摘 要

随着近年来云计算产业的飞速发展，国内数据中心的规模和数量快速增长，然而却伴随着机房设计不完善，热流通性差，运维成本上升等诸多问题。在这种前提下，为了保证服务的高可用性，数据中心必须使其中的空调系统全年以极高的负荷运行，产生了大量的能源浪费。同时，由于机房内服务器发热密度大且布置分散，房间内单独的温湿度监测点无法真实的反映机房内环境情况，若产生局部高温、高湿无法及时报警，可能导致严重的后果。本文针对上述问题，设计并实现了1套数据中心环境监测系统。

本课题在对数据中心环境监测的需求进行分析后将该系统划分为分布式监测节点，汇聚节点和上位机3个部分，并分别进行了设计和实现。测试表明，该系统能够很好的完成数据中心环境监测的任务。具体内容如下：

1. 设计了分布式监测节点和汇聚节点PCB，并完成了焊接和电路调试。

2. 基于2种嵌入式操作系统实现了监测节点和汇聚节点的程序设计。

3. 设计并测试了2种节点间的3层通讯协议，并对其性能做出了比较。

4. 基于Niagara框架设计了可视化界面，实现了对机房内温湿度的实时监控。

# 绪论

## 背景及意义

伴随着云计算，大数据产业的高速增长，新兴的5G，物联网，AI等技术快速演进，对数据中心的质量要求也逐年升高。从国际上看，截至2017年底，全球数据中心共44.4万个，由于大型化可显著的降低成本，全球数据中心数量近几年成下降趋势，预计到2020年将减少至 42.2 万个，同时，单个机架功率快速提升，总机架数量平稳增长，预计到2020年机架数量将达到493.3万架。相较而言，我国国内数据中心的规模和数量一直保持了较快的增长趋势，截至2017年底，我国在用数据中心总体规模166万架，比2016年增长10%，其中，大型或超大型数据中心为增长主力，数量比2016年增长68%。

随着单个数据中心规模的不断扩张，对其运维系统的要求也逐年上升，然而，由于我国数据中心起步较晚，其维护平台配置较为落后，维护人员经验较少，与国际先进水平还存在一定差距，一般来说，国内每个运维平均能够同时管理100个机柜，而国际先进的数据中心平均每个运维能够同时管理上千个机柜。

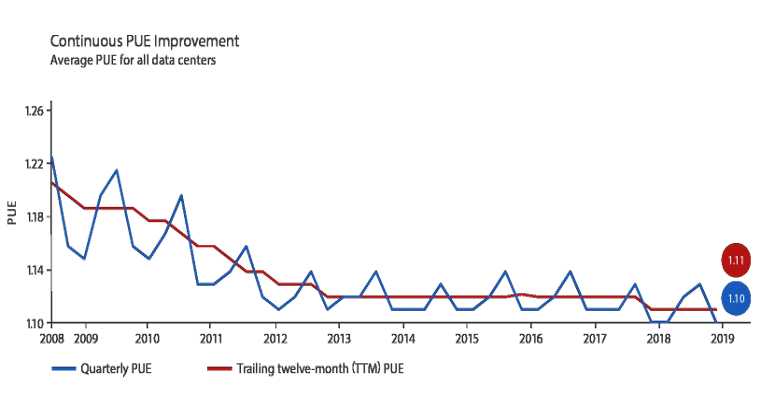
由于机单个服务器发热量大且机房内机柜密度高，机房内空气流动较差，温度分布不均匀，为了避免产生局部高温，使机房内的所有机柜都能够及时冷却，其空调系统需要全年以很高的负荷运行，这就造成了很多不必要的能源浪费。在2017年的数据中, 我国大型数据中心中对设备进行冷却所耗能约占总耗能的40%，对于中小型数据中心该占比甚至能达到60%以上。

由此可见，较为落后的服务器机房的监控设施和巨大的能源浪费严重制约了我国数据中心产业的发展，对此，我国在2019年由工业和信息化部，国家机关事务管理局，国家能源局联合发布了关于加强绿色数据中心建设的指导意见，该意见指出，我国在未来几年中，将着力打造高效，清洁的绿色数据中心，大力推动绿色数据中心的创建，运维和改造，到2022年，将数据中心的平均能耗降到国际先进水平，并实现运维水平的大幅提升。

因此，对数据中心机房内环境进行实时监测，不仅可以提升运维水平，降低运维成本，而且可以利用收集到的数据指导新建绿色数据中心的设计，具有重要的研究意义和应用价值。

## 国内外研究现状

如何提高数据中心运维水平，降低能耗，国际上的数据中心领域研究的热点问题。国际领先的如谷歌公司最新的数据表明，其全球数据中心的PUE（全部耗电量与用于计算的耗电量比值）平均值已经达到1.1，亚马逊旗下数据中心的PUE值（https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/）平均值为1.2，这些公司大多使用定制的服务器机架，并使用水冷技术，国内亦有相关研究（[1]诸凯,刘泽宽,何为,柴祥.数据中心服务器CPU水冷散热器的优化设计[J/OL].制冷学报,2019(02):1-7[2019-04-11]），



在对数据中心房间内环境监控方面，【翻译的这个】使用Arduino单片机套件设计了一种温湿度监控网络，实现了对机房内不同服务器的实时监控，但是该系统传感器节点体积较大，部署困难，且功耗较高，无法进行实际的应用。

【基于阵列传感器的数据中心温度场可视化系统研究\_沈华峰】对采用阵列传感器采集数据中心中的温度数据，并设计实现了一套数据中心温度场可视化系统，该系统可以对采集到的数据进行分析和可视化，有助于验证模拟结果和实际结果的差异。该系统重点在数据的分析和可视化方向，对数据的收集方向工作量较少，也无法进行实际应用，且所有数据是先收集再建模，无法达到实时性监控的要求。

