物联网工程设计实验

智能农业物联网系统项目说明书

物联网 1502 班

组长：刘港

组员：林源、任凯、张雅娴

2018-6-28

目录

1、 项目简介

1.1、项目背景

全球信息化进入全面渗透、跨界融合、加速创新、引领发展的新阶段。信息技术更快速度、更广范围、更深程度地引发新一轮科技革命和产业变革。物联网、云计算、大数据等新技术驱动从人人互联向万物互联演进，数字化、网络化、智能化服务将无处不在。现实世界和数字世界日益交汇融合，全球治理体系面临深刻变革。

从“智慧地球”到“感知中国”，“物联网”成为全球瞩目的关键词。被美国列为振兴经济的两大工具之一；被欧盟定位成使欧洲领先全球的基础战略；被中国纳入战略性新兴产业规划重点，实施“互联网+现代农业”行动计划，着力构建现代农业产业体系、生产体系、经营体系。

将“物联网、云计算、大数据”技术与传统农业生产相结合，着力构建现代农业产业体系、生产体系、经营体系，提供“互联网+”时代“智慧农业”解决方案，推动农产品“安全，高质，标准化”生产，提高农业生产智能化、经营网络化、管理数据化、服务在线化水平，促进农业转型升级和农业生产持续增收，为加快农业现代化发展提供强大的创新动力。

1.2、可行性分析

社会因素可行性：

现代高科技农业不仅强调规模化集约化,更要求农作物育、种、管以及与生长环境的自动监控、自动传感、自动控制技术相结合才能实现高效科学生产。“智慧农业”是物联网、云计算、大数据等多种信息技术在农业中综合、全面的应用,实现更完备的信息化基础支撑、更透彻的农业信息感知、更集中的数据资源、更广泛的互联互通、更深入的智能控制、更贴心的公众服务。“智慧农业”与现代生物技术、种植技术等高新技术融合于一体，有利于降低农业生产成本、提高效率，增加农村收益，并保护农村生态环境。

技术可行性：

随着物联网、云计算、大数据技术的发展，物联网技术对农业生产监测控制提供了可行性方案，其低廉的感知成本，自适应的网络拓扑，全面感知和计算等特性，在智能农业领域大放异彩。 云计算、大数据等技术也推进农业管理数字化和现代化，促进农业管理高效和透明，农资采购和农产品流通等数据将会得到实时监测和传递，有效解决信息不对称问题。

经济可行性：

在设计智能农业项目时，采用低廉、可靠、低功耗的传感器节点作为感知，覆盖控制全面，解决了监测控制成本问题。基于软件即服务的 Saas 云计算技术，简化在部署云平台时的繁琐操作，实现一键上云，简单上云。大大减少了生产项目的成本。

1.3、需求分析

需求描述：

1、针对现代农业生产的需求现状，提供了一套先进、全面的整体解决方案，解决方案集生产环境监测、智能设备控制、农业生产指导、和农产品互联网销售于一体。

2、提高农业生产的容灾能力，在面对强大的自然灾害或人为因素时，系统能提供可靠的农业运行机制，减少农业生产的风险，避免或者降低灾害所带来的损失。

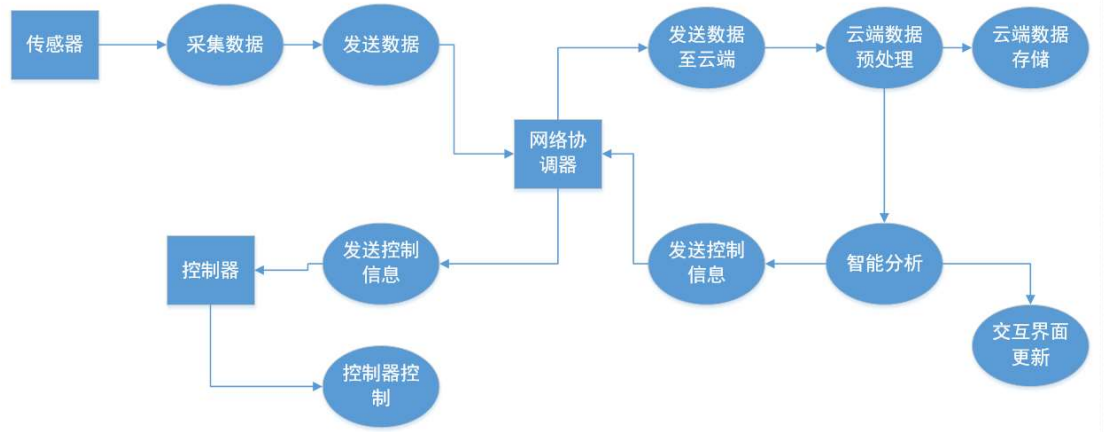
3、系统应为使用者提供良好的交互界面，交互界面要求简单易用。农资采购和农产品流通应实时监测和传递，解决信息不对称问题。

系统需求分析：

基于需求描述，将用户需求分为下面几个部分：

1、 实时、全面、可靠的监测与控制：

通过部署在农业大棚内的各种传感器，全面展示和监测农业大棚的大气环境、土壤环境、水质环境、作物长势、设备运行状态、病虫害情况等，并将监测到的数据传输至云平台，进行存储、分析，并根据环境进行智能控制处理，发出相应的控制信息实现智能控制。并根据状态信息更新交互界面。



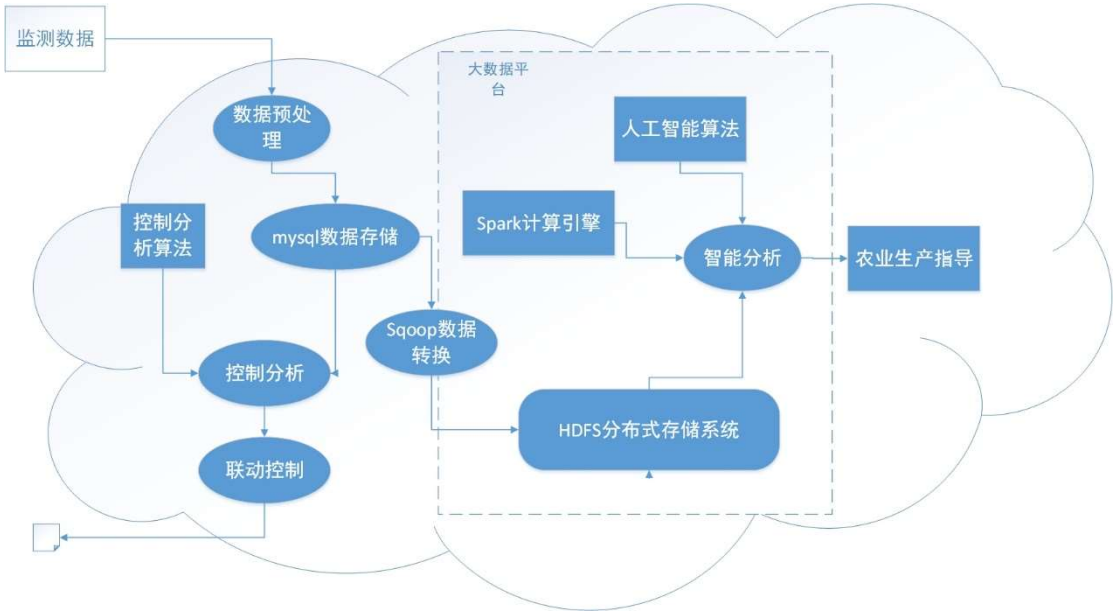
- 传感器感知周围环境进行数据采集，并将数据传至网络协调器。
- 网络协调器将数据发送至云端，云端进行数据预处理操作，并存储在云端数据库。
- 云端对数据进行智能分析，发送控制信息并更新交互界面。
- 网络协调器接收到控制信息并发送给控制器，控制器执行相应的操作。

