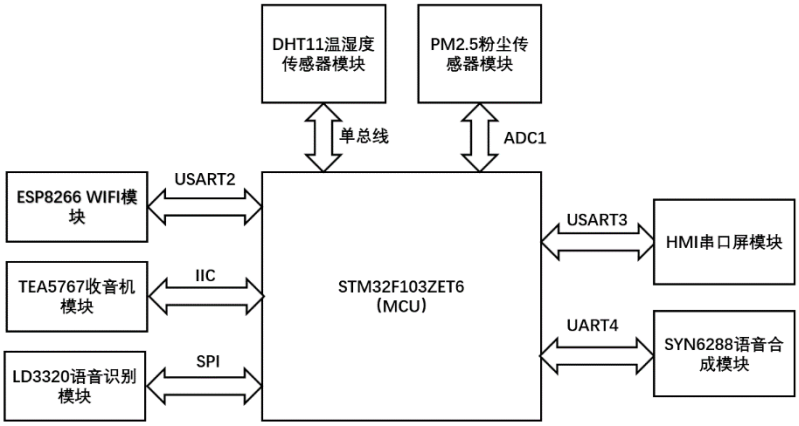

图 2-1 系统框图

图 2-1 为系统框图。整个系统的外围硬件模块主要有：DHT11 模块、粉尘传感器模块、WiFi 模块、TEA5767 收音机模块、LD3320 语音识别模块、HMI 串口屏模块、SYN6288 语音合成模块。

2.2 系统概述

系统上电，如果处于系统默认的 WiFi 环境，则无需设置 WiFi。若处于新的 WiFi 环境，则需要通过液晶触摸屏的 WiFi 设置界面设置 WiFi，WiFi 设置界面如图 2-2 所示：

图 2-2 WiFi 设置界面

并通过点击“Add”按钮将输入好的 WiFi 名称与密码通过串口发送给 MCU，并自动返回串口屏的主界面。MCU 收到 WiFi 信息后将开始连接 WiFi，我们可以观察 WiFi 指示灯是否闪烁完毕来判断 WiFi 是否连接完成，也可以查看串口屏上 WiFi 状态图标是否处于有信号的状态。等待大约 5s，WiFi 连接成功。WiFi 连接成功首先会获取一次网络时间，并用于设置 RTC 产生的时间，然后再把校准好的时间更新至串口屏进行显示。此时，可以通过串口屏主界面的搜索框进行搜索各城市的天气预报信息或者用语音进行搜索。开机未获取天气数据、未校准时间数据的初始主界面如图 2-3 所示：

图 2-3 串口屏初始的主界面

搜索的方式有两种，即支持中文搜索也支持英文搜索，键盘界面如图 2-4 所示：

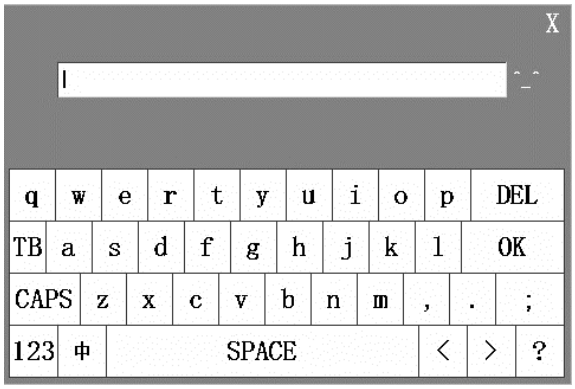

图 2-4 键盘界面

图 2-4 中按钮“中”为输入法切换按钮，即刚开始是中文输入状态，点击“中”按钮时切换为英文输入状态，再点击一次时，又将变回中文输入状态，以此类推。除此之外，点击图 2-3 主界面的“心知天气”四个字所在的屏幕区域将可进入菜单界面。菜单界面如图 2-5 所示：

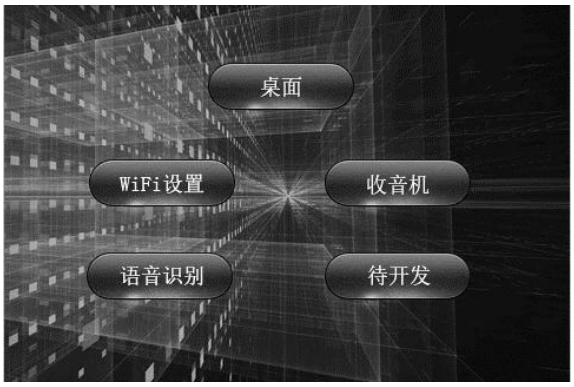

图 2-5 菜单界面

此时，“收音机”按钮可进入收音机频率设置界面。收音机频率设置界面如图 2-6 所示：

图 2-6 收音机频率设置界面

点击“<”、“>”键分别可以增大、减小频率。点击“80.0”、“85.0”、“90.0”、“95.0”、“100.0”、“105.0”这几个按钮时，频率值立刻变为所点的按钮上显示的值。设计这几个按钮的目的是为了避免特别频繁的按“<”按钮或者“>”按钮来调节频率。比如当前显示的频率是“80.1MHz”，若我们想要调节至“107.1MHz”时，就不需要通过一直点击“>”按钮来调节频率，只需要先点击“105.0”这个按钮，再在这个基础上点击少量次数的“>”按钮，即可达到快速调节频率的效果。调节好频率之后，点击“确定”按钮即可向单片机发送设置好的频率。单片机收到频率后即可设置 TEA5757 收音机模块的收音机频率然后开始接收音频信号，再通过音频放大器 TDA1308 放大之后即可通过耳机或者音箱播放。

3 系统硬件介绍

3.1 主控模块

本系统主控芯片采用的是 STM32F103ZET6，其资源丰富。其引脚图如图 3-1 所示：

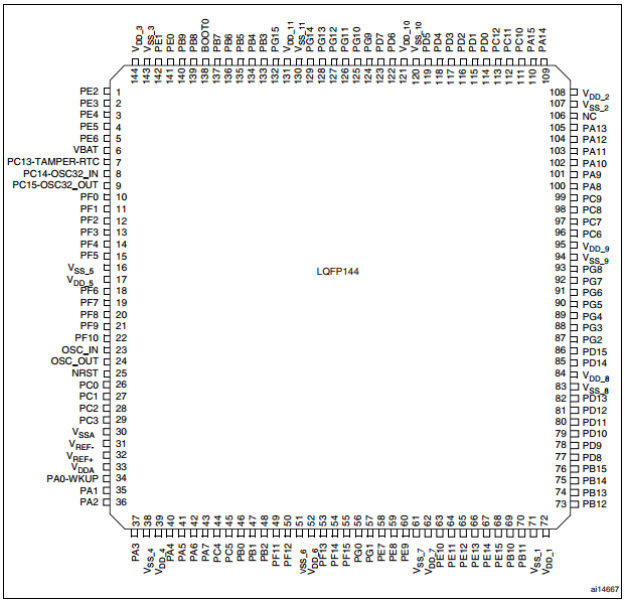

图 3-1 主控芯片引脚图

本设计使用到的资源如表 1 所示：

表 1 本设计使用的主控资源

资源	用途
串口 1	用于调试
串口 2	供 ESP8266 WiFi 模块使用
串口 3	供 HMI 智能串口屏使用
串口 4	用于与 SYN6288 语音合成模块通信
RTC 时钟	用于实现日历功能
定时器 1	虚拟出多个定时器用于协调系统任务
IIC1	用于与 TEA5767 收音机模块通信
单总线	用于与 DHT11 通信
SPI3	用于与 LD3320 语音识别模块通信
ADC1 通道 1	用于采集 PM2.5 值