【基于Ganglia的数据中心监控平台设计\_尤海鹏】设计了一套数据中心监控平台，对数据中心基础设施，空调系统和电力供应系统进行了一些整合，简化了数据中心的运维难度，但是在房间温湿度监测上还是使用了空调本身上传的温湿度数据，并不能真实的反映房间内的分湿度分布状况。

在已有的研究中，工作大都着眼于理论上的验证，其系统并不能很好的应用于实际的机房中，或者不能真实反映机房内的环境，本课题针对上述问题，设计了一套易部署，低成本，低功耗，高分辨率的数据中心环境检测系统。

## 主要研究内容

本课题的核心为设计一种数据中心环境状态监测系统。上位机通过收集分散在房间内的监测节点的监测数据，来实现房间内温湿度状态的实时监测。通过收集分析这些数据，可以更好的了解机房内的分湿度变化情况，或实现对房间内空调系统的实时控制，从而降低数据中心整体能耗

本课题使用软件Altium Designer设计了用于数据监测和汇聚的监测节点和汇聚节点的PCB，使用Keil软件通过C/C++对其编程，实现了房间内温湿度的监测。

此外，还设计了两种汇聚节点与监测节点间的3层通讯协议，来实现监测节点与汇聚节点之间的数据传输。

在上位机方面，本课题使用Java对Niagara进行编程，使其通过modbus协议读取汇聚节点的数据，对数据进行可视化后实现人机交互界面。

## 论文结构安排

本文共9章，具体内容安排如下：

1. 3本章首先分析了国内外数据中心发展的趋势，指出了在国内数据中心数量和规模蓬勃发展的过程中，出现了运维水平较低，能源浪费严重等问题，引出了数据中心环境监测的必要性，之后分析了目前已有的对数据中心环境监测工作，指出了他们的优势和缺陷，最后总结了本课题核心的研究工作以及论文的结构安排

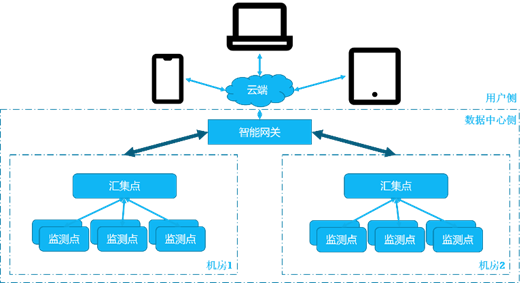
2.

# 总体设计方案

本课题由下位机和上位机两大部分组成,其中下位机主要负责数据中心房间内温湿度数据的采集和汇总,上位机主要负责对采集的数据进行分析处理以及提供用户可视化接口等工作。

在下位机部分的设计中存在两类设备，分别为监测节点以及汇聚节点，其中，监测节点负责对节点所处位置的环境状态进行实时监测和上传，汇聚节点负责汇集监测节点收集到的环境数据,进行初步处理后提供Modbus接口供上位机设备查询。

以下为本课题的系统架构：



# 监测节点的设计

## 监测节点的功能需求

在本课题中，监测节点会被分散的放置在需要被监测的数据中心房间内，用以测量房间内不同空间位置的温湿度数据。此外，监测节点还需要将测量的温湿度数据以一定频率发送到汇聚节点中。根据以上需求，监测节点需要具备以下几个特点：

1. 监测节点应有足够小的体积，从而可以方便的被安放在房间内的各个角落以及服务器机架狭窄的空间中

2. 监测节点应有足够小的的功耗，能仅依靠小型的锂电池工作较长时间，以降低如更换电池等操作的维护成本。

3. 监测节点应有较高的测量精度，能够真实的反映其所在位置的温湿度数据

4. 监测节点应能够以较高的频率及较低的误码率向汇聚节点更新其所测量的温湿度数据

5. 监测节点应有唯一且可更改的编号,以便将监测节点与其对应空间坐标进行映射

## 监测节点的硬件设计与实现

### 监测节点硬件设计总体方案

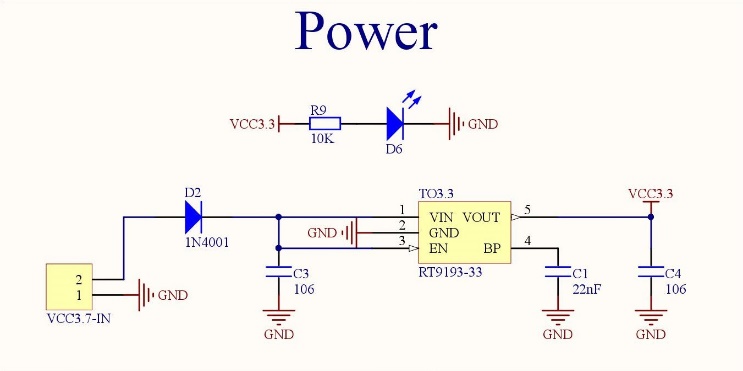
在监测节点的硬件设计中，本课题在使用3.3V稳压电路作为电源供电的基础上,采用ST公司生产的STM32微控制器为控制中枢,连接拨码开关,传输模块,传感器模块以完成所需的各个功能。



### 监测节点电源设计

考虑到监测节点需要较小的体积，采用大小为7mm\*20mm\*30mm的小型锂电池进行供电，能够提供的标称电压3.7v约550mA\*h的电量。

由于监测节点的所有用电模块均能使用3.3V的电压进行供电，所以本设计采用RT9193-3.3超低压差, 超快速线性稳压芯片对锂电池进行稳压，，该芯片具有极低的噪声和功耗，实践表明，该芯片能够满足监测节点节点各个模块的用电需求。

