智能渔林生态系统

郭凯，文福金

**第一部分 设计概述**

* 1. 设计目的

本项目目的是为了解决渔林生长问题，为用户提供一个操作简单、使用方便的基于NB-IoT的农业应用系统。用户可以在手机端下载手机APP进行数据的实时查看；通过设备端登录串口屏界面，查看数据及其变化趋势；可在电脑端直接登录网址进行相应的操作。

* 1. 应用领域

智能渔林生态广泛应用于各大果林和鱼业养殖中。通过传感器对环境的数据采集，实时感知环境状况，通过将数据传输至网端，可进行数据分析，实现

* 1. 主要技术特点

硬件部分：

本系统由STM32控制板、多种传感器、串口屏、增氧泵和水泵搭建而成。 当STM32处理器控制传感器采集系统的状况数据信息后，经过一定处理，一方面，显示在7寸串口屏上，另一方面，将其打包，通过NB-IoT模块上传到OneNET平台，从而发送至服务器端，而服务器下发的数据，也通过NB-IoT模块进行接收，接收到的数据经解析后可控制相应设备的运行，最终实现数据处理、数据库的可靠存储、后台统一管理，前端web网页、手机App、数据展示和设备控制等功能，实现对农业气候动态的监控管理，并能将动态信息传递至移动设备端。

* 1. 关键性能指标

该系统主要性能就是可以代替人工不间断的对该地水温数据等进行实时监测，而且采集的数据较为准确、规律性强、相对于认为观测误差小等优点。稳定性强、实用范围广。既可以室内也可以外界使用，通过互联网，将采集的数据上传到服务器，这样可以实现远距离观测数据。不受地域的限制，很方便。上电后，可在串口屏系统设置界面上打开数据采集按钮，对数据进行采集，关闭按钮，则系统停止采集数据。而APP或者网页从onenet平台不断的获取NB-IOT发送的数据，保存到数据库。上传的数据不仅仅能从web网页上面看到；还能在手机App上面看到上传的数据。

**第二部分 系统组成及功能说明**

* 1. 整体介绍

本项目的整体框架如图1所示，在终端采集到数据后，通过NB-IoT将数据上传到服务器。用户在PC端打开本系统网页或APP端打开本系统软件后，通过TCP协议连接到服务器，读取服务器中的数据。同时串口屏上也会将数据显示出来，通过串口屏还可以对数据进行修改。并且，当果林土壤湿度过低时，处理器会控制果林浇水中枢对果林进行浇水；每隔一段时间处理器会控制供氧中枢给鱼塘提供氧气。

图1 渔林生态系统框架图

* 1. 各模块介绍

2.2.1硬件介绍

（1）处理器。以STM32F407VET6芯片作为处理器，控制整个程序的运行。处理并接收传感器采集到的数据，将数据打包，并通过NB-IoT将打包好的数据发送给OneNET平台。同时芯片通过串口的方式将数据在串口屏上显示。

1. NB-IoT模块。作为硬件设备与服务器的通信纽扣，实现硬件设备与服务器数据与指令的传输。
2. 数据采集传感器。数据采集传感器包括土壤水分传感器、二氧化碳传感器、烟雾传感器、浑浊度传感器和水温传感器。芯片通过RS485，ADC采集数据方式，获取各传感器的数据。
3. 串口屏。采用F系列的电容触摸串口屏，实现数据的显示和对系统的操作控制。
4. 报警与控制模块。处理通过对IO口的控制，实现报警系统、对供水系统和供氧系统的控制。

2.2..2软件介绍

1. 手机客户端

对于手机APP，我们采用简洁、美观、方便、容量小的理念，既能有效的完成我们功能，也能减少对手机内存的负担。

在使用APP前先进行注册再登录。注册时需要点击“同意协议条款”后，出现注册按钮，注册成功后自动跳回“登录界面”，输入用户名、密码后用户可根据自行需要选择“记住密码”。 接着进入我们的app主页，在APP上我们主要集合了“首页信息浏览展示”、“渔林数据实时展示”、“个人信息获取及展示”和“专家系统建议展示”，“智能远程控制模块”五个模块。其中，在“渔林数据展示模块”点击刷新后，数据立即刷新，相比较于计时器定时传输数据，既能减轻了服务器的负担，也能满足用户实时更新的需求。