其最小系统原理图如图 3-2 所示：

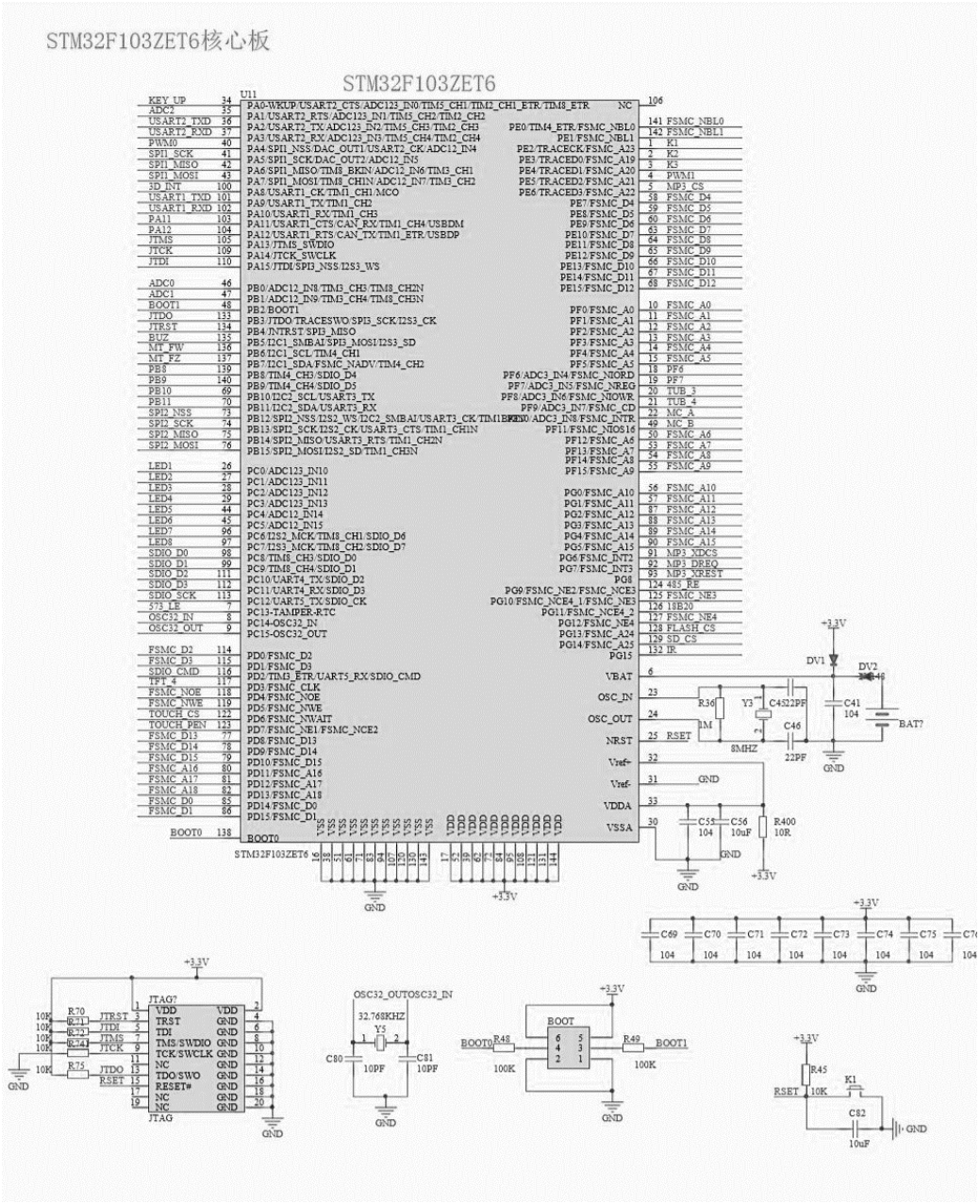

图 3-2 STM32 最小系统原理图

3.2 WiFi 模块

本系统 WiFi 模块采用的是 ESP8266 WiFi 模块，通过 STM32 的串口向其发送 AT 指令进行一些基本的配置即可使用。该 ESP8266 是串口型 WiFi，速度相对较低，虽然不能用来传输图像等容量比较数据，但也能一次性稳定传输几千字节的数据，用其来传输天气数据是绰绰有余的[3]。WiFi 模块的 PCB 图如图 3-3 所示：

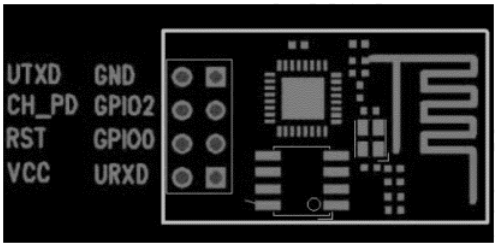

图 3-3 WiFi 模块的 PCB 图

3.3 显示模块

本系统显示模块采用的是串口液晶触摸屏。USART 屏是一种较简单的显示方案，其显示速度很快，因为界面的显示是屏幕内部自己实现的，用户主控参与的只是发送一些指令。再次，界面的布局也全都不需要用户的 MCU 来参与，使用“USART HMI”上位软件使用拖控件地形式可以快速布局。

3.4 收音机模块

本系统收音机模块采用的是 TEA5767 收音机模块，其频率范围从 76—108MHZ 自动数字调谐，其内部集成了中频选频和解调网络，可以做到完全免调[4]。模块实物图如图 3-4 所示：

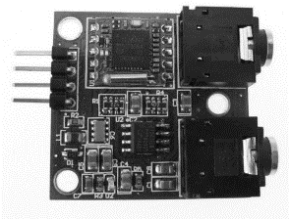

图 3-4 收音机模块实物图

3.5 语音识别模块

本系统语音识别模块采用的语音芯片是 LD3320。该芯片已经集成了语音识别的处理器，不需要外接其他的辅助芯片，直接嵌入在产品中就可以实现语音识别的功能[5]。其与单片机通信方式有并行与 SPI 两种方式。其实物图如图 3-5 所示：

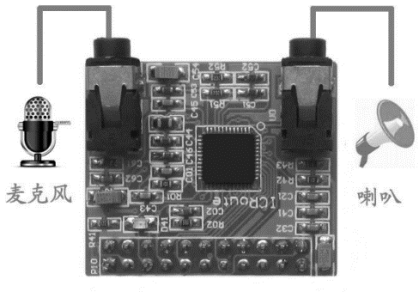

图 3-5 语音识别模块实物图

3.6 粉尘传感器模块

本系统粉尘传感器模块采用的是 GP2Y1010 模块。其内部结构原理图如图 3-6 所示：

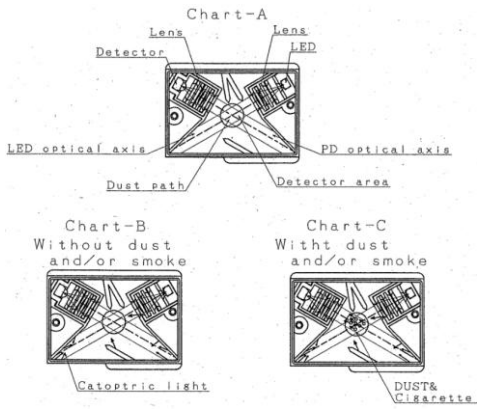

图 3-6 粉尘传感器内部结构图

GP2Y 模块检测原理：其中心有个洞可以让空气自由地流过，定向地发射 LED 光，通过检测经过的空气中的灰尘折射过后的光线来判断灰尘的含量[6]。其采样时序及输出脉冲时序如图 3-7 所示：

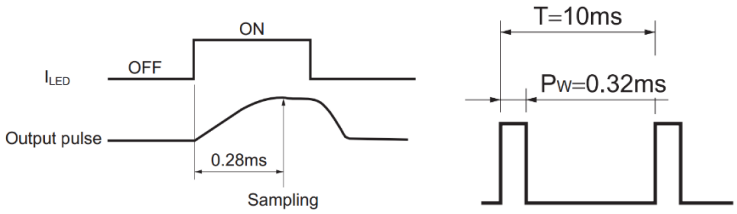

图 3-7 粉尘传感器采样时序及脉冲时序

采样时，需要开启内部的发光二极管并保持 280 微秒，然后再读取其输出值。并且，因为整个脉冲持续的时间为 320 微秒，所以还需等待 40 微秒发光二极管才能熄灭[7]。

3.7 温湿度传感器模块

本系统温湿度传感器采用的是 DHT11 模块。DHT11 是一款湿温度数字传感器[8]，其与单片机的通信方式为单总线通信。

3.8 语音合成模块

本系统语音合成模块采用的是 SYN6288 模块。SYN6288 通过 UART 与单片机通信，接收单片机发送的文本数据，就可以实现文本到语音的转换。其工作流程框图如图 3-8 所示：