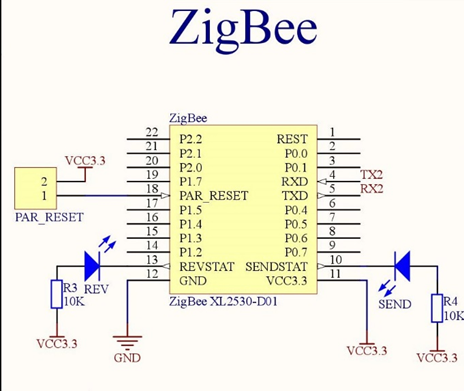


### 监测节点传输模块设计

本课题采用基于ZigBee2007/PRO协议栈的传输模块XL2530-232AP1进行数据的通讯。在配置完成后，该模块可以进行自动组网，并可以在物理层上实现点对点和广播透传

在标称的3.3V电压下，其接收电流和发射电流仅为27mA和28mA，非常适合于作为对功耗敏感监测节点的传输模块。

在电路设计中，该模块通过串口与微处理器连接，来接收控制信息和数据。此外，为了便于观察，本课题设计了了两个LED，可以分别在模块进行发送数据以及接收数据是进行闪烁。



### 监测节点微处理器电路设计

考虑到监测节点需要较小的体积以及较低的能耗,本课题采用了ST公司的STM32F1C8T6型芯片,在休眠模式下，其电流仅有11mA

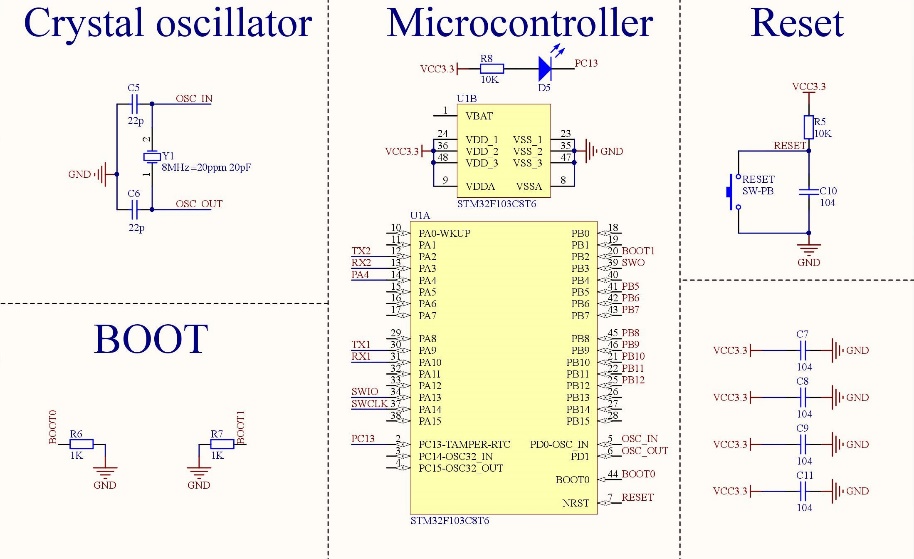
为使用该芯片，需要实现其必要的辅助电路，包括：

1. 频率为8MHz的无源晶振提供芯片运行所需的时钟

2. 选择芯片方式的BOOT0和BOOT1被设置为接地，即从Flash启动

3. 用于芯片复位的复位电路，并具有上电复位和手动复位的功能

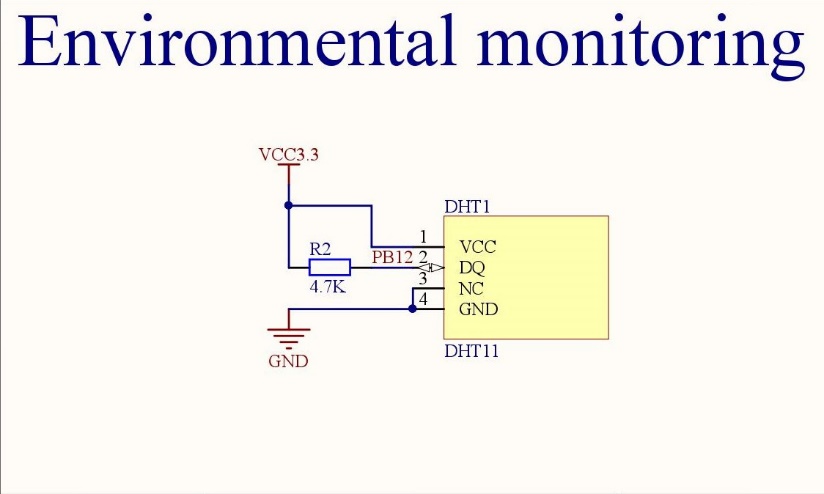
4. 用于下载程序的SWD调试接口



### 监测节点传感器的选型及设计

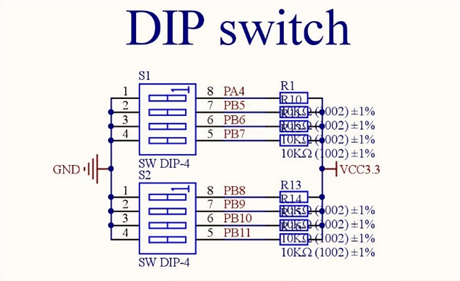
本课题选用DHT11型温湿度传感器,该传感器是一款在出厂时已被校准的数字信号输出的温湿度传感器。其温度测量精度为±2℃，湿度测量精度为5%。