**第三部分 完成情况及性能参数**

（具体完成的情况，可图文结合，具体的性能参数等量化指标）

本项目的硬件部分和软件部分均已设计完成。

3.1硬件部分

如图3.1所示，是本项目的整体模型图，主要由果林区域，鱼塘区域，和操作控制区三个区域组成。



图3.1整体模型图

3.1.1果林区域

装有各种传感器，用于采集果林中的二氧化碳浓度、土壤温湿度、空气温湿度和烟雾浓度的数据；并装有浇灌系统，实现对果林的浇灌。

3.1.2鱼塘区域

装有水温传感器和水浑浊传感器，实时对鱼塘的水质进行监测；并装有增氧系统，实现对鱼塘的增氧。

3.1.3操作控制区

由操作屏和控制板组成，如图3.2和图3.3所示。实现对整个系统的控制。



图3.2 操作屏

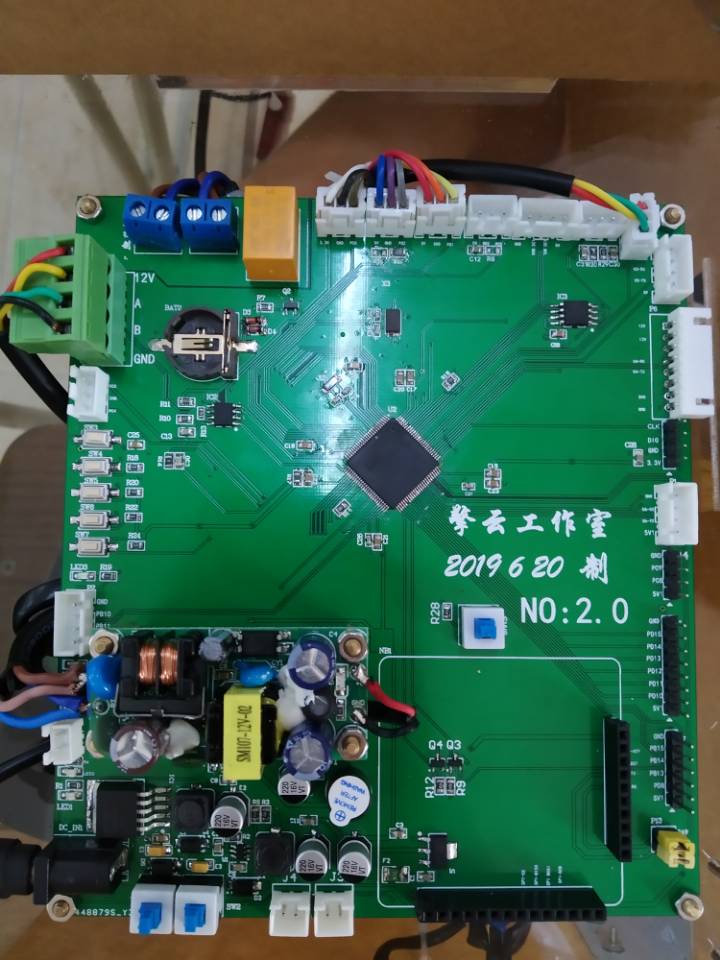


图3.3主控板

3.2操作屏界面介绍

3.1.1登入界面



图3.4 用户登录界面

图3.4为用户登录界面，用户通过输入正确的账号和密码即可进入主界面。

3.2.2 主界面



图3.5 主界面

从图3.5可以看到，主界面包括以下几个部分：

（1）数据显示区：显示传感器采集到的数据，主要数据有：空气温度、空气湿度、土壤温度、土壤湿度、氧气浓度、CO2浓度、水温、烟雾浓度、土壤盐分、土壤电导率和浑浊度。

（2）智能控制选项区：点击智能控制按钮，进入智能控制界面。

（3）阀值设置选项区：点击阀值设置按钮，进入阀值设置界面，可对各个数据阀值进行设置。

（4）系统设置选项区：点击系统设置按钮，进入系统设置界面。

（5）数据刷新选项区：可手动刷新数据。

（6）数据历史曲线选项区：点击每个数据的历史曲线按钮，进入数据历史曲线界面，查看数据变化趋势。

3.2.3阀值设置界面

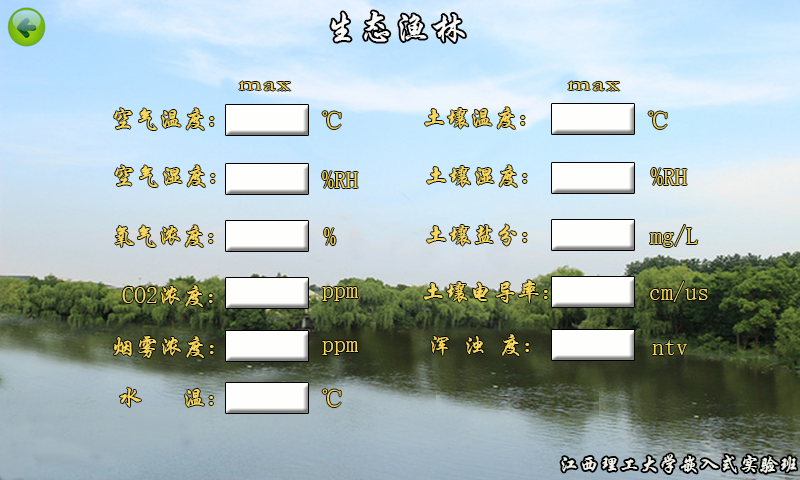


图3.6 阀值设置界面

图3.6为阀值设置界面，该界面显示了空气温度、空气湿度、土壤温度、土壤湿度、氧气浓度、CO2浓度、水温、烟雾浓度、土壤盐分、土壤电导率和浑浊度的最大值，用户也可以对阀值进行修改。

3.3.4系统设置界面

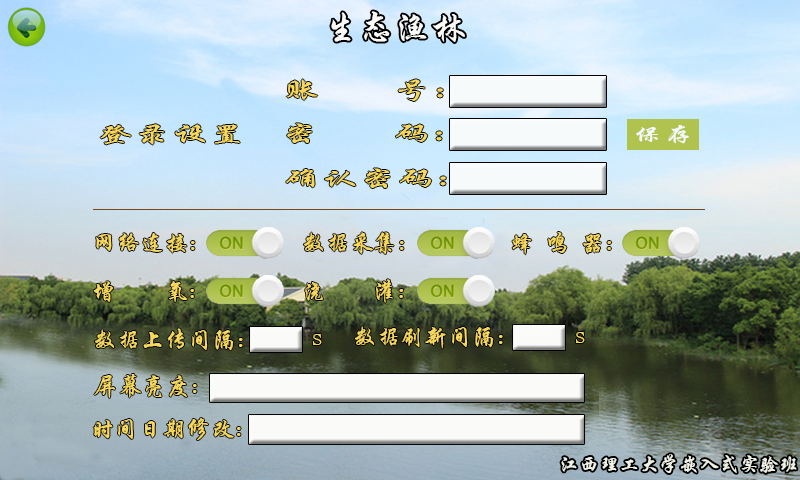


图3.7 系统设置界面

图3.7为系统设置界面，这个界面可通过开关按钮控制网络连接、数据采集、蜂鸣器、增氧和浇灌的状态，显示数据刷新和上传时间。用户可以在登录设置中添加新的账号和密码并保存，还可根据个人需要调节串口屏的亮度以及设置、修改当前时间。

3.3.5智能控制界面



图4-14 智能控制界面

此界面可以智能控制是否自动增氧和浇灌。当打开智能控制开关，可以对自动浇灌和自动增氧进行开关控制，实现对鱼塘的自动增氧和对果林的自动浇灌。

3.3软件部分

### 3.3.1服务器层



### 3.3.2前端网页层



### 3.3.3前端APP层



3.4软件操作界面介绍

3.4.1前端网页功能介绍

（1）注册界面

用户需要提供一下信息进行账户注册:

用户名：必填，长度在5到10位之间

邮箱 ：标准邮箱地址

密码：长度在6到10 位之间



图3.3.4 注册

（2）登录界面



图3.3.5 登录

（3）管理员登录后主界面



图3.3.6 管理员登录后主界面

在登录完，将进入我们的主页，这有“今日消息”、“历史消息”、“系统提示消息”、“设备相关设置”以及“关于我们”五大模块，“今日消息”模块下有A区、B区等，进行分区管理

（4）管理员功能界面

管理员具有管理权限，他可以在系统管理中修改设置最大值的界限，提前对设备进行提前设置，当它超过设置的上限时，硬件层能及时的做出处理



图3.3.7 设备设置区（1）



图3.3.8 设备设置区（2）

管理员可以对标准区进行设置和对设备进行设备设置。



图3.3.9 标准区

标准区的设置方式，先点击历史消息，再点击“日”查询，在历史信息的某一天中选择合适的某一天作为“标准”，待选择完毕后，再点击当前页面的“是否设置标准区参数”区域，点击确认即可