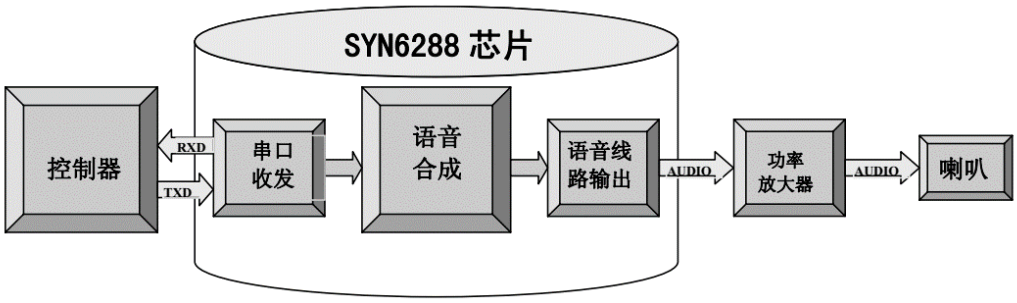

图 3-8 语音合成流程框图

4 系统软件设计

本系统软件总体框架采用分时多任务的方式对各个模块的各个任务进行协调，使得系统更为稳定。这里说的分时多任务即使用一个定时器来虚拟出多个定时器来供给各个任务定时使用[9]，当某个虚拟定时器的计数值递减至 0 时表明已达到定时的时间。代码层面的设计如图 4-1 所示：

```
/*********************************************************************************************************
** 函数: TIM1_IRQHandler,  定时器1中断服务程序
**-------------------------------------------------------------------------------------------------------
** 参数: 无
** 返回: 无
** 作者: 2017.12.23 by Hezhijie and Lizhengnian
*********************************************************************************************************/
void TIM1_UP_IRQHandler(void)   //TIM1中断
{
    uint8 i;

    if (TIM_GetITStatus(TIM1, TIM_IT_Update) != RESET)  //检查TIM1更新中断发生与否
    {
        //-----------------------------------------------------------------------------
        // 各种定时间器计时
        for (i = 0; i < MAX_TIMER; i++)                 // 定时时间递减
        if( g_Timer1[i] ) g_Timer1[i]-- ;
        TIM_ClearITPendingBit(TIM1, TIM_IT_Update  );   //清除TIMx更新中断标志
    }
}
```

```
#define  MAX_TIMER          5                       // 最大定时器个数
EXT volatile unsigned long  g_Timer1[MAX_TIMER];
#define  GetWeatherTimer    g_Timer1[0]             // 定时GET天气数据包
#define  SendWeatherTimer   g_Timer1[1]             // 定时发送天气数据至串口屏
#define  GetHMIDataTimer    g_Timer1[2]             // 定时获取串口屏下发的数据
#define  TimeCalibraTimer   g_Timer1[3]             // 定时校准时间
#define  DHT11CollectTimer  g_Timer1[4]             // 采集温湿度数据

#define  TIMER1_SEC         (1)                     // 秒
#define  TIMER1_MIN         (TIMER1_SEC*60)         // 分
#define  TIMER1_HOUR        (TIMER1_MIN*60)         // 时
#define  TIMER1_DAY         (TIMER1_HOUR*24)        // 天
```

图 4-1 虚拟定时器的定义

首先，进行的是系统的初始化，包括 RTC、串口屏、WiFi、定时器、收音机、DHT11、语音识别的初始化等。代码层面的设计如图 4-2 所示：

```
/*********************************************************************************************************
** 函数: SysInit，上电初始化
**--------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 无
** 作者: 2017.12.23 by Hezhijie and Lizhengnian
*********************************************************************************************************/
void SysInit(void)
{
//  CpuInit();                                          // 配置系统信息
    SysTick_Init(72);                                   // 初始化系统时钟
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_3);     // 选择第3组的中断优先级
    HMI_USART3_init(9600);                              // 串口屏初始化,默认波特率为9600!!
    my_mem_init(SRAMIN);                                // 初始化内部内存池
    USART1_Init(115200);                                // 串口1初始化
    PrintfInfo();                                       // 打印工程标题
    RTC_Init();                                         // RTC初始化
    ESP8266_USART2_Init(115200);                        // ESP8266 WIFI初始化,默认波特率为115200!!
    TIM1_Init(2000-1, 36000-1);                         // 1s溢出一次
    RadioInit();                                        // 收音机初始化
    DHT11_Init();                                       // DHT11温湿度传感器GPIO初始化
//  SD_CardInit();                                      // 初始化SD_Card模块
    LD3320_Init();                                      // 初始化语音识别模块
    InitSuccess();                                      // 初始化结束
}
```

图 4-2 上电初始化

其次，分别设计几个任务用于完成系统功能。如任务 1 是接收串口屏下发的数据、任务 2 是获取天气预报数据、任务 3 是更新天气数据至串口屏、任务 4 是更新天气图标、任务 5 是时间校准、任务 6 是采集温湿度数据、任务 7 是语音识别。主流程图如图 4-3 所示：