2、 云计算及大数据技术进行智能分析：

利用云计算技术，采用软件即服务模式（Saas）进行平台搭建，数据集中到云数据中心统一存储与处理。并搭建分布式计算框架进行人工智能分析。

设计符合农业生产过程中的各种传感器、控制器规则策略，建立智能分析策略模型，通过基于传感器监测数据，进行智能分析，实现农业生产的智能预警和联动控制。

基于监测数据和其他可靠数据，通过大数据平台和人工智能算法，对农产品的需求、作物生长影响因素等做出分析，提供基于大数据的准确的农业生产指导。



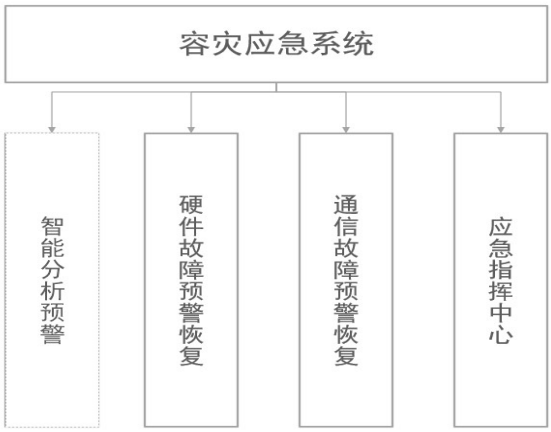
• 监测数据传送至云端，云端进行数据预处理并存储在 MySQL 数据库中。

• MySQL 数据库中的数据首先结合控制分析算法得出相应控制信息进行联动控制。

• 周期性地将 MySQL 数据库数据通过 sqoop 工具加载到 HDFS 分布式文件系统中，结合 spark 计算引擎、人工智能算法，得出相应的农业生产指导。