图3.3.10 历史信息置区

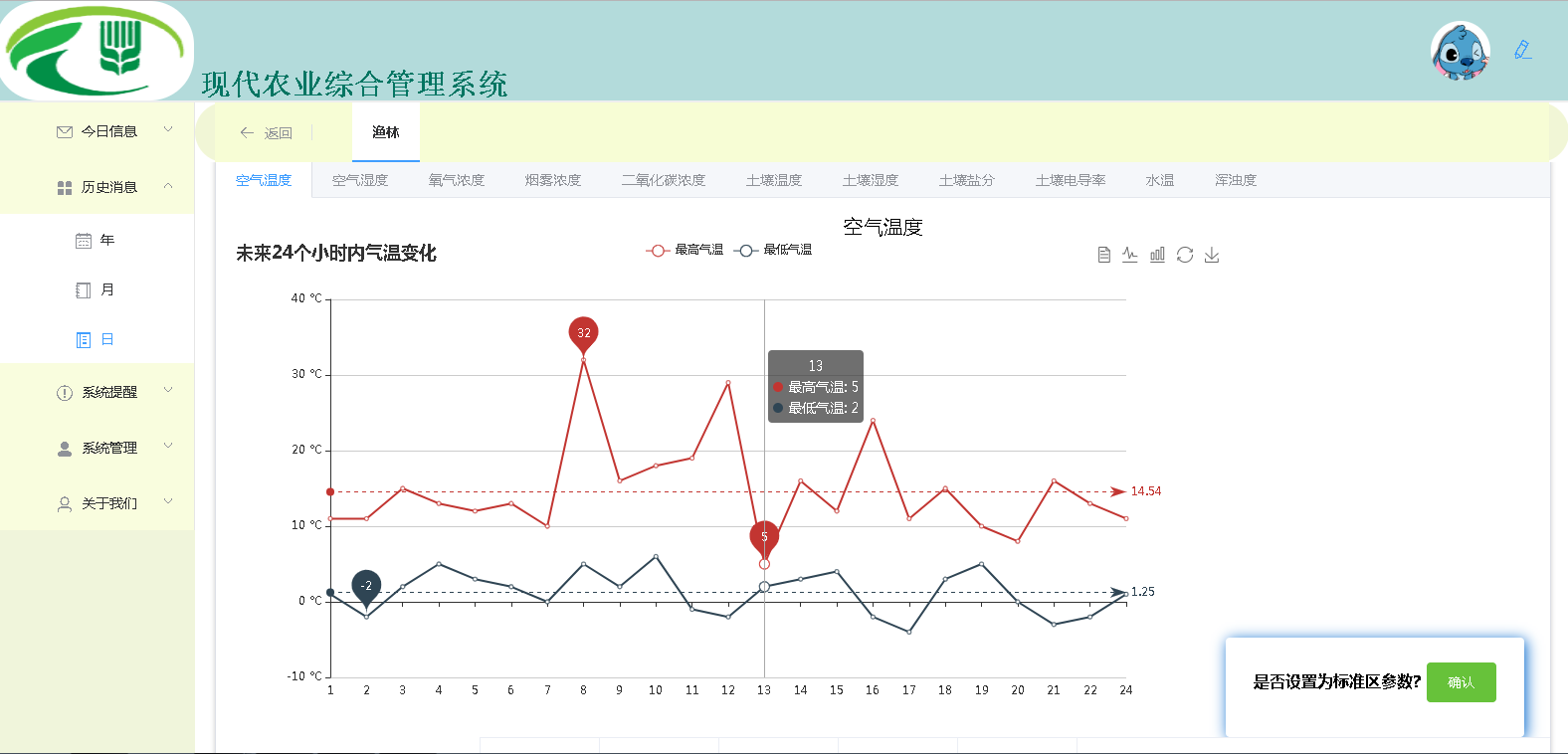


图3.3.11 标准设置区

（5）管理员及普通员工公用功能界面

今日信息查询：“今日消息”模块下有A区、B区等，进行分区管理，在区域页面中的表格区域为我们时刻展示着设备的当前最新数据，下面的统计图宏观的展示出设备数据的变化情况，同时我们设立了标准区（标准区的设置参阅上文），让我们可以选择历史消息中的某一天作为“标准”,在今日消息的统计图区域，生成一个标准，结合让当天的设备实际数据统计折现，让我们对设备数据有醒目的认识以及一定的预测判断处理。