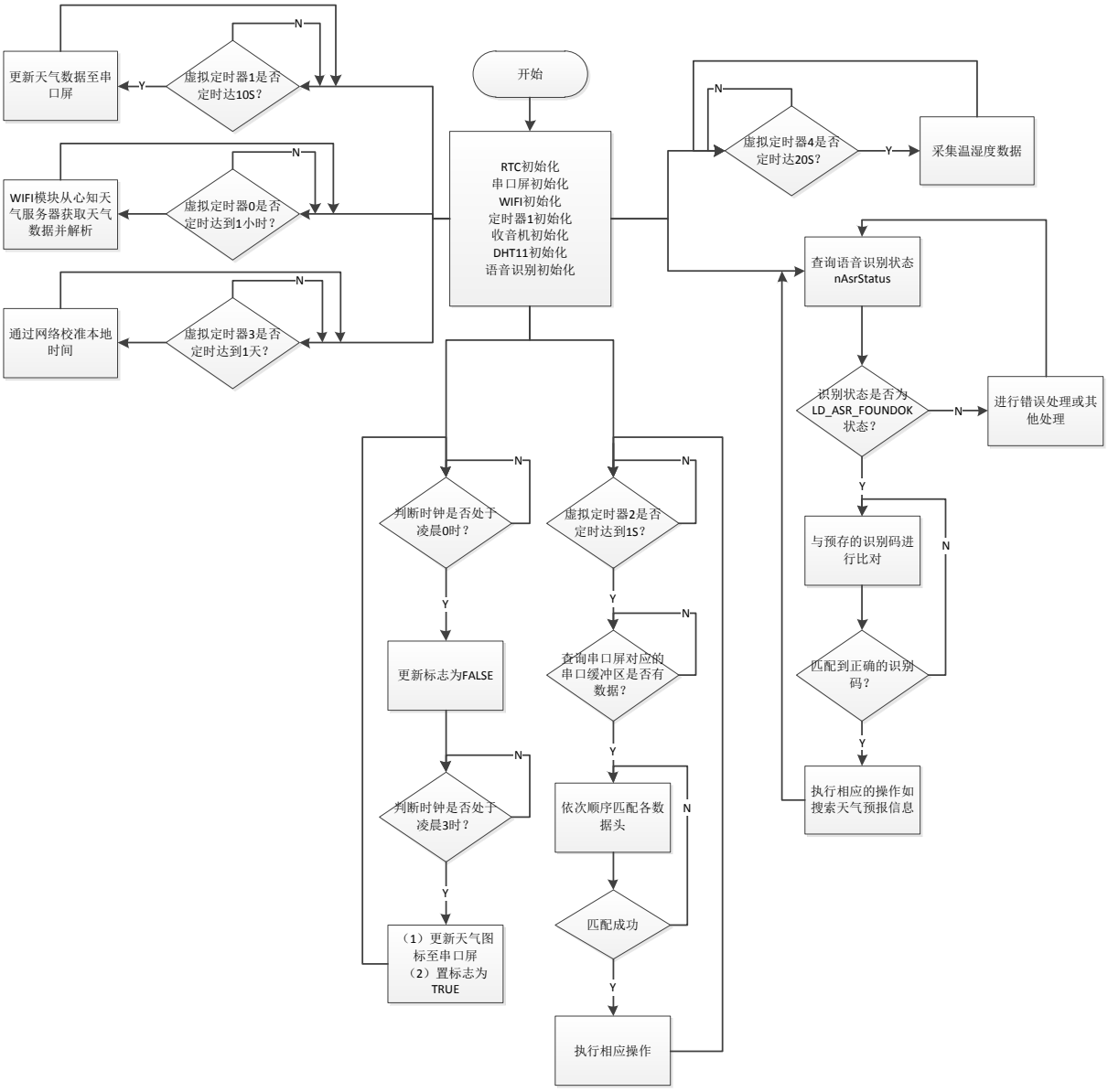

图 4-3 主流程图

4.1 驱动层

4.1.1 串口屏模块相关驱动函数

因为串口屏接收指令的结束符为“0xff 0xff 0xff”三个字节，所以需要构造一个函数使其可以往串口依次连续地发送三个字节的结束符以控制串口屏。该函数如图 4-4 所示：

```
/*********************************************************************************************************
** 函数: HMISendByte，往串口发送一个3个字节数据（供串口屏使用）
**--------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 无
** 作者: 2017.12.8 by Hezhijie and Lizhengnian
*********************************************************************************************************/
void HMISendByte(u8 byte)
{
    u8 i;

    for(i=0;i<3;i++)
    {
        if(byte!=0)
        {
            USART_SendData(USART3, byte);                                //发送数据
            while(USART_GetFlagStatus(USART3, USART_FLAG_TXE) == RESET){} //等待发送结束
        }
        else
            return;
    }
}
```

图 4-4 串口发送 3 字节数据函数

串口屏的操作指令是字符串形式的，所以需要构造一个函数使其可以往串口发送一串字符串以控制串口屏。该函数如图 4-5 所示：

```
/*********************************************************************************************************
** 函数: HMISendStr，向串口屏发送字符串
**--------------------------------------------------------------------------------------------------------
** 参数: buf
** 返回: 无
** 作者: 2017.12.8 by Hezhijie and Lizhengnian
*********************************************************************************************************/
void HMISendStr(char *buf)
{
    u8 i=0;

    while(1)
    {
        if(buf[i]!=0)
        {
            USART_SendData(USART3, buf[i]);
            while(USART_GetFlagStatus(USART3, USART_FLAG_TXE) == RESET){}
            i++;
        }
        else
            return;
    }
}
```

图 4-5 串口发送字符串函数

所以，操作串口屏的基本调用为以下两条语句：“HMISendStr(pbuf);”、 “HMISendByte(0xff);”。其中“pbuf”为字符串指令，“0xff”为结束符的重复的单位。

以上讨论的是单片机向串口屏发送数据，接下来讨论单片机接收串口屏数据，接收采用的是串口中断来接收串口屏下发的数据。该串口中断函数的设计思路如下：定义一个全局变量 USART_RX_STA 用于保存接收的状态。因为该变量在这里起到与寄存器一样的作用，所以称其为状态寄存器。其具体的位表示如图 4-6 所示：

USART_RX_STA		
bit15	bit14	bit13~0
接收完成标志	接收到 0X0D 标志	接收到的有效数据个数

图 4-6 状态寄存器位定义

定义一个数组 USART_RX_BUF，用于接收串口屏下发的数据，同时在状态寄存器中计数有效数据个数。我们都知道回车由 0x0d 和 0x0a 组成，当收到 0x0d 时，计数器停止计数，等待 0x0a 的到来，如果没有收到 0x0a，则此次接收失败，开始下一次接收。如果顺利接收到 0x0a，则标记状态寄存器的第 15 位，此时完成一次接收，等待该位被清除，从而开始下一次的接收，而如果迟迟没有收到 0x0d，那么在接收数据超过数组长度的时候，则会丢弃前面的数据，重新接收[10]。该中断函数的代码实现如图 4-7 所示：

```
void USART3_IRQHandler(void)
{
    u8 rec;
    if(USART_GetITStatus(USART3, USART_IT_RXNE) != RESET)  //接收中断(接收到的数据必须是0x0d 0x0a结尾)
    {
        rec = USART_ReceiveData(USART3);//(USART3->DR)，读取接收到的数据
        if((USART3_RX_STA&0x8000)==0)//接收未完成
        {
            if(USART3_RX_STA&0x4000)//接收到了0x0d
            {
                if(rec!=0x0a)USART3_RX_STA = 0;  //接收错误,重新开始
                else USART3_RX_STA |= 0x8000;    //接收完成了
            }
            else //还没收到0X0d
            {
                if(rec==0x0d)USART3_RX_STA|=0x4000;
                else
                {
                    USART3_RX_BUF[USART3_RX_STA&0X3FFF] = rec;
                    USART3_RX_STA++;
                    if(USART3_RX_STA>(USART3_REC_LEN-1))USART3_RX_STA=0;//接收数据错误,重新开始接收
                }
            }
        }
    }
}
```

图 4-7 串口中断函数

4.1.2 ESP8266 WiFi 模块相关驱动函数

ESP8266 的配置是通过串口发送 AT 指令来配置，所以需要封装一个可以发送 AT 指令的函数，其函数体如图 4-8 所示：

```
uint8 ESP8266_SendATCmd(uint8 *cmd, uint8 *ack, uint16 wait_time)
{
    uint8 res = 0;
    USART2_RX_STA = 0;

    ESP8266_Printf("%s\r\n", cmd);
    if(ack&&wait_time)
    {
        while(--wait_time)
        {
            delay_ms(10);
            if(USART2_RX_STA&0x8000) //接收到回车表明接收数据结束
            {
                if(ESP8266_CheckAck(ack))
                {
                    printf("%s ack=%s\n",cmd, ack);
                    USART2_RX_STA = 0;
                    break; //得到有效数据，退出while循环
                }
                USART2_RX_STA = 0;
            }
        }
        if(wait_time==0)  //超时
        {
            res = 1;      //失败
        }
    }

    return res;
}
```

图 4-8 AT 指令发送函数

4.2 应用层

4.2.1 获取天气数据

获取天气数据的过程为：连接天气服务——>WiFi 模块配置为透传模式——>发送 http 申请包 GET 天气数据——>获得天气数据——>关闭透传模式。具体代码如图 4-9 所示：

```
/************************************************************************************************************
** 函数: GET_NowWeather, GET 天气实况数据包now.json
**------------------------------------------------------------------------------------------------------------
** 参数: void
** 说明: 数据来源：心知天气（api.seniverse.com）
** 返回: 0:获取成功 其他：获取失败
** 作者: 2017.12.8 by Hezhijie and Lizhengnian
************************************************************************************************************/
uint8 GET_NowWeather(void)
{
    ESP8266_LinkServer(enumTCP, (uint8*)WEATHER_IP, (uint8*)WEATHER_PORT);
    delay_ms(300);
    ESP8266_SendATCmd("AT+CIPMODE=1","OK",100);      //传输模式为：透传
    USART2_RX_STA=0;
    ESP8266_SendATCmd("AT+CIPSEND","OK",100);         //开始透传
    ESP8266_Printf("GET https://api.seniverse.com/v3/weather/now.json?"
    "key=2owqvhhd2dd9o9f9&location=%s&language=zh-Hans&unit=c\r\n\r\n",g_city); //不要忘记\r\n\r\n,参数language、uint为可选参数
    delay_ms(20);//延时20ms返回的是指令发送成功的状态
    USART2_RX_STA=0;    //清零串口2数据
    delay_ms(1000);
    if(USART2_RX_STA&0X8000)        //此时再次接到一次数据，为天气的数据
    {
        USART2_RX_BUF[USART2_RX_STA&0X7FFF]=0;//添加结束符
    }
    cJSON_NowWeatherParse((char*)USART2_RX_BUF, &weather_data);
    ESP8266_ExitTran();//退出透传
    ESP8266_SendATCmd("AT+CIPCLOSE","OK",50);         //关闭连接

    return 0;
}
```