3、 可靠机制的容灾系统：

构建可靠的容灾应急系统，将应急系统进行分为 3 级，不同级别有不同的应急机制。主要功能分为智能预警功能、硬件故障预警恢复、通信故障预警恢复、应急指挥系统中心等。



智能分析预警：通过监测的数据进行预警分析，如对即将到来的暴雨等自然灾害进行预警并采取相应的容灾措施。

硬件故障预警与恢复：监测硬件状态并进行及时通知维修。如遇一些突发情况，启动相应的应急措施。如断电应启用备用电源和进行数据保护。

通信故障预警恢复：在云端检测数据的接收情况，对长时间未收到数据进行通知预警，遇此情况尽可能将数据保存到本地。

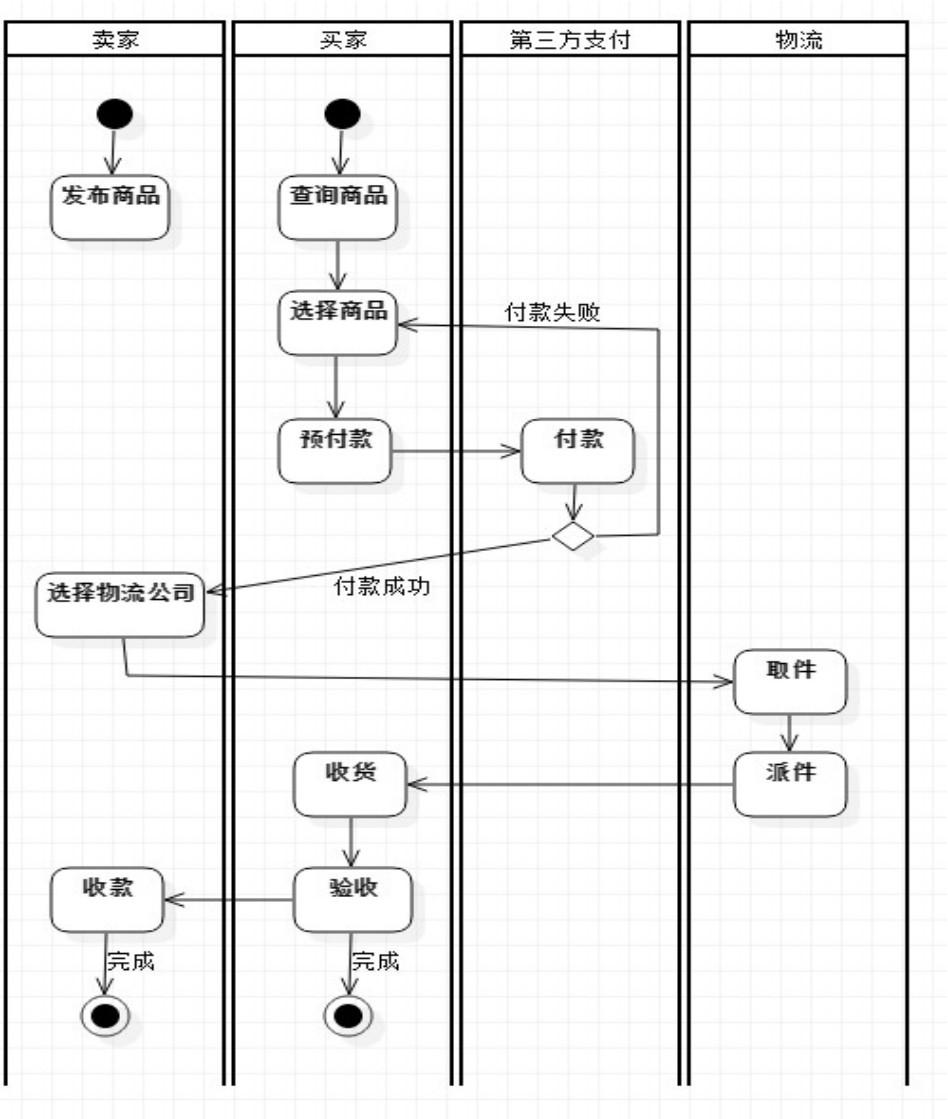
应急指挥中心：对接收到的预警进行人工处理。

4、 用户交互界面：

交互界面分为两个部分，一部分为农业系统的监测及控制状态界面，在本地端、web端、手机端均能进行查看相关的界面。实现多端查看与控制。

第二部分为农资采购和农产品互联网交易界面，实现农资采购和农产品销售在线化，如种子、农具的购买和农产品的销售。设计并实现手机端的交易界面。

第二部分的简要交易系统活动图如下：

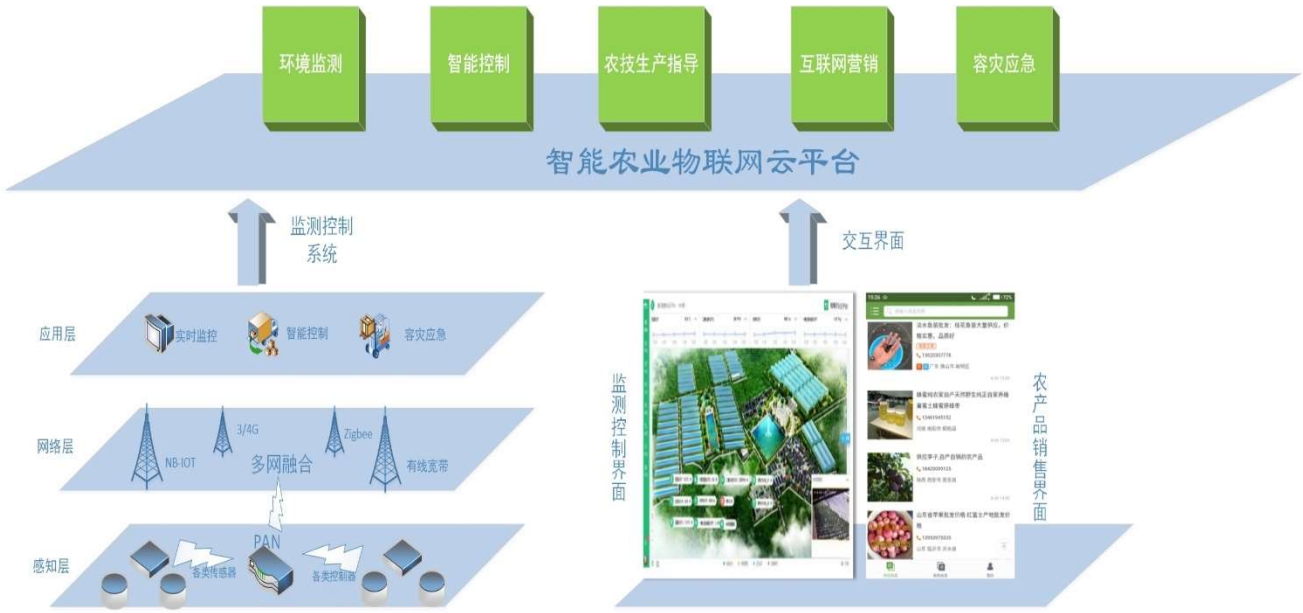


2、 系统设计

2.1、总体方案设计

系统将提供了一套可靠、全面的整体解决方案，解决方案集生产环境监测、智能设备控制、农业生产指导、和农产品互联网销售于一体。

系统总体设计图如下：



智能农业云平台主要功能为：环境监测功能、智能控制功能、基于大数据分析的农技生产指导功能、互联网营销功能和容灾应急功能。云平台同时分为两个部分进行开发：监测控制系统和交互界面系统。

监测控制系统：主要完成实时监控、智能控制、容灾应急系统的功能。进行物体的感知、数据的采集、网络的传输、智能分析与控制等功能的实现。

交互界面系统：主要实现与用户交互的界面。包括多端的监测控制界面和农产品互联网营销界面。

2.2、智能农业物联网平台

2.2.1、监测子系统

监测子系统由传感器监测信息类型、传输协议、数据融合三部分组成。实现对环境的监测并将数据传送至云端服务器。

监测信息类型：

大气环境信息：温湿度、光照强度、风速、气压、蒸发量、CO2 浓度、O2 浓度、NO2 浓度、PM2.5、降雨量等。

土壤环境信息：土壤温度、土壤湿度、土壤 PH 值、土壤含氧量。

水体环境信息：水压、水流量、水质 PH 值、溶氧量。

其他：叶面湿度、作物生长大小、虫灾情况。

传输协议：

采用 Zigbee 与 NB-IOT 结合的传输协议，节点向汇聚节点传输协议采用 zigbee 协议，汇聚节点向网关传输采用 NB-IOT 协议。

数据的传输为 16 位 16 进制传输，格式为：

起始标志	网络标号	数据类型	节点 ID	传输数据值	帧尾

如一条数据为

EE CC 01 03 02 03 29 00

EE CC 为起始标志

01 为网络标号

03 为温湿度传感器数据类型

02 为节点 2 收集的数据

03 29 为节点采集的数据

00 为帧尾

数据融合：

数据融合是将节点对按时序获得的若干监测信息，在汇聚节点进行自动分析、综合，并将融合的数据发送给云端。

2.2.2、控制子系统

控制子系统由控制设备类型、控制方式、控制信息格式三部分组成。实现多种方式对设备进行远端控制。

控制设备类型：

控制设备包括风机、外遮阳设备、内遮阳设备、喷滴灌设备、恻帘、水帘、阀门、加温灯和其他控制设备。

控制信息格式：

控制设备通过协调器发送控制命令，控制命令的格式如下：

起始标志	网络标号	设备类型	节点 ID	控制数据	帧尾

控制方式：

分为循环定时控制、单次人工控制、智能控制三种控制方式。

循环定时控制：实现对设备进行定时控制。如在每早定时地打开遮阳设备。

单次人工控制：通过 APP、本地端或 web 端对设备进行人工控制操作。

智能控制：通过云平台智能分析设备应处于的状态，并实现设备的控制。

2.2.3、云平台分析子系统

云平台分析子系统通过利用云计算技术，采用软件即服务模式(Saas)进行平台的搭建，数据集中到云数据中心统一存储与处理。并搭建分布式计算框架进行人工智能分析。云平台分析子系统主要分为两部分：即基于监测数据的智能控制分析和基于大数据的农业生产指导分析。

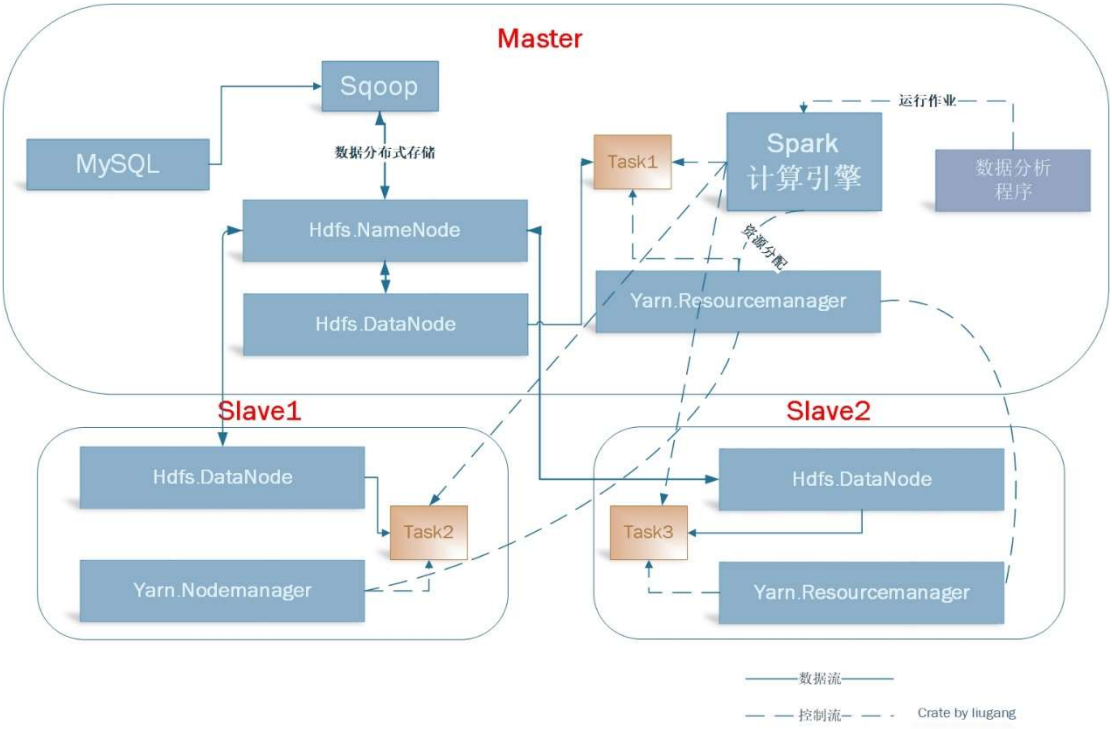
基于监测数据的智能控制分析：

分析各种传感器、控制器规则策略，建立智能分析策略模型，通过基于传感器监测数据，进行智能分析，实现农业生产的智能预警和联动控制。

基于大数据的农业生产指导分析：

基于监测数据和其他可靠数据，通过大数据平台和人工智能算法，对农产品的需求、作物生长影响因素等做出分析，提供基于大数据的准确的农业生产指导。

大数据分析框架如下（以三个云虚拟机为例）：



系统从 mysql 数据库中通过 sqoop 工具将数据加载到 HDFS 分布式文件系统中，通过运行数据分析程序，系统将自动使用 Yarn 进行资源的分配，驱动 spark 计算引擎进行计算分析任务，最后得出农业生产指导信息。