其电路设计较为简单，按照数据手册，在保证3V~5V供电的前提下，仅需要在数据DQ引脚上接5K左右的上拉电阻即可。

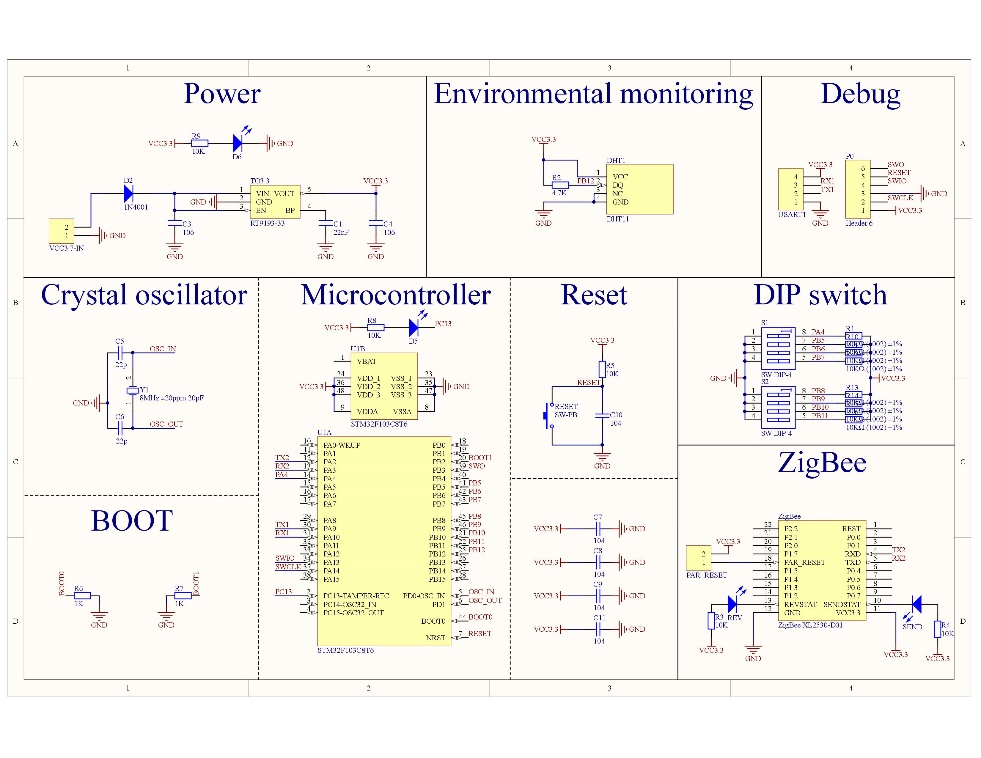


### 监测节点唯一ID的设计

由于传感节点是分散在房间中放置,为了能够分辨不同监测节点,每个节点必须被分配一个唯一的标识。本课题在每个监测节点上放置2个4位拨码开关进行标记,最多能够标识255个节点,能够满足监测的要求。