图 4-9 获取天气数据

4.2.2 解析天气数据

天气数据包的格式为 JSON 格式，采用 cJSON 库对其进行解析。解析的步骤为：

（1）第一步：调用 cJSON_Parse 函数来解析 JSON 格式数据包，返回“NULL”表示解析失败，否则解析成功，并把解析成功地数据保存在 cJSON 类型的变量中。

（2）第二步：调用 cJSON_GetObjectItem 函数匹配节点名称，匹配成功则把该节点保存到 cJSON 类型变量中。若数据包中有数组结构，则首先调用 cJSON_GetArraySize 获取数组中的对象个数，再调用 cJSON_GetArrayItem 依次获取各对象内容，然后再调用 cJSON_GetObjectItem 匹配对象节点。

（3）第三步：从 cJSON 结构体中的 valueint 或 valuestring 成员取出键值。

（4）第四步：调用 cJSON_Delete 函数释放内存空间[11]。

需要注意的是问题是：在（1）中使用 cJSON_Parse 函数会在内存中开辟一个空间，使用完毕需要手动释放。在一般的单片机编程中尽量不要使用 C 语言标准库里的 malloc 与 free 函数，因为一般的单片机的内存都比较小，而且没有 MMU 内存管理单元，使用 C 语言标准库里的 malloc 可能会产生大量内存碎片从而导致系统崩溃，所以需要自己实现内存管理。如果是大的内存分配，而且 malloc 与 free 的次数也不是特别频繁，使用 malloc 与 free 是比较合适的，但是如果内存分配比较小，而且次数特别频繁，那么使用 malloc 与 free 就有些不太合适了 [12]。此处应用在获取天气数据，会不断从服务器下载天气数据，当达到一定次数的时候就会导致内存碎片过多从而导致系统崩溃，所以在这里不适合使用 C 标准库里的 malloc。图 4-10 是解析 JSON 数据的代码片段：

```
if((child_Item = cJSON_GetObjectItem(sub_child_object,"date")) != NULL)//日期
{
    memcpy(result->date[i], child_Item->valuestring, strlen(child_Item->valuestring));
    ParseDate(result->date[i], &g_date[i]);          //解析日期得到年月日
    g_week[i] = RTC_GetWeek(g_date[i].date_year, g_date[i].date_month, g_date[i].date_day); //获得星期
    printf("daliy[%d]--%s:%s(After parse:%d %d %d %s)\n",i ,child_Item->string, child_Item->valuestring,
    g_date[i].date_year, g_date[i].date_month, g_date[i].date_day,wday[g_week[i]]);
}
memset(utf8str, 0, 64);
memset(gbkstr, 0, 64);
if((child_Item = cJSON_GetObjectItem(sub_child_object,"text_day")) != NULL)//白天天气现象文字
{
    utf8str = child_Item->valuestring;
    SwitchToGbk((const unsigned char*)utf8str, strlen(utf8str), (unsigned char*)gbkstr, &gbkstr_len);
    printf("daliy[%d]--%s:%s\n",i ,child_Item->string, gbkstr);
    memcpy(result->text_day[i], gbkstr, strlen(gbkstr));
}
if((child_Item = cJSON_GetObjectItem(sub_child_object,"code_day")) != NULL)//白天天气现象代码
{
    printf("daliy[%d]--%s:%s\n",i ,child_Item->string, child_Item->valuestring);
    memcpy(result->code_day[i], child_Item->valuestring, strlen(child_Item->valuestring));
}
if((child_Item = cJSON_GetObjectItem(sub_child_object,"code_night")) != NULL)//夜间天气现象代码
{
    printf("daliy[%d]--%s:%s\n",i ,child_Item->string, child_Item->valuestring);
    memcpy(result->code_night[i], child_Item->valuestring, strlen(child_Item->valuestring));
}
if((child_Item = cJSON_GetObjectItem(sub_child_object,"high")) != NULL)//最高温度
{
    printf("daliy[%d]--%s:%s\n",i ,child_Item->string, child_Item->valuestring);
    memcpy(result->high[i], child_Item->valuestring, strlen(child_Item->valuestring));
}
if((child_Item = cJSON_GetObjectItem(sub_child_object,"low")) != NULL)//最低温度
{
    printf("daliy[%d]--%s:%s\n",i ,child_Item->string, child_Item->valuestring);
    memcpy(result->low[i], child_Item->valuestring, strlen(child_Item->valuestring));
}
```

图 4-10 解析天气数据

4.2.3 后台任务

图 4-11 是后台任务入口，这些任务的功能为：接收串口屏下发的数据、获取天气预报数据、更新天气数据至串口屏、更新天气图标、时间校准、采集温湿度数据、语音识别。

```
/*************************************************************************************************************
** 函数: BackTask，后台任务
**-------------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 无
** 作者: 2017.12.23 by Hezhijie and Lizhengnian
*************************************************************************************************************/
void BackTask(void)
{
    Task_RecHMIDate();      // 接收串口屏下发的数据
    Task_GetWeather();      // 获取天气预报数据
    Task_DisplayWeather();  // 更新天气数据至串口屏
    Task_UpdateIcon();      // 更新天气图标
    Task_TimeCalibration(); // 时间校准
    Task_DHT11Collect();    // 采集温湿度数据
    Task_ASR();             // 语音识别
}
```

图 4-11 后台任务

（1）任务一：接收串口屏下发的数据代码片段

```
/* 地名 */
else if((p=strstr((char*)USART3_RX_BUF,"place:"))!=NULL)
{
    if(sscanf(p,"place:%s",g_place)==1)
    {
        if(strcmp(g_serch_method,"China")==0)
        {
            SwithToUtf_8((const unsigned char*)g_place, strlen((const char*)g_place),(unsigned char*) g_place_utf8, &utf8_len);
            printf("place:%s\n",g_place_utf8);
            printf("place:%X%X%X %X%X%X\n",g_place_utf8[0],g_place_utf8[1],g_place_utf8[2],
            g_place_utf8[3],g_place_utf8[4],g_place_utf8[5]);
            sprintf(g_city,"%%%X%%%X%%%X%%%X%%%X%%%X",g_place_utf8[0],g_place_utf8[1],
            g_place_utf8[2],g_place_utf8[3],g_place_utf8[4],g_place_utf8[5]);
        }
        else if(strcmp(g_serch_method,"English")==0)
        {
            printf("place:%s\n",g_place);
            memcpy(g_city,g_place,sizeof(g_place));
        }
        memset(&weather_data, 0, sizeof(weather_data));
        GET_NowWeather();
        GET_DailyWeather();
        GetWeatherTimer = TIMER1_HOUR;
        TTSPlay(0, "[t3][2]正在为您显示天气信息！");
        DisplayWeather(weather_data);
        DisplayWeatherIcon(weather_data);

    }
}
```

图 4-12 接收并处理串口屏下发数据

接收串口屏下发数据的函数如图 4-12 所示，其中接收的数据包括城市名、wifi 名称密码、收音机频率、中英文切换、页面标识等信息。

（2）任务二：获取天气预报数据

```
/*********************************************************************************************************
** 函数: TaskGetWeather，后台任务--获取天气数据,每一个小时抓一次天气预报数据包
**-------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 1:成功 0:失败
** 作者: 2017.12.23 by Hezhijie and Lizhengnian
*********************************************************************************************************/
static void Task_GetWeather(void)
{
    if( GetWeatherTimer ) return;
    GetWeatherTimer = TIMER1_HOUR;
    memset(&weather_data, 0, sizeof(weather_data));
    GET_NowWeather();
    GET_DailyWeather();
}
```

图 4-13 获取天气预报数据

获取天气数据的函数如图 4-13 所示。虚拟定时器 GetWeatherTimer 定时每 1 小时抓取一次天气数据包，每一次抓包之前都先清空接收缓存，否则可能会导致数据覆盖的问题。