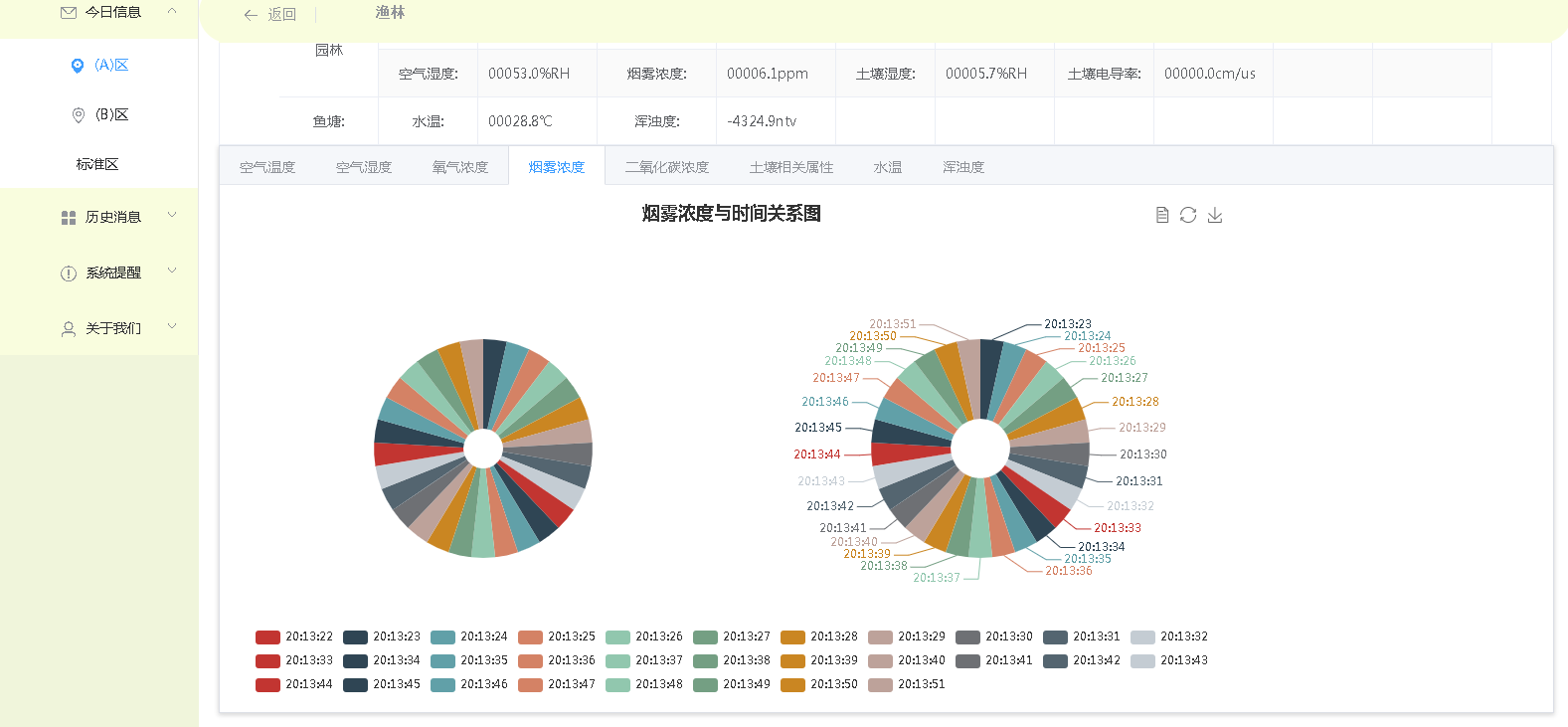


图3.3.12 今日信息（A区）



图3.3.15 设备设置区（B区）

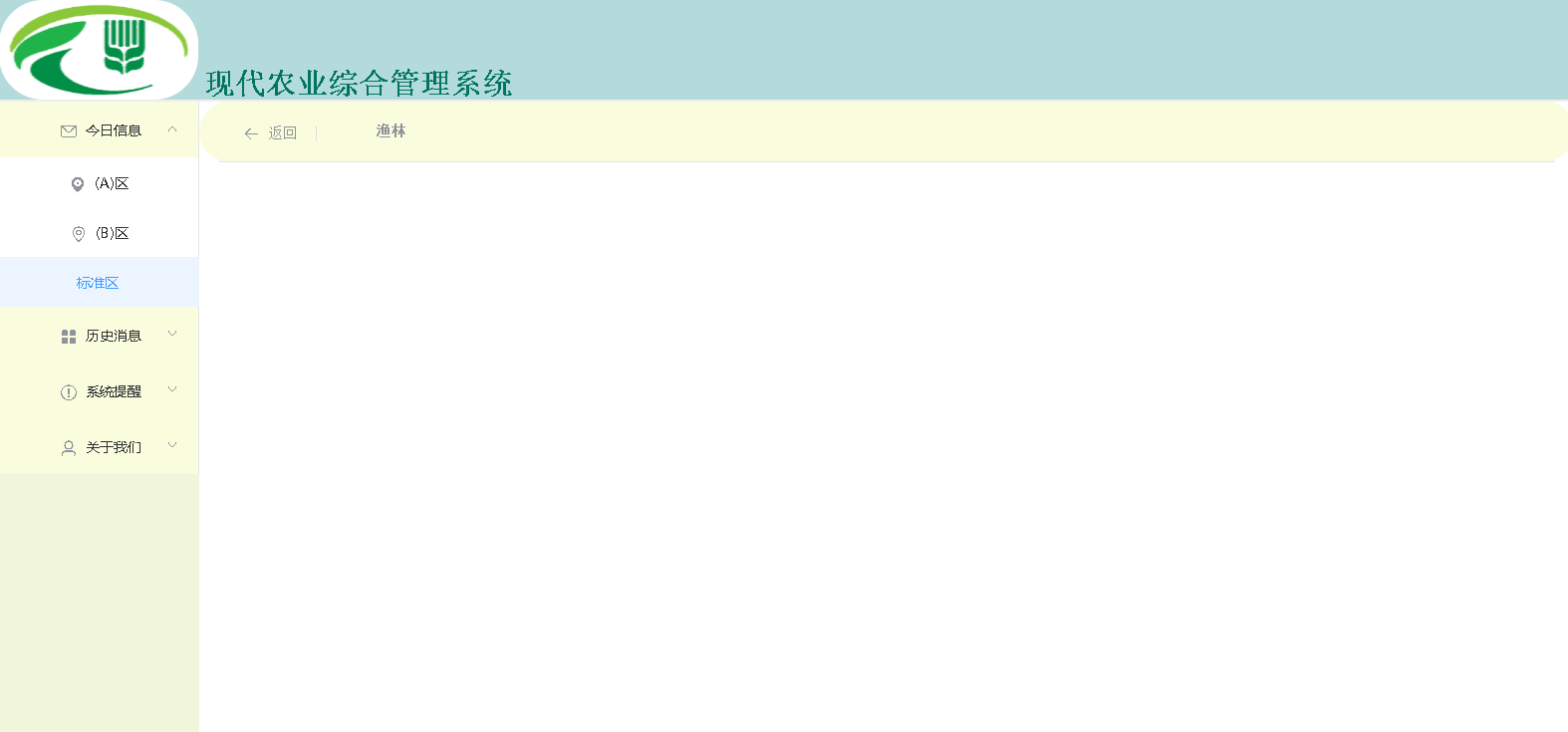


图3.3.16 设备设置区（标准区）

历史消息：在这区域将展示过去的信息，供相关人员查看，让其从中获取经验，为现在和以后的发展奠定基础。（因先处于开发阶段，我们数据库中的数据无法支持当前的数据规模，所以数据统计图先无法呈现，但已具备功能）