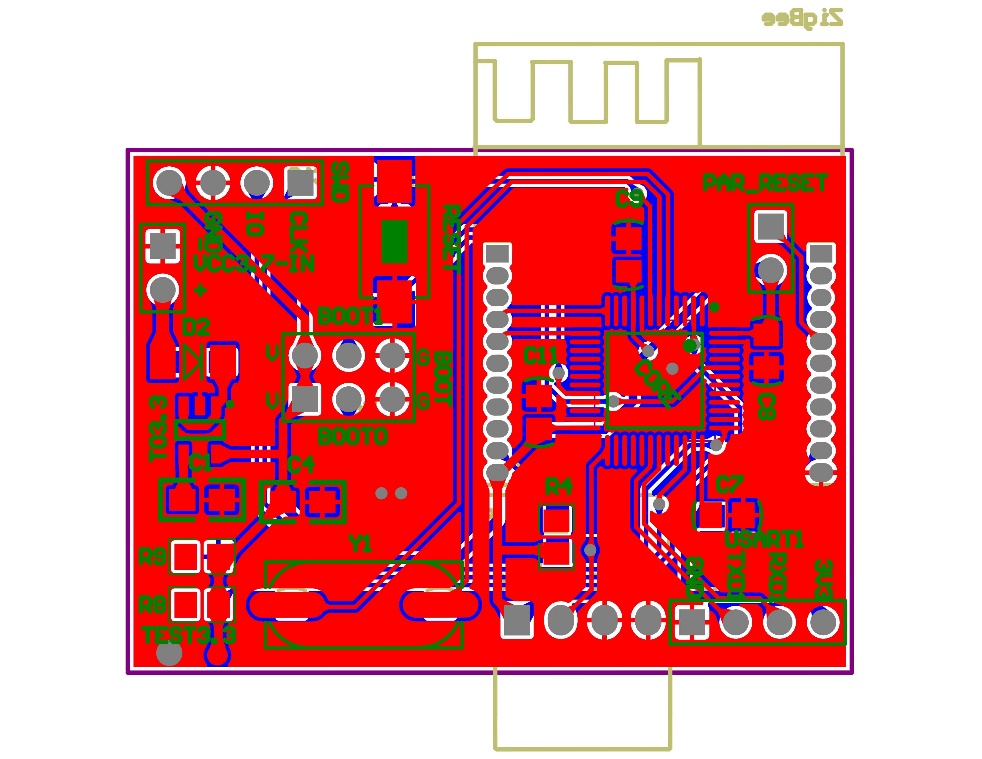


### 监测节点PCB设计方案



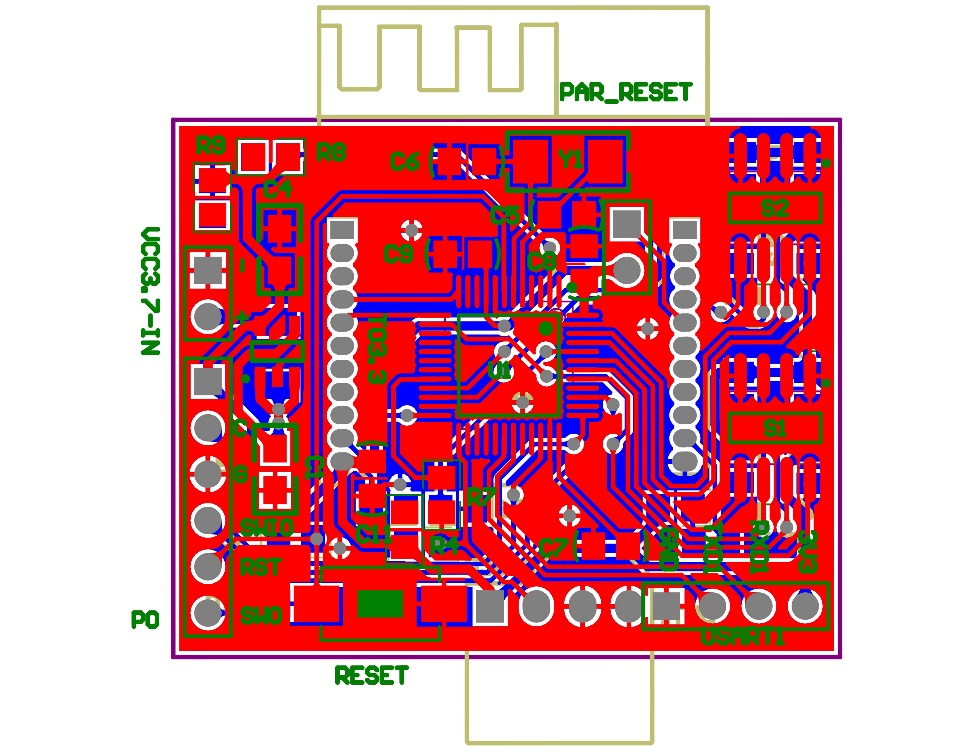
对于监测节点，其最主要的设计需求就是较小的体积，因此，在元件的选型上，监测节点均使用了尽量小的封装，采用双面板设计，有效的减小了监测节点电路板的大小。

在监测节点PCB的设计过程中，曾因考虑到DIP开关占用较大的体积，故取消了该模块的设计，其PCB成品大小位42mm\*30mm：

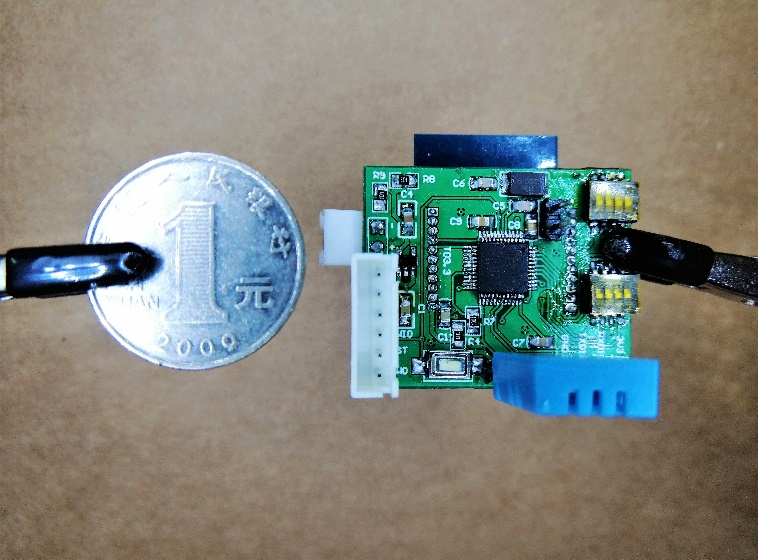


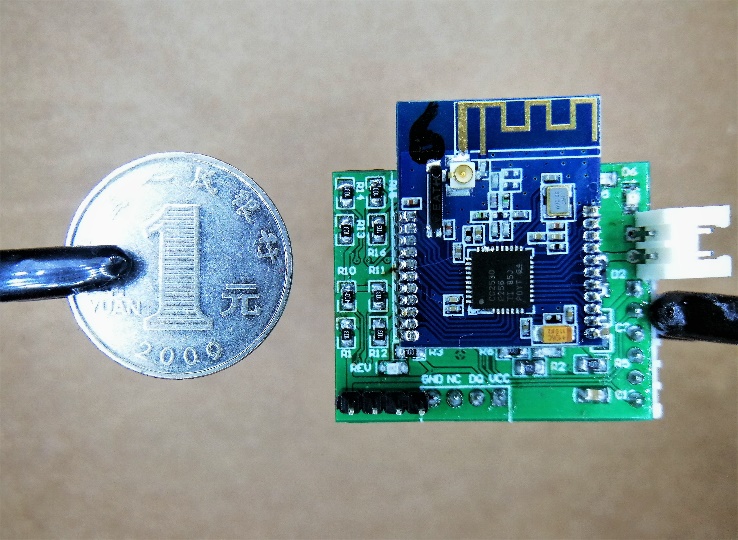
然而在之后的测试中发现，取消DIP开关后，由于无法方便的更改监测节点的编号，调试较为困难，且在实际应用中，该功能使用频率较高，故在后期的设计中恢复了DIP开关部分。