（3）任务三：更新天气数据至串口屏

```
/*********************************************************************************************************
** 函数: Task_DisplayWeather，定时更新天气数据至串口屏
**-------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 1:成功 0:失败
** 作者: 2017.12.23 by Hezhijie and Lizhengnian
*********************************************************************************************************/
static void Task_DisplayWeather(void)
{
    if(SendWeatherTimer) return;
    SendWeatherTimer = 10*TIMER1_SEC;
    DisplayWeather(weather_data);
}
```

图 4-14 更新天气数据至串口屏

更新天气数据至串口屏的函数如图 4-14 所示。虚拟定时器 SendWeatherTimer 定时每 10S 更新一次天气数据至串口屏。

（4）任务四：更新天气图标

```
static void Task_UpdateIcon(void)
{
    if( 0==calendar_t.hour )
    {
        WeatherIconFlag = FALSE;
    }
    else if( (3==calendar_t.hour) && (FALSE==WeatherIconFlag) )
    {
        DisplayWeatherIcon(weather_data);
        WeatherIconFlag = TRUE;
    }
}
```

图 4-15 更新天气图标

更新天气图标的函数如图 4-15 所示。调试过程中需要一个问题：在切换串口屏页面时，天气图标可能会相应地被更新至切换的界面。程序上的解决思路为：判断当前时间为凌晨 3 点至 4 点间并且处于主界面，并且 3 点到 4 点的更新标志为 FALSE，则执行更新图标操作。更新完之后，3 点到 4 点的更新标志改为 TRUE，即使还在这个时间段也不会更新。主程序可以在凌晨 0 点左右修改 3 点至 4 点的更新标志为 FLASE，即可启动下一次更新。

（5）任务五:时间校准

```
/*********************************************************************************************************
** 函数: Task_TimeCalibration，时间校准，一天校准一次
**-------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 1:成功 0:失败
** 作者: 2018.1.1 by Hezhijie and Lizhengnian
*********************************************************************************************************/
static void Task_TimeCalibration(void)
{
    if(TimeCalibraTimer) return;
    TimeCalibraTimer = TIMER1_DAY;
    GET_BeiJingTime();
}
```

图 4-16 时间校准

时间校准函数如图 4-16 所示。虚拟定时器 TimeCalibraTimer 定时从此刻开始之后每一天通过网络校准一次本地时间。

（6）任务六：DHT11 温湿度采集

```
/*********************************************************************************************************
** 函数: Task_DHT11Collect，DHT11温湿度数据采集，20s采集一次
**-------------------------------------------------------------------------------------------------------
** 参数: void
** 返回: 1:成功 0:失败
** 作者: 2018.1.1 by Hezhijie and Lizhengnian
*********************************************************************************************************/
uint8 temperature=0,humidity=0;
static void Task_DHT11Collect(void)
{
    if(DHT11CollectTimer) return;
    DHT11CollectTimer = 20*TIMER1_SEC;
    DHT11_Read_Data(&temperature, &humidity);
//  printf("temperatur:%d;humidity:%d\n",temperature,humidity);
}
```

图 4-17 DHT11 温湿度采集

温湿度数据采集的函数如图 4-17 所示。虚拟定时器 DHT11CollectTimer 定时 20S 采集一次温湿度数据，并把采集得到的温度、湿度数据分别保存在变量“temperature”、“humidity”中。

（7）任务七：语音识别任务

```
static void Task_ASR(void)
{
    switch(nAsrStatus)
    {
        case LD_ASR_RUNING:
        case LD_ASR_ERROR:
            break;
        case LD_ASR_NONE:
            nAsrStatus=LD_ASR_RUNING;
            if (RunASR()==0)    //  启动一次ASR识别流程：ASR初始化，ASR添加关键词语，启动ASR运算
            {
                nAsrStatus = LD_ASR_ERROR;
            }
            break;

        case LD_ASR_FOUNDOK:
            nAsrRes = LD_GetResult( );  //  一次ASR识别流程结束，去取ASR识别结果
            ASRSuccess_Handle(nAsrRes);
            nAsrStatus = LD_ASR_NONE;
        break;

        case LD_ASR_FOUNDZERO:
        default:
            nAsrStatus = LD_ASR_NONE;
            break;
    }
}
```

图 4-18 语音识别任务

语音识别任务的函数片段如图 4-18 所示。其中 nAsrStatus 是用来表示语音识别的状态，不是 LD3320 芯片内部的状态寄存器[13]。nAsrStatus 有几种情况。我们比较关注的是 LD_ASR_FOUNDOK 状态。LD_ASR_FOUNDOK 状态为识别成功，识别成功后将调用“ASRSuccess_Handle”函数进行识别后的操作。语音识别成功后的操作函数的代码片段如图 4-29 所示：

```
/************************************************************************************************
** 函数: ASRSuccess_Handle，识别成功，执行相应操作
**------------------------------------------------------------------------------------------------
** 参数: asr_code: 识别码
** 返回: 无
** 作者: 2018.1.10 by Hezhijie and Lizhengnian
************************************************************************************************/
uint8 RunFlag = FALSE;
void ASRSuccess_Handle(uint8 asr_code)
{
    printf("\r\n识别码:%d\n",asr_code);
    if(0 == asr_code)
    {
        printf("我在，需要我的帮助吗？\n");
        TTSPlay(0, "[t3][2]我在,[2]需要[2]我的[3]帮助吗");
        RunFlag = TRUE;
    }
    else if(RunFlag)
    {
        RunFlag = FALSE;
        /* 识别码0-10为搜索天气识别码 */
        if(asr_code>=0&&asr_code<=10)
        {
            switch(asr_code)
            {
                case CODE01:
                    printf("“福州”命令识别成功\r\n");
                    TTSPlay(0, "[t3][2]正在为您搜索福州天气");
                    memcpy(g_city,"fujianfuzhou",sizeof(g_place));
                    break;
                case CODE02:
                    printf("“上海”命令识别成功\r\n");
                    TTSPlay(0, "[t3][2]正在为您搜索上海天气");
                    memcpy(g_city,"shanghai",sizeof(g_place));
                    break;
```

图 4-29 语音识别操作函数片段

从该函数可以看到，设计识别码 0 为一级指令码，当一级指令码识别成功后，才能进行二级指令地

识别。语音识别成功后，会调用“TTSPlay”函数播报所设定的语音，这样每一次地识别与播报就可以达到了一种人机互动的效果。“TTSPlay”函数如图 4-20 所示：

```
void TTSPlay(uint8_t Music,uint8_t *Text)
{
    /****************需要发送的文本**********************************/
    uint8_t DataPacket[50];     //
    uint8_t Text_Len;
    uint8_t ecc  = 0;           //定义校验字节
    uint8_t i=0;
    Text_Len =strlen((const char*)Text);            //需要发送文本的长度

    /****************帧固定配置信息**********************************/
    DataPacket[0] = 0xFD ;              //构造帧头FD
    DataPacket[1] = 0x00 ;              //构造数据区长度的高字节
    DataPacket[2] = Text_Len + 3;       //构造数据区长度的低字节
    DataPacket[3] = 0x01 ;              //构造命令字：合成播放命令
    DataPacket[4] = 0x01 | Music<<4 ;   //构造命令参数：背景音乐设定

    /******************校验码计算*****************************************/
    for(i = 0; i<5; i++)                //依次发送构造好的5个帧头字节
    {
        ecc=ecc^(DataPacket[i]);        //对发送的字节进行异或校验
    }

    for(i= 0; i<Text_Len; i++)          //依次发送待合成的文本数据
    {
        ecc=ecc^(Text[i]);              //对发送的字节进行异或校验
    }
    /******************发送帧信息*****************************************/
    memcpy(&DataPacket[5], Text, Text_Len);
    DataPacket[5+Text_Len]=ecc;
    UART4_SendStr((char*)DataPacket,5+Text_Len+1);
}
```