关于我们：在这个模块中，可以让使用者了解到我们。

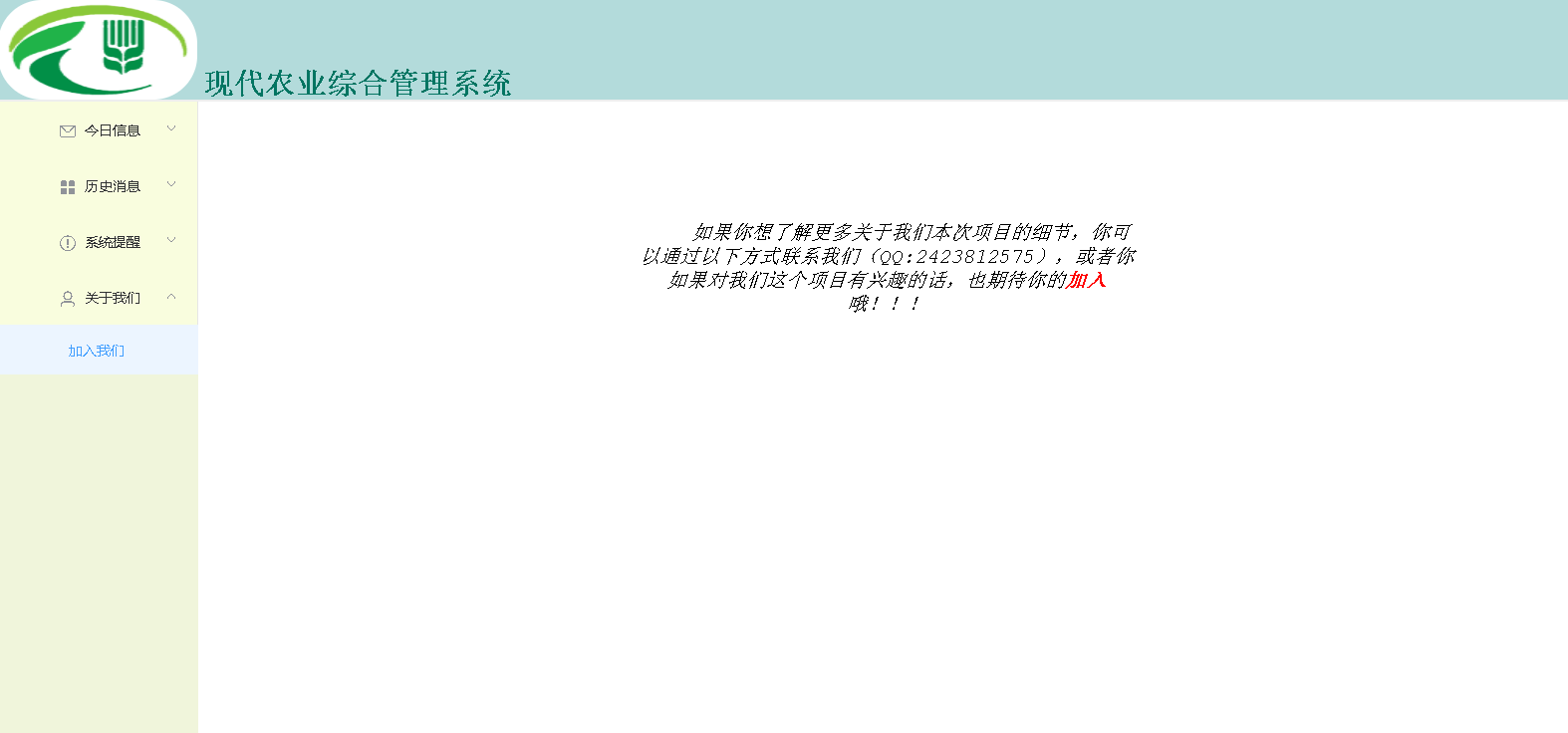


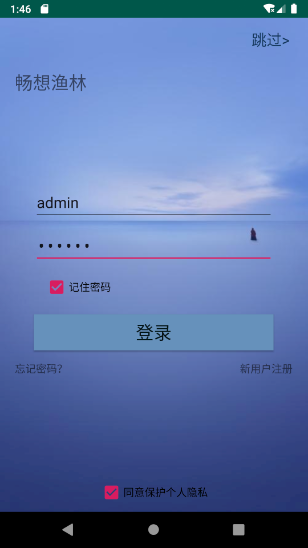
图3.3.20 加入我们

3.4.2前端APP功能介绍

（1）用户登录、注册、忘记密码界面

为了提高用户体验感，在实现登录注册、忘记密码的布局以及UI设置上进行了严格的美化控制，在布局上使用了相对布局、线性布局以及表格布局相结合的界面规划，在颜色上使用了蓝色和浅绿色为主要背景色，增加了app的色彩丰富性。从以下图中可以看出，

用户使用App时，首次使用需要先进行注册，注册信息会发送到服务器然后保存在后台数据库中。登录时输入正确的用户名和密码，才能够正常使用App，而且服务器端只是储存用户个人信息起到校正数据的作用，并不会将用户的信息公开或用于非法目的， 请用户放心使用。为了使app更大地提高用户方便性，在登录界面增加了“记住密码”的控件，减少多次输入用户名和密码的复杂。（其中第三张图片为用户注册成功效果图展示）

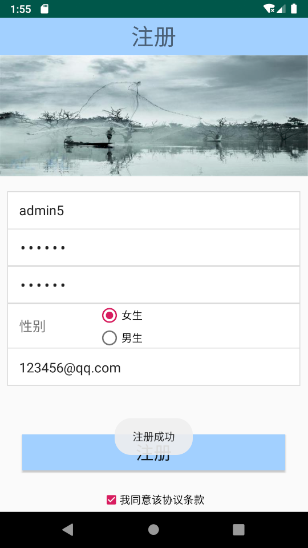
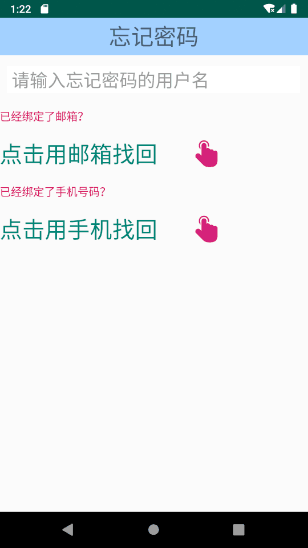
 

图3.4.1登录、注册、忘记密码界面

1. 用户首页浏览页面

成功登录进入APP后，首先进入首页的信息浏览页面，此页面包含以下特点：

①页面可滑动性：页面可以根据用户自己的爱好进行手指滑动或者底部的控件按钮进行页面切换。

②采用左上角控件风格，极大的提高用户的体验感。

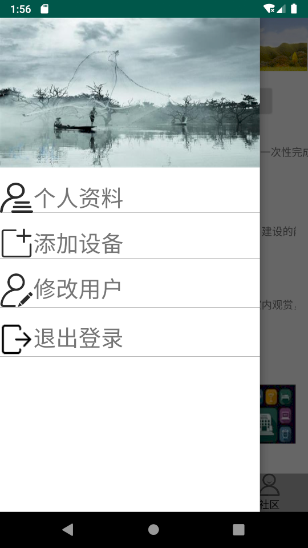
 

图3.4.2 用户首页浏览页面

（3）数据展示页面

当用户点击底部导航栏“数据”控件后，可以进入数据展示页面，在该页面的每一项数据前都使用了可视化的图标进行区分，可以清楚的查看从服务器接受的各项数据信息。并通过点击“刷新”按钮后，更新各项数据，满足用户实时更新的需求。