经重新设计和改良后，最终的监测节点PCB大小仅为40mm\*35mm。



在采购元件并完成焊接工作后，实物图如下：





## 监测节点软件设计

本章主要介绍在监测点上微处理器运行的程序的设计。

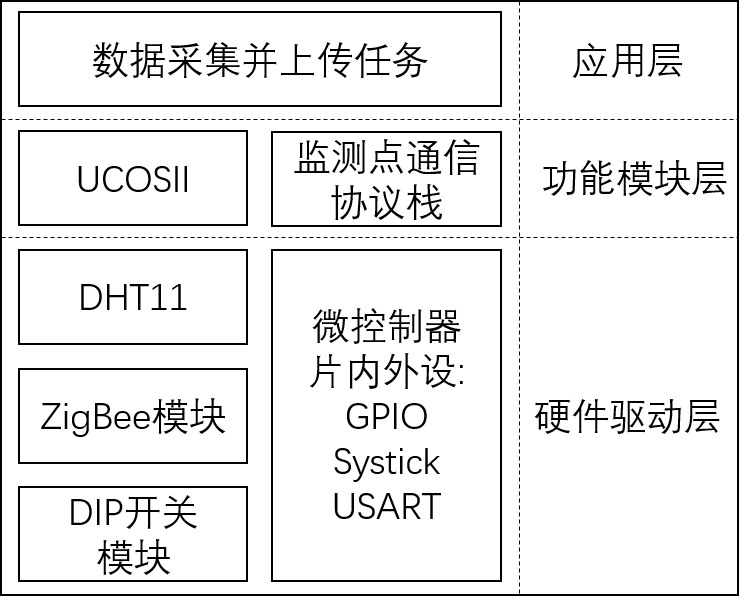
### 监测节点程序架构

监测点的程序架构分为硬件驱动层，功能模块层和应用层三层。

1. 硬件驱动层的主要作用为初始化监测点所需的各种硬件模块并为上层调用提供接口。如实现温湿度传感器模块DHT11和片内外设的初始化，提供ZigBee模块，DIP开关等外部模块的接口等

2. 功能模块层的主要作用为通过UCOS2操作系统提供任务调度，消息队列，延时功能等接口，通过监测点协议栈提供监测点向汇聚点发送消息的功能接口

3. 应用层通过调用功能模块层提供的接口实现监测点的所有功能



### 监测节点操作系统的选择及移植

在基于单片机的编程中，相对于裸机编程，使用嵌入式操作系统可以提供丰富的软件接口，并提供多任务的并行能力，可以有效提高软件的可拓展性。

在监测节点的软件设计中，本课题使用UCOS2嵌入式操作系统，他是一款可裁剪的基于优先级调度的实时操作系统。

为了能够在STM32上使用该操作系统，需要对UCOSII源码的其中三个文件进行修改

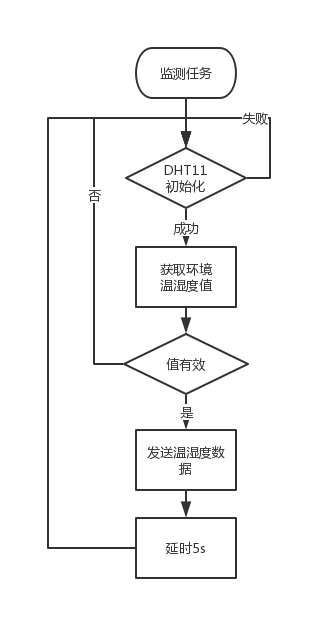
1. os\_cpu.h： 进行数据类型的定义，以及处理器相关代码和几个函数原型；

2. os\_cpu\_a.asm：是移植过程中需要汇编完成的一些函数，主要为任务切换函数

3. os\_cpu.c：定义一些用户 HOOK 函数

### 监测节点程序流程图

监测节点的主要任务就是监测节点所在位置环境温湿度值并传输给汇聚节点。其整个流程作为一个任务使用操作系统进行调度。



其主要流程如下：

1. 对DHT11进行初始化。在上电后，首先检测DHT11模块是否存在，若存在，则对该模块进行一次复位。

2. 获取环境温湿度值。调用DHT11模块的接口获取环境温湿度值。

3. 检测读取的值是否为有效值。若出现异常值，如同上次值比较出现较大偏差，则舍弃该值

4. 发送数据。调用自定义的通讯协议接口，将读取到的温湿度值放入结构体后传入发送函数进行数据发送

5. 延时。由于DHT11模块对环境温湿度值的采样需要一定时间，故此做延时处理，以等待DHT11模块准备好进行下次检测

其中

# 汇聚节点的设计

## 汇聚节点功能需求

汇聚节点主要功能为收集所辖范围内所有监测节点的环境信息,做出适当整合后提供接口供上层设备查询,并且,为了便于工作人员查看,还应提供图形界面来实时显示数据中心机房内各个监测节点的温湿度值。