2.2.4、用户交互子系统

2.2.4.1、监测控制状态界面

监测界面由 APP 端、web 端、本地端多端进行展示，与用户进行交互。监测界面分为数据类型和数据展示方式两个部分。

监测界面所监测的信息类型：

大气环境信息：温湿度、光照强度、风速、气压、蒸发量、CO2 浓度、O2 浓度、NO2 浓度、PM2.5、降雨量等。

土壤环境信息：土壤温度、土壤湿度、土壤 PH 值、土壤含氧量。

水体环境信息：水压、水流量、水质 PH 值、溶氧量。

其他：叶面湿度、作物生长大小、虫灾情况。

监测数据展现方式：

分为数值展示和图形展示两个方面。

展示示例：

控制界面由控制设备类型、控制设备状态、控制设备参数设定组成。

控制设备类型：

控制设备包括风机、外遮阳设备、内遮阳设备、喷滴灌设备、侧帘、水帘、阀门、加温灯和其他控制设备。

控制设备状态：

主要有开启状态、关闭状态、亮度状态、喷灌强度状态等。

控制设备参数设定：

主要参数有开启时间、关闭时间、启动状态、持续时间、间隔时间等。

2.2.4.2、互联网销售界面

互联网销售界面采用基于 APP 的交互方式，为使用者提供良好的交互界面，实现农资采购和农产品互联网销售的功能。

互联网销售界面分为三个部分：供应信息、采购信息、账号管理。

供应信息：

针对农产品销售者，即卖家的销售界面。提供蔬菜、水果及其他农副产品的销售界面，为购买者提供供应信息。

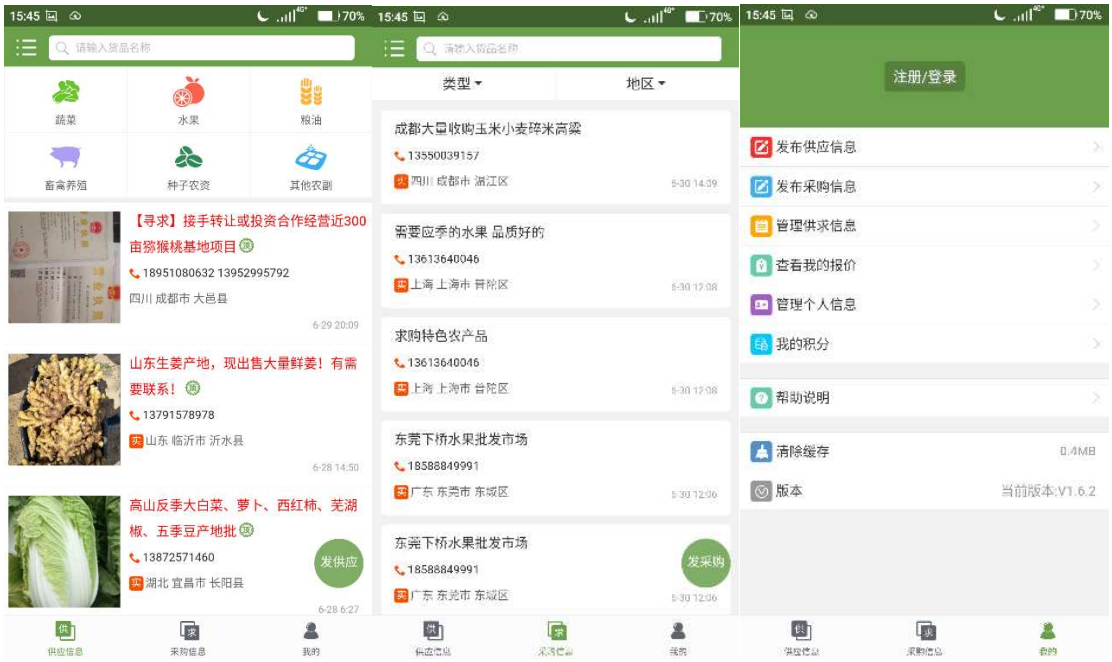
采购信息：

针对农产品的购买者，即买家的购买页面。提供各种类型，各地区的求购信息。

账号管理：

提供用户的注册登录，发布供应、采购信息，管理供求信息的功能。

功能示例如下：



2.2.5、容灾子系统

构建可靠的容灾应急系统，将应急系统进行分为 3 级，不同级别有不同的应急机制。主要功能分为智能预警功能、硬件故障预警恢复、通信故障预警恢复、应急指挥系统中心等。

智能分析预警：通过监测的数据进行预警分析，分为监测信息异常的基础预警和暴雨等自然灾害的预测报警。根据预警采取相应的容灾措施。

硬件故障预警与恢复：监测硬件状态并进行及时通知与维修。

通信故障预警恢复：在云端检测数据的接收情况，对长时间未收到数据进行通知预警，遇此情况尽可能将数据保存到本地。

应急指挥中心：对接收到的预警进行人工处理。

3、 系统功能实现

3.1、监测传感器模块

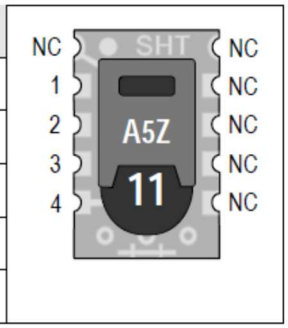
温湿度传感器模块：

1.传感器选型（参考数据手册）

SHT10 的测量范围为：湿度—0~100%RH，温度— -40 ~123.8 ° C。

SHT10 为单片数字温湿度传感器，采用 CMOSens 专利技术将温度湿度传感器、A/D 转换器及数字接口无缝结合，使传感器具有体积小、响应速度快、接口简单、性价比高等特点。其引脚定义如下图所示：

引脚	名称	描述
1	GND	地
2	DATA	串行数据,双向
3	SCK	串行时钟,输入口
4	VDD	电源
NC	NC	必须为空



（1） SHT10 的主要特点如下：

相对湿度和温度的测量兼有露点输出；

全部校准，数字输出；

接口简单（2-wire）,响应速度快；

超低功耗，自动休眠；

出色的长期稳定性；

超小体积（表面贴装）；

测湿精度±4.5%RH，测温精度±0.5℃（25℃）。

2. 温度传感器代码（C 语言版）：

（1）、 void SerialApp_Init(uint8 task_id)

{ ……

P0SEL |= 0x01; //打开 P0_0 口的 ADC 功能

P0DIR &= ~0x01; //将 P_0 设置为输入模式 …… } 函数功能说明：将 P0_0 设置为 ADC.0 模式。

（2）、 uint16 HalAdcRead (uint8 channel, uint8 resolution)

该函数由系统提供,位于 hal_adc.c 中。功能:读取 ADC 转换的结果; 参数:channel, ADC 通道选择, 0 到 7 可选; resolution, ADC 精度选择, 8bit、10bit、12bit、14bit 可选; 返回值: ADC 转换的结果。