图 4-20 语音合成函数

该函数为文本合成语音的函数，第一个参数为背景音乐选择参数，第二个参数为所需要转化为语音的文本指针[14]。

4.2.4 更新天气图标至串口屏

```
/*************************************************************************************************************
** 函数: DisplayWeatherIcon, 更新天气图标至液晶触摸串口屏
**-------------------------------------------------------------------------------------------------------------
** 参数: weather：获取到的天气数据
** 返回: 无
** 说明: 刷图指令：pic x,y,picid  (x:起始点x坐标，y:起始点y坐标，picid:图片ID)
** 作者: 2017.12.18 by Hezhijie and Lizhengnian
*************************************************************************************************************/
char icon_buf[32] = {0};
void DisplayWeatherIcon(Result_T weather)
{
    if(strcmp(g_page,"page_Desk")==0)
    {
        printf("page Desk!\n");
        memset(icon_buf, 0, sizeof(icon_buf));
        sprintf(icon_buf, "pic %d,%d,%d", Today_Icon_X, Today_Icon_Y, Today_Icon_ID);
        HMISendStr(icon_buf);
        HMISendByte(0xff);
        memset(icon_buf, 0, sizeof(icon_buf));
        sprintf(icon_buf, "pic %d,%d,%d", Tomorrow_Icon_X, Tomorrow_Icon_Y, Tomorrow_Icon_ID);
        HMISendStr(icon_buf);
        HMISendByte(0xff);
        memset(icon_buf, 0, sizeof(icon_buf));
        sprintf(icon_buf, "pic %d,%d,%d", AfterTomor_Icon_X, AfterTomor_Icon_Y, AfterTomor_Icon_ID);
        HMISendStr(icon_buf);
        HMISendByte(0xff);
    }
}
```

图 4-21 更新天气图标至串口屏

从软件系统的架构来看，更下面一层的更新天气图标至串口屏的操作函数如图 4-21 所示。其中，先判断页面标识是不是在桌面主页，如果是，则执行发送动作。首先进行的是当天天气图标的更新，往串口屏发送字符串命令之前，先清空用于存储串口屏命令的缓存，再调用字符串格式化函数 sprintf 函

数去格式化刷图指令。刷图指令为“pic x,y,picid”，其中 x，y 为图片左上角坐标，picid 为存储在本地的图片 ID，不同的 ID 以区分不同的图片。sprintf 格式化刷图指令后并存储到 icon_buf 缓存中，再调用 HMISendStr 函数发送该缓存中的命令，最后调用 HMISendByte 函数发送“0Xff”结束符。至此，完成当天天气图标的更新。然后重复以上操作两次即可完成明天、后天两天天气图标的更新。

4.3 串口屏上位机代码示例

图 4-22 串口屏上位机代码示例一

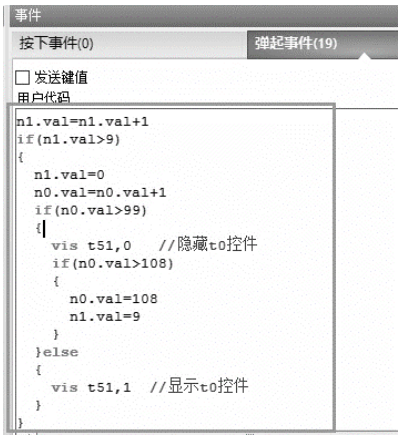

图 4-23 串口屏上位机代码示例二

串口屏上位机代码示例如图 4-22 和图 4-23 所示。串口屏指令为字符串指令，根据需要为每个控件编写代码。图 4-22 为串口屏上位机与单片机交互的代码示例，运用 print 指令发送 WiFi 名称与密码至下位机串口，其中“wifi_ssid”与“wifi_passsword”为自定义的协议头，这些协议头的制定是为了区分不同的数据。图 4-23 为串口屏上位机控件与控件之间交互的代码示例，其功能为：调节控件“t51”和控件“n0”的值。

5 系统调试及遇到的问题

5.1 JSON 格式数据解析

在做解析 JSON 数据的测试时，我模拟一个天气数据包来验证我编写的测试代码，我把该天气数据包保存在文本文件 now.txt 中[15]，如图 5-1 所示：


```
{
    "results":
    [
        {
            "location":
            {
            "id":"WSSU6EXX52RE",
            "name":"Fuzhou",
            "country":"CN",
            "path":"Fuzhou,Fuzhou,Fujian,China",
            "timezone":"Asia/Shanghai",
            "timezone_offset":"+08:00"
            },
            "now":
            {
            "text":"Sunny",
            "code":"0",
            "temperature":"20"
            },
            "last_update":"2017-12-10T16:00:00+08:00"
        }
    ]
}
```


图 5-1 模拟的天气实况天气数据包

然后通过 cJSON 库进行解析，解析得到的结果如图 5-2 所示：

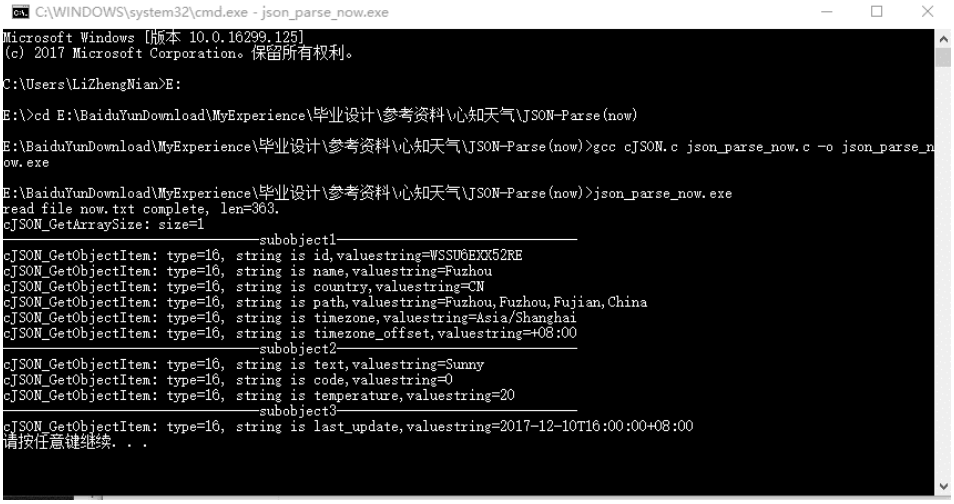

图 5-2 解析结果

由图 5-2 可看到，解析得到的结果与 now.txt 中的有用的数据完全一致，解析成功！

5.2 系统死机问题

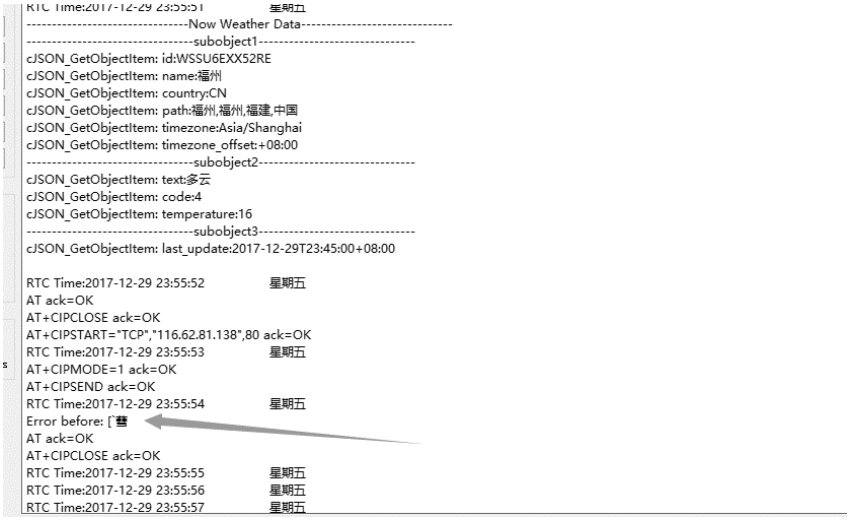

图 5-3 系统死机时的现象

原因：在 cJSON_Parse 函数内部已经有申请一整块的内存空间的情况下，在外部又给各个指针变量单独申请空间，多次调用获取天气数据的函数之后就会导致系统因为空间不足而崩溃。