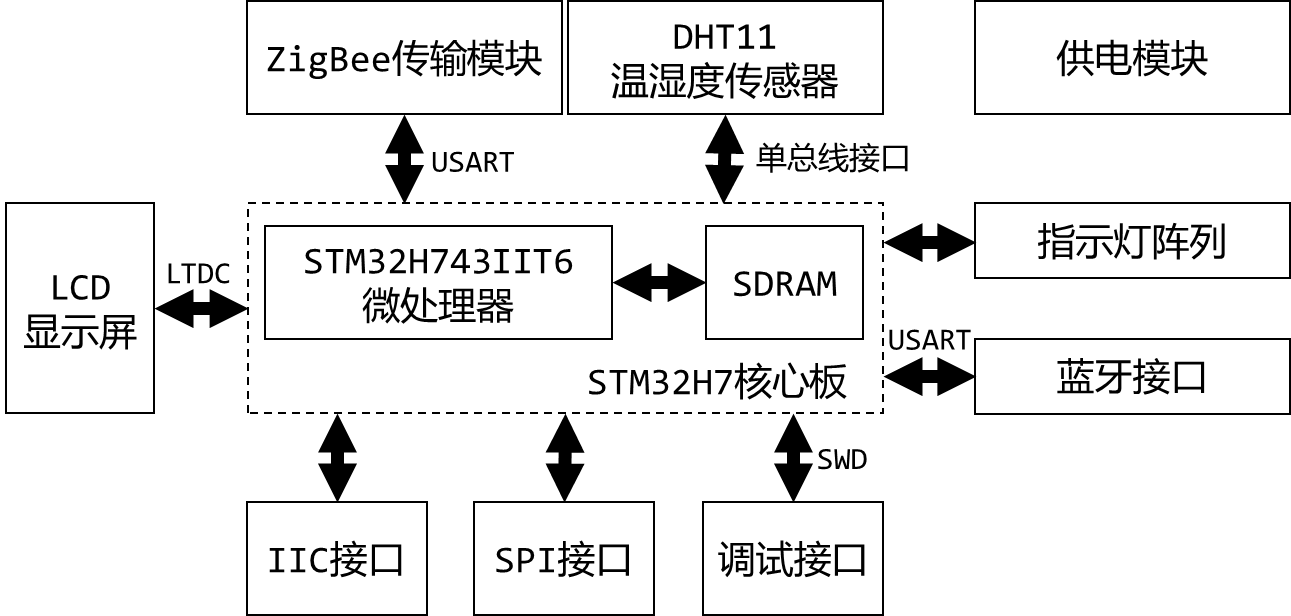
## 汇聚节点硬件设计

### 汇聚节点硬件设计总体方案

汇聚节点的硬件设计主要由以下几个部分组成：在供电模块提供的3.3V基础上，以STM32核心板模块为核心，辅助以USART连接的ZigBee传输模块和蓝牙透传模块，以单总线接口连接的DHT11温湿度传感器，以LTDC接口连接的显示屏等。

由于汇聚节点需要同时运行多个任务，故本课题设计了可独立控制的5个LED灯，用以直观的展示各种任务的运行情况。

此外，考虑到汇聚点的可拓展性，在设计时，为汇聚节点预留出了多种通讯协议硬件接口，包括IIC接口，SPI接口，USART接口，SWD调试接口等。



### 汇聚节点硬件的选型

1. 核心板：为了实现快速的数据处理，屏幕显示等功能，本课题选用了ST公司的STM32H743IIT6型处理器，该处理器使用ARM Cortex M7 内核，采用 6 级流水线，主频可达到400MHz，是监测节点微处理器性能的5.5倍。此外，该核心板上还集成了一片32MB的SDRAM和LCD接口，配合微处理器自带的LTDC功能，可以方便的驱动LCD屏幕，为开发带来了很大的方便。

2. LCD显示屏：汇聚节点的LCD显示屏直接采用了与核心板配套的4.3寸，分辨率为480\*272的液晶显示模块

3. 蓝牙透传模块：选用了ALIENTEK公司的ATK-HC05-V11型主从一体蓝牙模块，连接电脑或手机后，可以方便的实现基于串口的透传功能

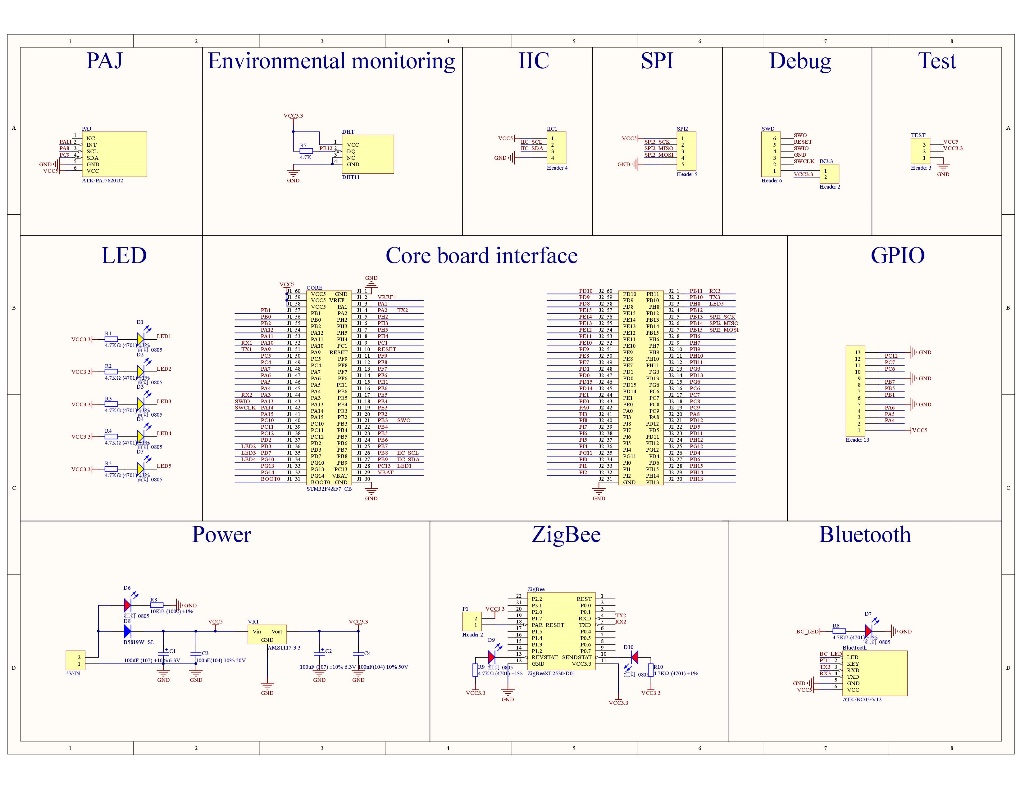
4. ZigBee传输模块。汇聚节点同传输节点一致，使用基于ZigBee的传输模块与监测节点进行通讯。