（3）采集数据

```
if ( events & TEMPANDHUM_READ_EVT )
{
 Temperature = ReadSHT10(TEMPERATURE); //读取温度
 UART0_Format.Command = 0x01;
 UART0_Format.Data[0] = Temperature>>8;
 UART0_Format.Data[1] = Temperature;
 Humidity = ReadSHT10(HUMIDITY); //读取湿度
 UART0_Format.Data[2] = Humidity>>8;
 UART0_Format.Data[3] = Humidity;
 osal_set_event(SerialApp_TaskID,SERIALAPP_SEND_EVT); //发送 RF 消息
 //周期性启动温湿度采集事件
 osal_start_timerEx(SerialApp_TaskID, TEMPANDHUM_READ_EVT, 5000);
 return ( events ^ TEMPANDHUM_READ_EVT ); }
```

调光模块:

```
#coding=utf8
'''************************************************
*模块: 数字调光模块
*主要功能: 根据光照强度进行自动 LED 调光
************************************************'''
import binascii
import mysqlcon as mc
'''************************************************
*函数: led_light
*相关参数: num 亮度调节值
```

*主要功能： 根据 num 亮度值进行 LED 调光 并存入数据库

**'''

```
def led_light(ser,num):
    senddata='ccee010a01'+num+'00000000000000000ff'
    senddata=binascii.b2a_hex(senddata)
    ser.write(senddata)
    #更新状态
    con=mc.mysqlconnect()
    cursor=con.cursor()
    cursor.execute('update sensorinfo set led=%s where id=0',str(num))
    con.commit()
    cursor.close()
    con.close()
```

协调器相关代码（python 版）：

```
#coding=utf8
'''**********************************************
```

*模块： 协调器

*主要功能： 1.作为其他传感器的网关，与 pc 机进行连接交互

2.采集管理各传感器信息

**'''

```
import serial
import binascii
import mysqlcon as mc
import time
'''**********************************************
```

*函数： CoorConnectPc

*主要功能： 与 PC 机进行端口连接

*相关参数： 端口：COM3

波特率：115200

```
            数据位：8
            停止位：1
            校验位：None
**********************************************'''
def CoorConnectPc():
    ser = serial.Serial(
        port='COM3',
        baudrate=115200,
        bytesize = 8,
        stopbits = 1,
        parity = 'N',
    )
    return ser

'''**********************************************
*函数： ReadPort
*主要功能： 端口数据读取
**********************************************'''
def ReadPort(ser):
    data = ''
    data=str(binascii.b2a_hex(ser.read(16)))
    while 1 :
        if (data)!='':
            print data
            insertlog(data)
            break
    return data

'''**********************************************
*函数： insertlog
```

```
*主要功能：    端口日志记录
**********************************************'''
def insertlog(data):
    con=mc.mysqlconnect()
    cursor=con.cursor()
    cursor.execute('insert into log(addtime,data)
values(%s,%s)',(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),data))
    con.commit()
    cursor.close()
    con.close()
    #linux mysql
    linux_con=mc.linux_mysqlconnect()
    linux_cursor=linux_con.cursor()
    linux_cursor.execute('insert into log(addtime,data)
values(%s,%s)',(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),data))
    linux_con.commit()
    linux_cursor.close()
    linux_con.close()
```

附：其他监测传感器模块请参考相关技术文档。

3.2、控制部件模块

电机模块（python 版）：

```
#coding=utf8
'''**********************************************
*模块：  电机及灯光控制模块
*主要功能：  1.控制电机的转动
             2.控制电机的状态灯光
**********************************************'''
import binascii
import time
import mysqlcon as mc
'''**********************************************
```

```
*函数： op_motor
*主要功能： 电机启动和停止
*相关参数： ser 发送端口连接句柄
           op  选择启动或停止操作
           num 选择电机号
**********************************************'''
def opp_motor(ser,op,num):
    con=mc.mysqlconnect()
    cursor=con.cursor()

    if op=='open':
        senddata='ccee'+num+'090b00000000000000000000ff'
        senddata=binascii.b2a_hex(senddata)
        try:
            ser.write(senddata)
            cursor.execute('update  sensorinfo  set  motor%s_status=%s  where
id=0',(num[:-1],'OPEN'))
            con.commit()
            print 'motor '+num+' status start!'
        except:
            print 'motor '+num+' status close!'

    if op=='close':
        senddata='ccee'+num+'090b00000000000000000000ff'
        senddata=binascii.b2a_hex(senddata)
        try:
            ser.write(senddata)
            cursor.execute('update  sensorinfo  set  motor%s_status=%s  where
id=0',(num[:-1],'CLOSE'))
            con.commit()
            print 'motor '+num+' status start!'
        except:
            print 'motor '+num+' status close!'
    cursor.close()
    con.close()
'''**********************************************
*函数： status_motor
*主要功能： 电机状态灯光控制
*相关参数： ser 发送端口连接句柄
           data 电机回复数据
*功能： 1.当电机回复数据为 EE CC NO 09 DD 09 时
         开启 LED1 灯]
       2.当电机回复数据为 EE CC NO 09 DD 0b 时
         关闭 LED1 灯
```

```
        LED1 灯为电机状态灯，电机开启时为亮状态
**********************************************'''
def status_motor(ser,data):
    num=data[4:6]
    if data[10:12]=='09':
        senddata='ccee'+num+'0901000000000000000000ff'
        senddata=binascii.b2a_hex(senddata)
        ser.write(senddata)
        print 'motor ',num,' light open!'
    if data[10:12]=='0b':
        senddata='ccee'+num+'0902000000000000000000ff'
        senddata=binascii.b2a_hex(senddata)
        ser.write(senddata)
        print 'motor ',num,' light close!'
'''**************************************************
*函数： time_motor
*主要功能： 电机在开启设定的时间后自动关闭
*相关参数： ser 发送端口连接句柄
           num 选择电机
           set_time 自动关闭延迟时间
**************************************************'''
def time_motor(ser,num,set_time):
    opp_motor(ser,'open',num)
    time.sleep(set_time)
    opp_motor(ser,'close',num)
```

附：更多控制模块请参考相关技术文档

3.3、云平台分析模块

大数据平台搭建：

相关软件及版本：

openssh-server-7.4

Hadoop2.8.2

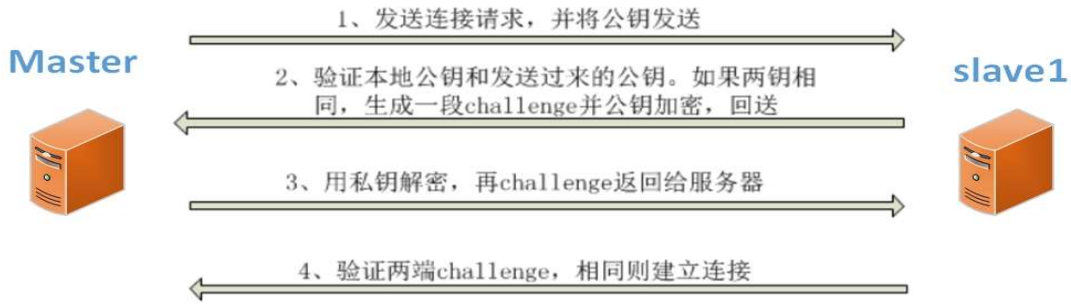
Spark-2.1.2-bin-hadoop2.7

集群部署实现（简略版）

相关源码及过程请访问

https://github.com/efishliu/-

1. ssh 免密登录



同理配置其他节点，是各节点之间能相互免密连接通信

2. **Hadoop 安装**

解压安装包

```
[root@master package]# tar -zxvf hadoop-2.8.2.tar.gz
```

切换至./etc/Hadoop/ 修改配置文件

```
[root@master hadoop]# ls
capacity-scheduler.xml  hadoop-env.cmd            hadoop-policy.xml       httpfs-signature.secret  kms-log4j.properties  mapred-env.sh             slaves                    yarn-env.sh
configuration.xsl       hadoop-env.sh             hdfs-site.xml           httpfs-site.xml          kms-site.xml          mapred-queues.xml.template  ssl-client.xml.example  yarn-site.xml
container-executor.cfg  hadoop-metrics2.properties  httpfs-env.sh           kms-acls.xml             log4j.properties      mapred-site.xml           ssl-server.xml.example
core-site.xml           hadoop-metrics.properties   httpfs-log4j.properties  kms-env.sh               mapred-env.cmd        mapred-site.xml.template  yarn-env.cmd
```