解决办法：把多余的申请空间的语句删除掉。

5.3 使用中文搜索天气的问题

在实现中文搜索这一项功能的时候出现了乱码问题，问题出在申请包的 URL 上。错误示例："GET https://api.seniverse.com/v3/wea ... 9o9f9&location=北京&language=zh-Hans&unit=c\n\n"。正确的应该为："GET https://api.seniverse.com/v3/wea...9o9f9&location=%E5%8C%97%E4%BA%AC&language=zh-Hans&unit=c\n\n"。即问题出现在城市名称的编码上面，汉字在 URL 中只有转换为 UTF-8 心知天气服务器才可以解析得到。如“北”对应的 UTF-8 编码为：E58C97。可通过“千千秀字”网站查询。如图 5-4 所示：

图 5-4 查询汉字 UTF-8 编码

5.4 串口屏天气数据显示问题

在更新天气数据至串口屏的过程中出现显示不正确的问题。问题如图 5-5 所示：

图 5-5 串口屏天气数据显示不正确

问题原因：在每次获取天气数据之前没有清空接收缓冲，导致数据覆盖。

解决办法：每次接收数据之前清空缓存。对应代码语句为：memset(&weather_data, 0, sizeof(weather_data));。

5.5 语音识别调试

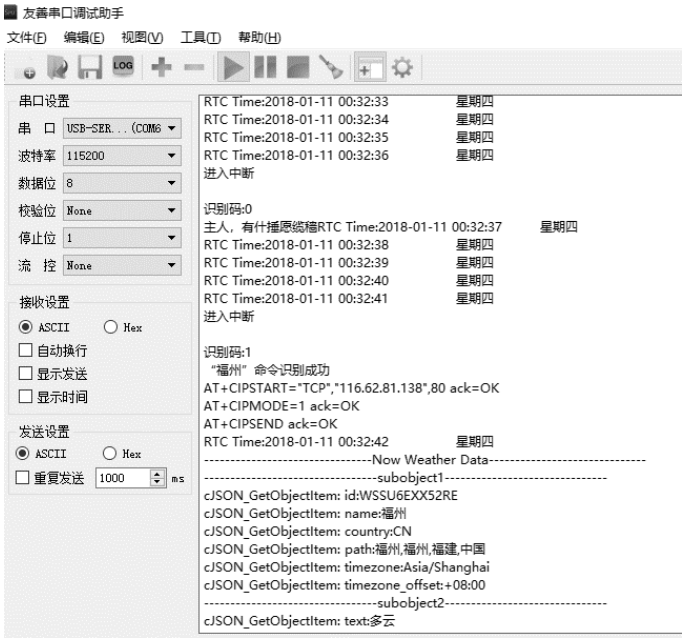

图 5-6 语音识别成功

语音识别成功时串口上位机打印输出的信息如图 5-6 所示。我把识别码 0 设定为一级指令，识别码 0 之外的其他识别码设定为二级指令。只有先触发一级指令，才能触发二级指令，触发一级指令的语音

信息为“小天”。如图 5-6，一级指令触发时，串口打印出“主人，有什么吩咐吗”这一句话。此时，才可触发二级指令来搜索天气信息或者其他信息，如搜索福州天气的语音指令为“搜索福州天气”，该条语音指令的返回结果如图 5-6 所示，即打印“‘福州’命令识别成功”，并打印搜索得到的天气信息。

5.6 实物图

图 5-7 实物图主界面

本设计的实物图主界面（桌面）如图 5-7 所示。左上角是 WiFi 连接成功的图标，图标右侧是心知天气的 logo 以及“心知天气”四个大字，为了支持心知天气团队的资源共享精神以及遵守其相关协议（即免费用户，需在数据展示页面注明数据来源为心知天气）而在此处放置其 logo。其中，点击“心知天气”这四个字可以进入菜单界面。再往右是搜索框，点击搜索框即可跳转到键盘界面，输入完城市名称并确认后即可返回主界面，此时，点击搜索框左边的搜索图标即可搜索天气。主界面中间部分显示的是天气信息以及时间日期与地点。屏幕最下方的滚动条为温馨提示信息，可根据天气的变化显示出合适的温馨提示。

6 总结

至此，本毕设的工作内容已经完成得差不多了，基本达到预期的效果。本课题采用 STM32F103ZET6 为主控，外围模块有 WiFi、串口屏、收音机、语音识别、语音合成、温湿度、粉尘传感器等 7 个模块。本设计已经实现的主要功能为：（1）液晶屏实时显示天气、温湿度、时间日期等信息；（2）具有收音机功能，可通过液晶触摸屏调节收音机频率；（3）可以通过液晶触摸屏配置 WiFi 信息以适应不同的 WiFi 环境；（4）可通过液晶触摸屏搜索不同城市的天气，并且支持中、英文搜索；（5）具有人机对话功能，如可通过语音搜索天气、可以询问当前温度、可以询问时间信息等。

本毕设历经几个月的时间，经过了三个阶段。 初期，查阅大量资料，确立方案。中期，编写、调节各模块。后期，系统联调及编写论文。期间，遇到很多问题，如解析天气数据时解析 JSON 数据错误、WiFi 连接问题、系统死机等问题，后来都得以解决。总之，本次毕业设计，提高了我分析、解决问题的能力。

7 致谢

在此感谢物理与能源学院提供一个很好地学习环境，感谢何志杰老师对我的帮助与教导。从大二至今一直在帮助着我，带着我学习软硬件知识、做实验、做项目，以及在本次设计的过程中，耐心地指导我解决很多问题。

参考文献

[1] 心知天气-行业客户案例[EB/OL]. https://www.seniverse.com/cases

[2] 徐成波. 智能建筑照明节能控制系统的研究[D]. 吉林建筑大学, 2015.

[3] ESP8266_用户手册_V03[EB/OL].
https://wenku.baidu.com/view/dd2f8965581b6bd97e19ea37.html

[4] 张书源, 许莹, 张磊晶. 基于 TEA5767 与 BH1415F 的数字调频收发系统[J]. 电子技术与软件工程, 2015(16):128-129.

[5] 陈守满. 基于 LD3320 语音遥控器的设计与实现[J]. 安康学院学报, 2013, 25(6):1-3.

[6] 曾维鹏, 蔡莉莎. 基于手机短信的智能交互式环境监测仪的设计与实现[J]. 苏州市职业大学学报, 2015(2):19-21.

[7] 雾霾远离我：PM2.5 检测器自制解析[EB/OL]. http://39.elecfans.com/20140930355383_a.html

[8] 王雷. 基于 ARM 和 ZigBee 技术智能家居系统的设计与实现[D]. 河北科技大学, 2014.

[9] 徐永超. 基于 LonWorks 网络的消防应急与逃生系统硬件研究[D]. 浙江大学, 2006.

[10] 刘婧. 电梯及扶梯 GSM 无线远程报警系统设计与应用[D]. 上海交通大学, 2011.

[11] 使用 cJSON 解析 JSON 字符串[EB/OL].
https://blog.csdn.net/lintax/article/details/50993958

[12] 关于 STM32 能否使用 malloc 申请动态内存问题[EB/OL].
https://blog.csdn.net/c12345423/article/details/53004465

[13] 基于 Cortex-M3 的语音识别[EB/OL].
http://www.openedv.com/forum.php?mod=viewthread&tid=21183&highlight=%D3%EF%D2%F4%CA%B6%B1%F0

[14] 孙鹏飞. 移动综合网络应用客户端和服务器的研究与开发[D]. 北京邮电大学, 2013.

[15] 学习笔记——天气数据解析 1（JSON 格式数据）[EB/OL].
https://blog.csdn.net/zhengnianli/article/details/79223457

Design of desktop weather forecast system

College of Physics and Energy New Energy Science and Engineering

135032014207 Li Zhengnian Adviser:He Zhijie

Abstract: It is of great practical significance to forecast the weather. The system uses STM32 microcontroller as the control chip, and obtains weather forecast data through WiFi module GET weather API interface. The controller sends the acquired weather data to the serial screen through the serial port, and combines the beautiful GUI interface to form a set of dazzling desktop weather forecast system. This system has the function of touch screen search and voice search weather. It can search weather data of every city and update it to serial port display. In addition, the system also has the following functions: WiFi configuration function, air quality detection, radio function, voice recognition function, real-time display time.

Key words: Weather forecast, WIFI, Serial port screen, Radio, Speech recognition.