图3.4.3 渔林数据展示页面

（4）专家系统页面

专家系统是App推出的为用户服务的一款分析系统，用户可以通过APP点击专家系统页面进入，在此页面中，用户查看渔林分类等信息。另外，在安全数据区间设置上，用户可以修改专家系统提供的数据控制区间。图下给出了专家系统的页面设计效果图。

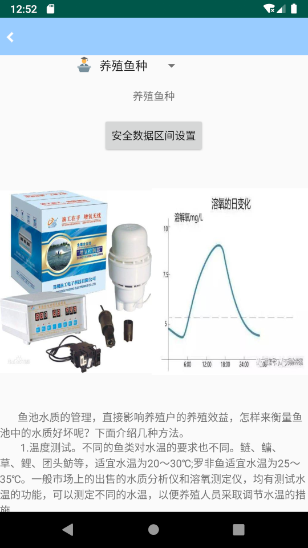
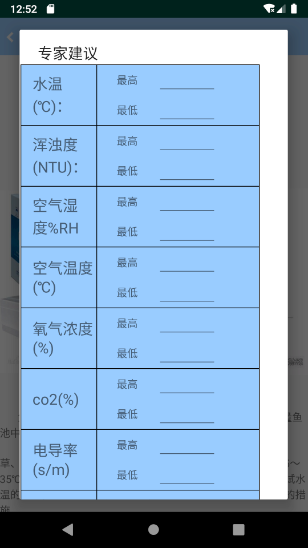
 

图3.4.4专家系统页面

（5）个人资料页面

同时畅想App还提供了个人资料查询，包括用户ID，头像、用户名，真实姓名、性别、身份、邮箱并可以进行信息的添加与修改。下面是个人信息页面如图所示。

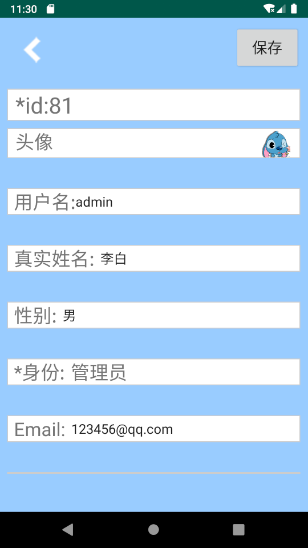
 

图3.4.5 个人资料页面

（6）智能远程控制页面

为了方便用户对设备的管理，在App上提供了简单的远程控制界面，可以对硬件进行简单的控制。当网络连接后，数据也自动联网，实时更新，用户可以在App中点击关闭设备，选择开启或关闭设备，远程控制设备的开关。下面是智能远程控制页面如图所示。

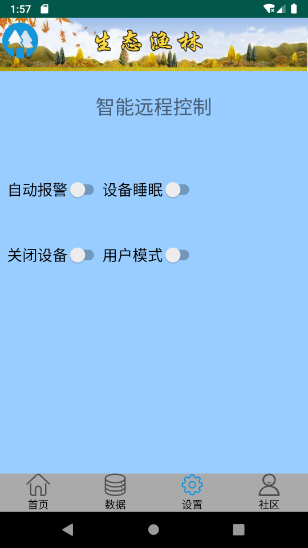


图3.4.6 智能远程控制页面

**第四部分 总结**

* 1. 主要创新点

（1）采用NB-IoT通信方式，实现数据的实时上传。

（2）可智能、手动控制增氧和浇灌。

（3）当采集到的数据超过所设置的阀值时系统自动报警。

（4）服务器远程监控，用户可通过手机、电脑等设备随时查看各数据变化趋势，了解到渔林的生长情况。克服以往使用蓝牙、WiFi等对信号距离的限制，系统能在有网络的情况下，不管距离因素，都能实现对硬件设备的数据监视以及对设备下发指令。

* 1. 可扩展之处

（1）可在本系统的环境中添加家禽的养殖，家禽肥料入塘当饲料，鱼塘塘底肥土给果林做养料，形成完整的渔林生态链，达到资源的高效率利用。

（2）多个本系统一起应用时，上传至网端的数据可行成大数据，通过对这些数据进行大数据分析，分析得到渔林养殖的最佳生长条件，实现用最小的养殖成本达到最大的养殖效益。

* 1. 心得体会

渔林生态系统的设计是一个完整的，复杂的框架。集成用户层、服务层、传输层、控制层和感知层一体的系统。

完整的结构框架使我们认识到团队合作，相互沟通是非常重要的，明确的分工让开发效率明显提升，可以减少了很多不必要的人力物力的浪费。同时我们了解到项目团队成员的团结合作、互相沟通是非常重要的，团队成员之间要互相学习彼此的优点和技术，才能使团队的能力不断的提高。

**第五部分 参考文献**

[1].吉雷主编. Protel99 从入门到精通. 西安电子科技大学出版社，2000.10

[2].康华光主编. 电子技术基础（模拟部分第五版）.高等教育出版社, 1999.6

[3].康华光主编. 电子技术基础（数字部分第五版）.高等教育出版社, 2005.7

[4]唐程山.《电子技术基础》[M]. 高等教育出版社

[5]嵌入式系统综合项目开发指导书，谢小云

[6] 嵌入式实时操作系统μCOS-III，Jean J.Labrosse，西安电子科技大学出版社

[7]Jean J.Labrosse著，宫辉等译. 嵌入式实时操作 系统μc/os-III[M].北京：北京航空航天大学出版 社，2012.

[8]Jean J.Labrosse著，何小庆等译. 嵌入式实时操作 系统μc/os-III应用开发.北京：北京航空航天大学 出版社，2012.