格式化 namenode

```
[root@master hadoop]# hadoop namenode -format
```

启动 hadoop 并查看

```
[root@master sbin]# ./start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/hadoop-2.8.2/logs/hadoop-root-namenode-master.out
master: starting datanode, logging to /usr/hadoop-2.8.2/logs/hadoop-root-datanode-master.out
slave1: starting datanode, logging to /usr/hadoop-2.8.2/logs/hadoop-root-datanode-slave1.out
slave2: starting datanode, logging to /usr/hadoop-2.8.2/logs/hadoop-root-datanode-slave2.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/hadoop-2.8.2/logs/hadoop-root-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/hadoop-2.8.2/logs/yarn-root-resourcemanager-master.out
slave1: starting nodemanager, logging to /usr/hadoop-2.8.2/logs/yarn-root-nodemanager-slave1.out
slave2: starting nodemanager, logging to /usr/hadoop-2.8.2/logs/yarn-root-nodemanager-slave2.out
master: starting nodemanager, logging to /usr/hadoop-2.8.2/logs/yarn-root-nodemanager-master.out
[root@master sbin]# jps
22801 Jps
22049 DataNode
22373 ResourceManager
22217 SecondaryNameNode
22475 NodeManager
21916 NameNode
```

Hadoop 集群中，着重介绍 hdfs 分布式文件系统和 yarn 介绍

即 namenode-datanode resourcemanager-nodemanager 两部分

Mapreduce 部分的数据处理层部分由 spark 替代

3. **Spark 安装和启动**

解压 spark 安装包

```
[root@master package]# tar -zxvf spark-2.1.2-bin-hadoop2.7.tgz
```

配置配置文件，把 Hadoop 配置文件导入 spark-env.sh

启动 spark 并查看

```
[root@master sbin]# ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/spark-2.1.2-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.master.Master-1-master.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark-2.1.2-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave1.out
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark-2.1.2-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave2.out
master: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark-2.1.2-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-master.out
[root@master sbin]# jps
22049 DataNode
22884 Master
22373 ResourceManager
22965 Worker
22217 SecondaryNameNode
22475 NodeManager
21916 NameNode
23022 Jps
```

多元线性回归分析简要源码（python 版）：

```
from pyspark.ml.linalg import Vectors
from pyspark.sql import Row
from pyspark.ml.classification import LogisticRegression

#双变量 Logistic 回归
bdf = sc.parallelize([Row(label=1.0,weight=2.0,features=Vectors.dense(1.0)),Row(label=0.0, weight=2.0, features=Vectors.sparse(1,[],[]))]).toDF()
bdf.show()
blor = LogisticRegression(maxIter=5, regParam=0.01,weightCol='weight')
blorModel = blor.fit(bdf)
blorModel.coefficients
blorModel.intercept

#多元 Logistic 回归
mdf = sc.parallelize([Row(label=1.0,weight=2.0,
features=Vectors.dense(1.0)),Row(label=0.0,weight=2.0,
features=Vectors.sparse(1,[],[])),Row(label=2.,weight=2.0,
features=Vectors.dense(3.0))]).toDF()
mlor=LogisticRegression(maxIter=5,regParam=0.01,weightCol='weight',family='multinomial')
mlorModel = mlor.fit(mdf)
print mlorModel.coefficientMatrix
mlorModel.interceptVector

#模型预测
test0=sc.parallelize([Row(features=Vectors.dense(-1.0))]).toDF()
result = blorModel.transform(test0).head()
result.prediction

result.probability
result.rawPrediction

test1 = sc.parallelize([Row(features=Vectors.sparse(1,[0],[1.0]))]).toDF()
blorModel.transform(test1).head().prediction
```

```
blorModel.transform(test1).show()
#模型评估
blorModel.summary.roc.show()
blorModel.summary.pr.show()
```

3.4、交互界面模块

监测界面：

相关测试代码（python 版）：

```
#coding=utf8
'''**********************************************
*模块： 图形界面
*主要功能： 1.实时显示各个传感器的信息
            2.实时刷新温湿度信息
            3.实时显示传感器曲线
**********************************************'''
import matplotlib
matplotlib.use('TkAgg')
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from  Tkinter import *
import mysqlcon as mc
import pandas as pd

'''**********************************************
*函数： flush
*主要功能： 1.从数据库中读取各传感器状态
            2.将数据进行实时刷新
*相关参数： flag,sleep_flag 触摸开关
            set_temp,set_hum 设定温湿度
            temp,hum,ill 温湿度光照强度
            mo1_st,mo2_st   空调加湿器
**********************************************'''
def
flush(flag,sleep_flag,set_temp,set_hum,temp,hum,ill,mo1_st,mo2_st,led,root):
    con=mc.mysqlconnect()
    realtime_data_sql='select * from sensorinfo'
    realtime_data=pd.read_sql(realtime_data_sql,con)
    con.close()
    flag.set(realtime_data['flag'][0])
    sleep_flag.set(realtime_data['sleep_flag'][0])
    set_temp.set(realtime_data['set_temp'][0])
    set_hum.set(realtime_data['set_hum'][0])
    temp.set(realtime_data['temp'][0])
```

```
    hum.set(realtime_data['hum'][0])
    ill.set(realtime_data['ill'][0])
    mo1_st.set(realtime_data['motor1_status'][0])
    mo2_st.set(realtime_data['motor2_status'][0])
    led.set(realtime_data['led'][0])
    #500ms 刷新数据

root.after(500,flush,flag,sleep_flag,set_temp,set_hum,temp,hum,ill,mo1_st,mo2_st,led,root)

'''**********************************************
*函数： draw
*主要功能： 绘制并刷新温湿度和光照强度曲线
*相关参数： root 图形根界面
**********************************************'''
def draw(root):
    #读取温湿度光照强度数据 10 条
    con=mc.mysqlconnect()
    recent_10_temphum_data_sql='select temp,hum,addtime from temp_hum group by id desc limit 10'
    recent_10_temphum_data=pd.read_sql(recent_10_temphum_data_sql,con)
    recent_10_ill_data_sql='select illumination,addtime from photores group by id desc limit 10'
    recent_10_ill_data=pd.read_sql(recent_10_ill_data_sql,con)
    con.close()
    #创建画板并进行图形绘制
    f = Figure(figsize=(10,4), dpi=100)
    temp_photo=f.add_subplot(221)
    hum_photo=f.add_subplot(222)
    ill_photo=f.add_subplot(223)

    th_x=recent_10_temphum_data['addtime']

temhum_x=[th_x[9],th_x[8],th_x[7],th_x[6],th_x[5],th_x[4],th_x[3],th_x[2],th_x[1],th_x[0]]

    ty=recent_10_temphum_data['temp']
    temp_y=[ty[9],ty[8],ty[7],ty[6],ty[5],ty[4],ty[3],ty[2],ty[1],ty[0]]

    hy=recent_10_temphum_data['hum']
    hum_y=[hy[9],hy[8],hy[7],hy[6],hy[5],hy[4],hy[3],hy[2],hy[1],hy[0]]

    ix=recent_10_ill_data['addtime']
    i_x=[ix[9],ix[8],ix[7],ix[6],ix[5],ix[4],ix[3],ix[2],ix[1],ix[0]]
```

```
    iy=recent_10_ill_data['illumination']
    i_y=[iy[9],iy[8],iy[7],iy[6],iy[5],iy[4],iy[3],iy[2],iy[1],iy[0]]

    #设置横纵坐标
    temp_photo.plot(th_x,temp_y)
    hum_photo.plot(th_x,hum_y)
    ill_photo.plot(i_x,i_y)

    temp_photo.set_xlabel('time')
    temp_photo.set_ylabel('temp')
    hum_photo.set_xlabel('time')
    hum_photo.set_ylabel('hum')
    ill_photo.set_xlabel('time')
    ill_photo.set_ylabel('illumination')

    #图像展示
    temp_photo.grid()
    hum_photo.grid()
    ill_photo.grid()
    dataPlot = FigureCanvasTkAgg(f, master=root)
    dataPlot.show()
    dataPlot.get_tk_widget().pack()
    #root.after(500,draw,root)

'''**********************************************
*函数： windows
*主要功能： 主窗口
**********************************************'''
def windows():
    root=Tk()
    root.title('农业检测')
    #root.geometry("400x300")

    #设置状态更新
    flag=StringVar()
    sleep_flag=StringVar()
    set_temp=StringVar()
    set_hum=StringVar()
    mo1_st=StringVar()
    mo2_st=StringVar()
    temp=StringVar()
    hum=StringVar()
    ill=StringVar()
```

```
led=StringVar()


Label(root,  text='基于 Zigbee 的智能农业大棚环境检测系统',font=('宋体', 20)).pack(pady=10)

fm_flag_set=Frame(root)

flaglabel_name=Label(fm_flag_set,text='总开关',font=('宋体', 15))
flaglabel_name.pack(side=LEFT,pady=10)
flaglabel_val=Label(fm_flag_set,textvariable=flag,font=('宋体', 15))
flaglabel_val.pack(side=LEFT,padx=30)

sleep_flaglabel_name=Label(fm_flag_set,text='节能模式开关',font=('宋体', 15))
sleep_flaglabel_name.pack(side=LEFT,pady=10)
sleep_flaglabel_val=Label(fm_flag_set,textvariable=sleep_flag,font=('宋体', 15))
sleep_flaglabel_val.pack(side=LEFT,padx=30)

mo2_stlabel_name=Label(fm_flag_set,text='空调状态',font=('宋体', 15))
mo2_stlabel_name.pack(side=LEFT,pady=10)
mo2_stlabel_val=Label(fm_flag_set,textvariable=mo2_st,font=('宋体', 15))
mo2_stlabel_val.pack(side=LEFT,padx=30)

mo1_stlabel_name=Label(fm_flag_set,text='加湿器状态',font=('宋体', 15))
mo1_stlabel_name.pack(side=LEFT,pady=10)
mo1_stlabel_val=Label(fm_flag_set,textvariable=mo1_st,font=('宋体', 15))
mo1_stlabel_val.pack(side=LEFT,padx=30)
fm_flag_set.pack(side=TOP,pady=10)

#设定温湿度显示
fm_flag_set2=Frame(root)
set_templabel_name=Label(fm_flag_set2,text='设定温度阈值',font=('宋体', 15))
set_templabel_name.pack(side=LEFT,pady=10)
set_templabel_val=Label(fm_flag_set2,textvariable=set_temp,font=('宋 体 ', 15))
set_templabel_val.pack(side=LEFT,padx=30)

set_humlabel_name=Label(fm_flag_set2,text='设定湿度阈值',font=('宋体', 15))
set_humlabel_name.pack(side=LEFT,pady=10)
set_humlabel_val=Label(fm_flag_set2,textvariable=set_hum,font=('宋体', 15))
set_humlabel_val.pack(side=LEFT,padx=30)
fm_flag_set2.pack(side=TOP,pady=10)
```

#温度-湿度-光照强度模块实时展示

```
fm1=Frame(root)
templabel_name=Label(fm1,text='温度',font=('宋体', 15))
templabel_name.pack(side=LEFT,pady=10)
templabel_val=Label(fm1,textvariable=temp,font=('宋体', 15))
templabel_val.pack(side=LEFT,padx=30)

humlabel_name=Label(fm1,text='湿度',font=('宋体', 15))
humlabel_name.pack(side=LEFT,pady=10)
humlabel_val=Label(fm1,textvariable=hum,font=('宋体', 15))
humlabel_val.pack(side=LEFT,padx=30)

illlabel_name=Label(fm1,text='光照强度',font=('宋体', 15))
illlabel_name.pack(side=LEFT,pady=10)
illlabel_val=Label(fm1,textvariable=ill,font=('宋体', 15))
illlabel_val.pack(side=LEFT,padx=30)

ledlabel_name=Label(fm1,text='LED灯亮度等级',font=('宋体', 15))
ledlabel_name.pack(side=LEFT,pady=10)
ledlabel_val=Label(fm1,textvariable=led,font=('宋体', 15))
ledlabel_val.pack(side=LEFT,padx=30)
fm1.pack(side=TOP,pady=15)
```

#刷新各传感器状态和图像

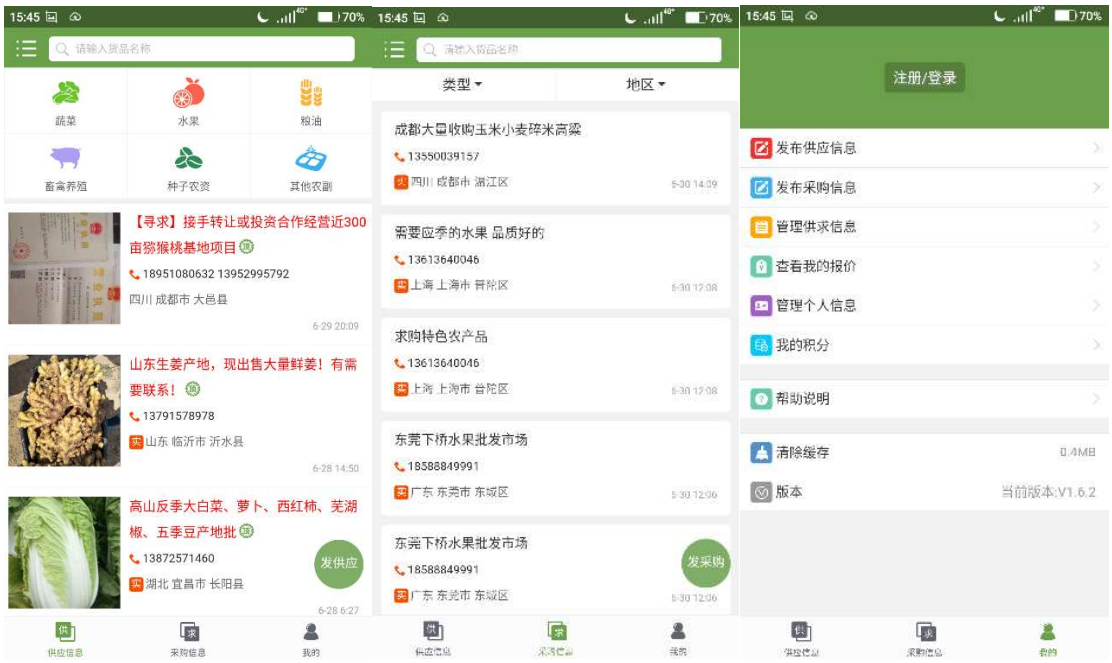
```
flush(flag,sleep_flag,set_temp,set_hum,temp,hum,ill,mo1_st,mo2_st,led,root)
draw(root)
root.update_idletasks()
root.mainloop()
```