[9]嵌入式实时操作系统C/OSC-III应用开发-基于STM32微控制器，机械工业出版社出版，拉伯罗斯，北京航空航天大学出版社

[9]OneNet开发文，OneNet平台

[10]《软件工程导论》，张海藩，清华大学出版社

[11]《Java EE企业级应用开发教程》，人民邮电出版社

[12]《编写可维护的JavaScript》，【美】Nicbolas C. Zakas 著，人民邮电出版社

[13]《mysql必知必会》，【英】福塔 著， 人名邮电出版社

[14]《图解http》，【日】上野宣；于均良 著，人民邮电出版社

[15]《Android从入门到精通》，赵洛育，清华大学出版社

[16]《软件测试基础教程》，清华大学出版社

**第六部分 附录**

重要代码、推导过程等不便于在正文中体现的内容

硬件部分

#include "includes.h"

#include "os\_app\_hooks.h"

#include "sys.h"

#include "delay.h"

#include "led.h"

//#include "usart.h"

#include "UART2\_6.h"

#include "TX\_RX.h"

#include "key.h"

#include "rs485.h"

#include "stdio.h"

#include "send\_rec.h"

#include "dht11.h"

#include "ds18b20.h"

#include "adc.h"

#include "uart\_lcd.h"

#include "nbiot.h"

#include "beep.h"

//#include "WindRain.h"

//#include "EnvTr.h"

#include "shuiban.h"

//任务优先级

#define START\_TASK\_PRIO 3

//任务堆栈大小

#define START\_STK\_SIZE 512

//任务控制块

OS\_TCB StartTaskTCB;

//任务堆栈

CPU\_STK START\_TASK\_STK[START\_STK\_SIZE];

//任务函数

void start\_task(void \*p\_arg);

//NBIoT任务

//任务优先级

#define NB\_TASK\_PRIO 10

//任务堆栈大小

#define NB\_STK\_SIZE 128

//任务控制块

OS\_TCB NBTaskTCB;

//任务堆栈

\_\_align(8) CPU\_STK NB\_TASK\_STK[NB\_STK\_SIZE];

//任务函数

void NB\_task(void \*p\_arg);

//串口屏显示任务

//任务优先级

#define DIS\_TASK\_PRIO 5

//任务堆栈大小

#define DIS\_STK\_SIZE 128

//任务控制块

OS\_TCB DISTaskTCB;

//任务堆栈

\_\_align(8) CPU\_STK DIS\_TASK\_STK[DIS\_STK\_SIZE];

//任务函数

void uart\_dis\_task(void \*p\_arg);

//各操作响应任务

//任务优先级

#define RESP\_TASK\_PRIO 7

//任务堆栈大小

#define RESP\_STK\_SIZE 128

//任务控制块

OS\_TCB RespTASKTCB;

//任务堆栈

\_\_align(8) CPU\_STK RESP\_TASK\_STK[RESP\_STK\_SIZE];

//任务函数

void resp\_task(void \*p\_arg);

//////////////////////////////////////////////////////////

OS\_TMR tmr1; //定时器1

void tmr1\_callback(void \*p\_tmr,void \*p\_arfg); //定时器1回调函数

u16 TMR\_times=0,reckontime=0,renewcount=0,upcount=0,up\_up=0;

/////////////////////////////////////////////////////////////////////

int main(void)

{

OS\_ERR err;

CPU\_SR\_ALLOC();

delay\_init(168); //时钟初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//中断分组配置

LED\_Init(); //LED初始化

KEY\_Init();

usart1\_init(115200); //串口初始化

usart4\_init(115200);

usart5\_init(115200);

// usart2\_init(115200);

// usart3\_init(115200);

// usart4\_init(115200);

// usart6\_init(115200);

RS485\_Init(9600); //初始化RS485

Adc\_Init(); //初始化ADC

UART\_LCD\_Init();

SHUIBAN\_Init();

NB\_init();

BEEP\_Init();

OSInit(&err); //初始化UCOSIII

OS\_CRITICAL\_ENTER();//进入临界区

//创建开始任务

OSTaskCreate((OS\_TCB \* )&StartTaskTCB, //任务控制块

(CPU\_CHAR \* )"start task", //任务名字

(OS\_TASK\_PTR )start\_task, //任务函数

(void \* )0, //传递给任务函数的参数

(OS\_PRIO )START\_TASK\_PRIO, //任务优先级

(CPU\_STK \* )&START\_TASK\_STK[0], //任务堆栈基地址

(CPU\_STK\_SIZE)START\_STK\_SIZE/10, //任务堆栈深度限位

(CPU\_STK\_SIZE)START\_STK\_SIZE, //任务堆栈大小

(OS\_MSG\_QTY )0, //任务内部消息队列能够接收的最大消息数目,为0时禁止接收消息

(OS\_TICK )0, //当使能时间片轮转时的时间片长度，为0时为默认长度，

(void \* )0, //用户补充的存储区

(OS\_OPT )OS\_OPT\_TASK\_STK\_CHK|OS\_OPT\_TASK\_STK\_CLR, //任务选项

(OS\_ERR \* )&err); //存放该函数错误时的返回值

OS\_CRITICAL\_EXIT(); //退出临界区

OSStart(&err); //开启UCOSIII

while(1);

}

//开始任务函数

void start\_task(void \*p\_arg)

{

OS\_ERR err;

CPU\_SR\_ALLOC();

p\_arg = p\_arg;

CPU\_Init();

#if OS\_CFG\_STAT\_TASK\_EN > 0u

OSStatTaskCPUUsageInit(&err); //统计任务

#endif

#ifdef CPU\_CFG\_INT\_DIS\_MEAS\_EN //如果使能了测量中断关闭时间

CPU\_IntDisMeasMaxCurReset();

#endif

#if OS\_CFG\_SCHED\_ROUND\_ROBIN\_EN //当使用时间片轮转的时候

//使能时间片轮转调度功能,时间片长度为1个系统时钟节拍，既1\*5=5ms

OSSchedRoundRobinCfg(DEF\_ENABLED,1,&err);

#endif

//创建定时器1

OSTmrCreate((OS\_TMR \*)&tmr1, //定时器1

(CPU\_CHAR \*)"tmr1", //定时器名字

(OS\_TICK )0, //0ms dly

(OS\_TICK )2, //2\*10=20ms period

(OS\_OPT )OS\_OPT\_TMR\_PERIODIC, //周期模式

(OS\_TMR\_CALLBACK\_PTR)tmr1\_callback,//定时器1回调函数

(void \*)0, //参数为0

(OS\_ERR \*)&err); //返回的错误码

OS\_CRITICAL\_ENTER(); //进入临界区

//创建NBIoT任务

OSTaskCreate((OS\_TCB \* )&NBTaskTCB,

(CPU\_CHAR \* )"nb test task",

(OS\_TASK\_PTR )NB\_task,

(void \* )0,

(OS\_PRIO )NB\_TASK\_PRIO,

(CPU\_STK \* )&NB\_TASK\_STK[0],

(CPU\_STK\_SIZE)NB\_STK\_SIZE/10,

(CPU\_STK\_SIZE)NB\_STK\_SIZE,

(OS\_MSG\_QTY )0,

(OS\_TICK )0,

(void \* )0,

(OS\_OPT )OS\_OPT\_TASK\_STK\_CHK|OS\_OPT\_TASK\_STK\_CLR,

(OS\_ERR \* )&err);