附：更多代码请参考相关技术文档

效果图：

控制界面：

互联网营销界面：

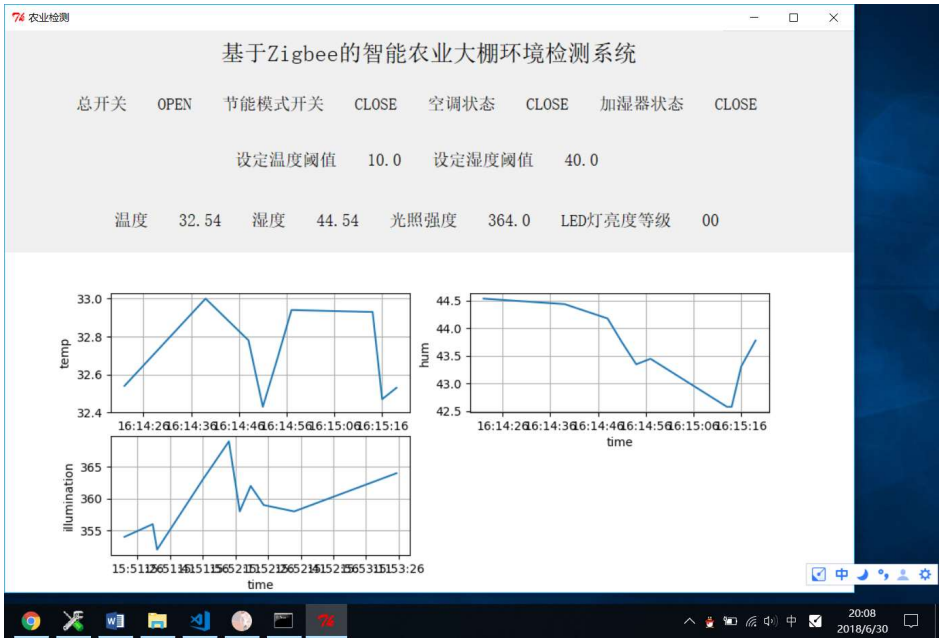


3.5、容灾模块

附：容灾模块实现请参考项目二期技术文档

4、 系统集成与测试

将监测、控制、容灾系统、监测控制界面进行集成。测试单元模块效果如下：



数据库存储接收数据

```
mysql> select * from sensorinfo;
+------+------+------------+----------+---------+-------+-------+------+---------------+---------------+------+
| id   | flag | sleep_flag | set_temp | set_hum | temp  | hum   | ill  | motor1_status | motor2_status | led  |
+------+------+------------+----------+---------+-------+-------+------+---------------+---------------+------+
|    0 | OPEN | CLOSE      |       10 |      40 | 32.54 | 44.54 |  364 | CLOSE         | CLOSE         | 00   |
+------+------+------------+----------+---------+-------+-------+------+---------------+---------------+------+
1 row in set (0.02 sec)

mysql>
```

```
| 1049 | 32.56 | 43.81 | 2018-06-22 16:14:16 |
| 1050 | 32.56 | 43.81 | 2018-06-22 16:14:19 |
| 1051 | 32.53 | 43.78 | 2018-06-22 16:14:22 |
| 1052 | 32.47 | 43.31 | 2018-06-22 16:14:39 |
| 1053 | 32.93 | 42.58 | 2018-06-22 16:14:48 |
| 1054 | 32.93 | 42.58 | 2018-06-22 16:14:51 |
| 1055 | 32.94 | 43.45 | 2018-06-22 16:14:54 |
| 1056 | 32.68 | 43.35 | 2018-06-22 16:14:57 |
| 1057 | 32.43 | 43.75 | 2018-06-22 16:15:13 |
| 1058 | 32.78 | 44.18 | 2018-06-22 16:15:14 |
| 1059 |    33 | 44.44 | 2018-06-22 16:15:16 |
| 1060 | 32.54 | 44.54 | 2018-06-22 16:15:19 |
+------+-------+-------+---------------------+
937 rows in set (0.08 sec)
```

附：其他集成、测试请参考集成文档和测试文档

5、 附录

5.1、相关术语说明

Sass: SaaS 是 Software-as-a-Service（软件即服务）的简称，随着互联网技术的发展和应用软件的成熟， 在 21 世纪开始兴起的一种完全创新的软件应用模式。它是一种通过 Internet 提供软件的模式，厂商将应用软件统一部署在自己的服务器上，客户可以根据自己实际需求，通过互联网向厂商定购所需的应用软件服务，按定购的服务多少和时间长短向厂商支付费用，并通过互联网获得厂商提供的服务。用户不用再购买软件，而改用向提供商租用基于 Web 的软件，来管理企业经营活动，且无需对软件进行维护，服务提供商会全权管理和维护软件，软件厂商在向客户提供互联网应用的同时，也提供软件的离线操作和本地数据存储，让用户随时随地都可以使用其定购的软件和服务。

sqoop: Sqoop(发音：skup)是一款开源的工具，主要用于在 Hadoop(Hive)与传统的数据库(mysql、postgresql...)间进行数据的传递，可以将一个关系型数据库（*例如：MySQL ,Oracle ,Postgres 等*）中的数据导进到 Hadoop 的 HDFS 中，也可以将 HDFS 的数据导进到关系型数据库中。

HDFS: Hadoop 分布式文件系统(HDFS)被设计成适合运行在通用硬件(commodity hardware)上的分布式文件系统。它和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS 是一个高度容错性的系统，适合部署在廉价的机器上。HDFS 能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。HDFS 放宽了一部分 POSIX 约束，来实现流式读取文件系统数据的目的。HDFS 在最开始是作为 Apache Nutch 搜索引擎项目的基础架构而开发的。HDFS 是 Apache Hadoop Core 项目的一部分。

Spark: Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 是 UC Berkeley AMP lab (加州大学伯克利分校的 AMP 实验室)所开源的类 Hadoop MapReduce 的通用并行框架，Spark，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是——Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

Zigbee: ZigBee 是基于 IEEE802.15.4 标准的低功耗局域网协议。根据国际标准规定，ZigBee 技术是一种短距离、低功耗的无线通信技术。其特点是近距离、低复杂度、自组织、低功耗、低数据速率。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。简而言之，ZigBee 就是一种便宜的，低功耗的近距离无线组网通讯技术。ZigBee 是一种低速短距离传输的无线网络协议。

NB-IOT: NB-IoT 是 IoT 领域一个新兴的技术，支持低功耗设备在广域网的蜂窝数据连接，也被叫作低功耗广域网(LPWAN)。NB-IoT 支持待机时间长、对网络连接要求较高设备的高效连接。据说 NB-IoT 设备电池寿命可以提高至至少 10 年，同时还能提供非常全面的室内蜂窝数据连接覆盖。

ssh: SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议。SSH 是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。

hadoop: Hadoop 实现了一个分布式文件系统（Hadoop Distributed File System），简称

HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS 放宽了（relax）POSIX 的要求，可以以流的形式访问（streaming access）文件系统中的数据。

yarn: Apache Hadoop YARN （Yet Another Resource Negotiator，另一种资源协调者）是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。

多元线性回归：在回归分析中，如果有两个或两个以上的自变量，就称为多元回归。事实上，一种现象常常是与多个因素相联系的，由多个自变量的最优组合共同来预测或估计因变量，比只用一个自变量进行预测或估计更有效，更符合实际。因此多元线性回归比一元线性回归的实用意义更大。

5.2、小组分工说明

学号	姓名	分工模块
2015003588	刘港	
	林源	
	任凯	
	张雅娴	