//创建串口屏显示任务

OSTaskCreate((OS\_TCB \* )&DISTaskTCB,

(CPU\_CHAR \* )"DIS test task",

(OS\_TASK\_PTR )uart\_dis\_task,

(void \* )0,

(OS\_PRIO )DIS\_TASK\_PRIO,

(CPU\_STK \* )&DIS\_TASK\_STK[0],

(CPU\_STK\_SIZE)DIS\_STK\_SIZE/10,

(CPU\_STK\_SIZE)DIS\_STK\_SIZE,

(OS\_MSG\_QTY )0,

(OS\_TICK )0,

(void \* )0,

(OS\_OPT )OS\_OPT\_TASK\_STK\_CHK|OS\_OPT\_TASK\_STK\_CLR,

(OS\_ERR \* )&err);

//创建各操作响应任务

OSTaskCreate((OS\_TCB \* )&RespTASKTCB,

(CPU\_CHAR \* )"RESP test task",

(OS\_TASK\_PTR )resp\_task,

(void \* )0,

(OS\_PRIO )RESP\_TASK\_PRIO,

(CPU\_STK \* )&RESP\_TASK\_STK[0],

(CPU\_STK\_SIZE)RESP\_STK\_SIZE/10,

(CPU\_STK\_SIZE)RESP\_STK\_SIZE,

(OS\_MSG\_QTY )0,

(OS\_TICK )0,

(void \* )0,

(OS\_OPT )OS\_OPT\_TASK\_STK\_CHK|OS\_OPT\_TASK\_STK\_CLR,

(OS\_ERR \* )&err);

OSTmrStart(&tmr1,&err); //开启定时器1

OS\_TaskSuspend((OS\_TCB\*)&StartTaskTCB,&err); //挂起开始任务

OS\_CRITICAL\_EXIT(); //进入临界区

}

void resp\_task(void \*p\_arg)

{

OS\_ERR err;

p\_arg =p\_arg;

ReadUserFlash(0,100);

while(1)

{

if(renewcount>=refresh\_time)

{

renewcount=0;

refresh();

u5\_printf("\r\nrefreshing!\r\n");

}

if(SHUAXIN)

{

SHUAXIN=0;

refresh();

}

UART\_LCD\_Deal();

fz\_data\_deal();

if(up\_up&&nb\_flag)

{

fz\_up();

nb\_flag=0;

}

OSTimeDlyHMSM(0,0,0,70,OS\_OPT\_TIME\_PERIODIC,&err); //延时30ms

}

}

void uart\_dis\_task(void \*p\_arg)

{

OS\_ERR err;

u8 time,timeflag=1;

u8 ok\_air=1,ok\_ds=1;

p\_arg =p\_arg;

while(1)

{

if(datacollect)

{

time++;

if(timeflag)

{

timeflag=0;

rs485\_rec();

send\_carbon\_dioxide();

}

else

{

timeflag=1;

rs485\_rec();

send\_soil();

}

if(time%30==0)

{

time=0;

if(ok\_air)//DHT11温湿度传感器（包括空温和空湿）

{

if(DHT11\_Init()==0)

{

ok\_air=0;

}

else

{

u5\_printf("\r\nDHT11 error！\r\n");

}

}

else

{

dht11\_datadis();

}

if(ok\_ds)//DS18B20温度传感器（水温）

{

if(DS18B20\_Init()==0)

{

ok\_ds=0;

}

else

{

u5\_printf("\r\nDS18B20 error！\r\n");

}

}

else

{

ds18b20\_datadeal();

}

wt\_turbidity();//水浑浊传感器

smogdatadeal();//烟雾传感器

u5\_printf("\r\n传感器ing!\r\n");

}

}

else

{

;

}

OSTimeDlyHMSM(0,0,0,50,OS\_OPT\_TIME\_PERIODIC,&err); //延时30ms

}

}

//NBIoT任务

void NB\_task(void \*p\_arg)

{

OS\_ERR err;

p\_arg =p\_arg;

NB\_Create();

while(1)

{

if(link)

{

up\_up=1;

if(upcount>=upload\_time)

{

upcount = 0;

LED1=!LED1;

u5\_printf("\r\nNBing!\r\n");

Notify\_date();

}

else

{

Deal\_ONEnet\_CMD();

}

}

OSTimeDlyHMSM(0,0,0,30,OS\_OPT\_TIME\_PERIODIC,&err); //延时30ms

}

}

//定时器1的回调函数 20ms

void tmr1\_callback(void \*p\_tmr, void \*p\_arg)

{

TMR\_times++;

reckontime++;

if(reckontime>=50)//1S

{

reckontime=0;

renewcount++;

upcount++;

}

if(TMR\_times>=501)

{

TMR\_times=0;

}

if(U1\_Tim\_en <=3 && U1\_Tim\_en != 0)

{

U1\_Tim\_en++;

if(U1\_Tim\_en==3)

{

U1\_Tim\_en =0;

USART1\_RX\_STA |=1 <<15; //标记接收完成

}

}

if(U3\_Tim\_en <=3 && U3\_Tim\_en != 0)

{

U3\_Tim\_en++;

if(U3\_Tim\_en==3)

{

U3\_Tim\_en =0;

USART3\_RX\_STA |= 1<<15; //标记接收完成

}

}

if(U4\_Tim\_en <= 3 && U4\_Tim\_en != 0)

{

U4\_Tim\_en++;

if(U4\_Tim\_en == 3)

{

U4\_Tim\_en = 0;

USART4\_RX\_STA |= 1<<15; //标记接收完成

}

}

if(U5\_Tim\_en <= 3 && U5\_Tim\_en != 0)

{

U5\_Tim\_en++;

if(U5\_Tim\_en == 3)

{

U5\_Tim\_en = 0;

USART5\_RX\_STA |= 1<<15; //标记接收完成

}

}

if(U6\_Tim\_en <= 3 && U6\_Tim\_en != 0)

{

U6\_Tim\_en++;

if(U6\_Tim\_en == 3)

{

U6\_Tim\_en = 0;

USART6\_RX\_STA |= 1<<15; //标记接收完成

}

}

}