5. 温湿度传感器模块。汇聚节点也搭配了与监测节点相同的DHT11温湿度检测模块，使汇聚节点也能提供同监测节点一样的功能。

6. 供电模块：由于汇聚节点存在较多模块，所以本课题选用了最大供电电流达800mA的LM1117-3.3型稳压芯片，经测试，该芯片为汇聚节点提供稳定的电源供应

### 汇聚节点电路的设计

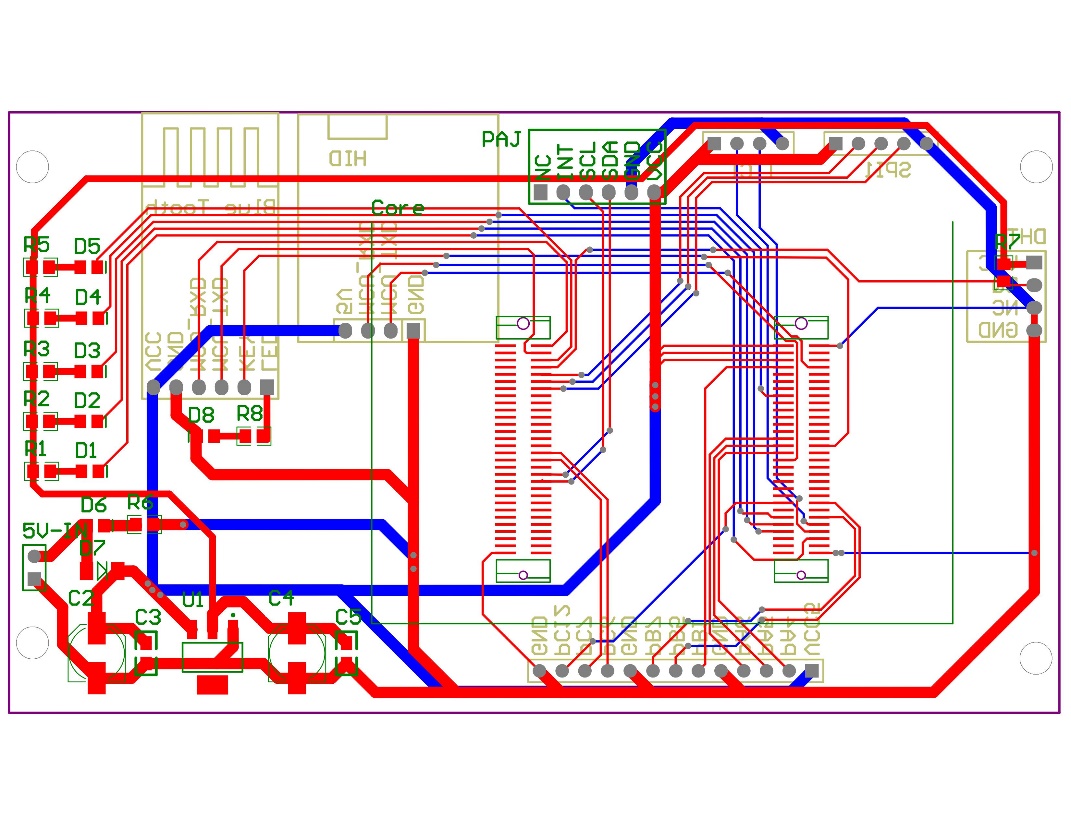
汇聚节点的电路设计较为简单，除稳压电源芯片及ZigBee模块需要一些外围电路，其他接口均直接从核心板引出。

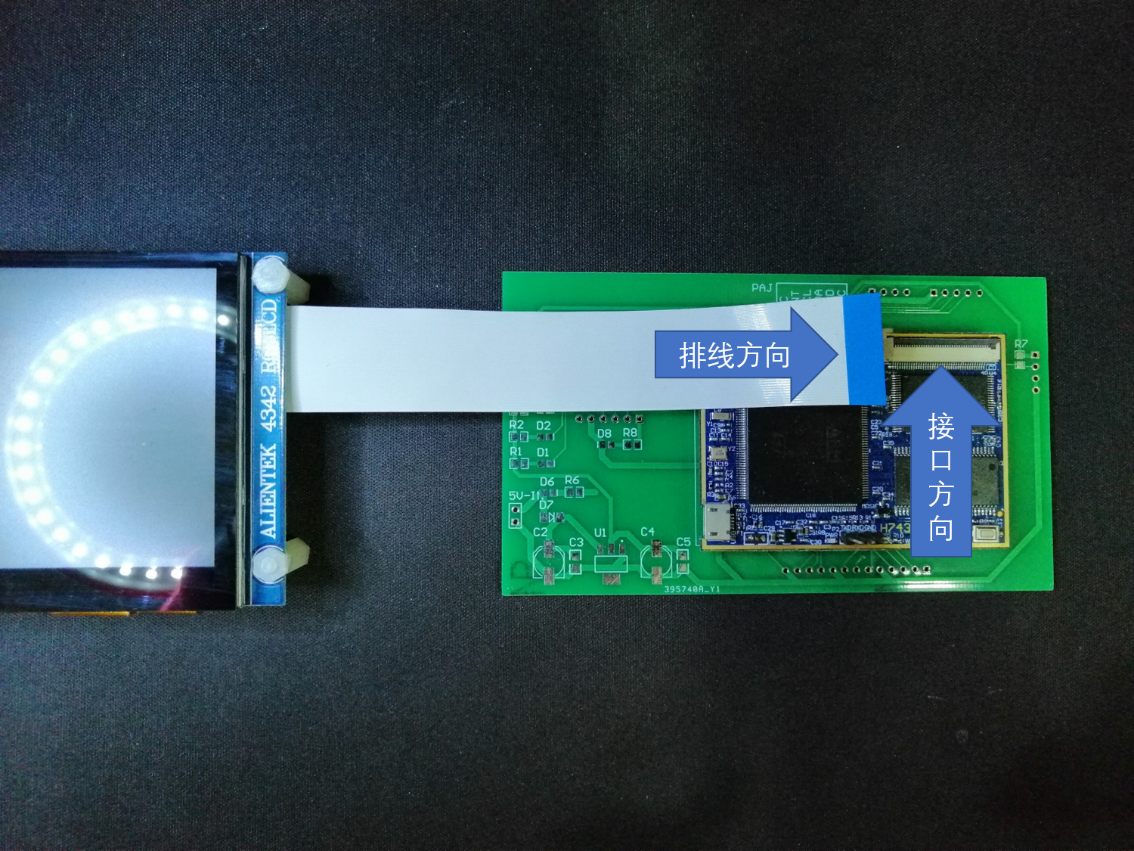


### 汇聚节点PCB设计方案

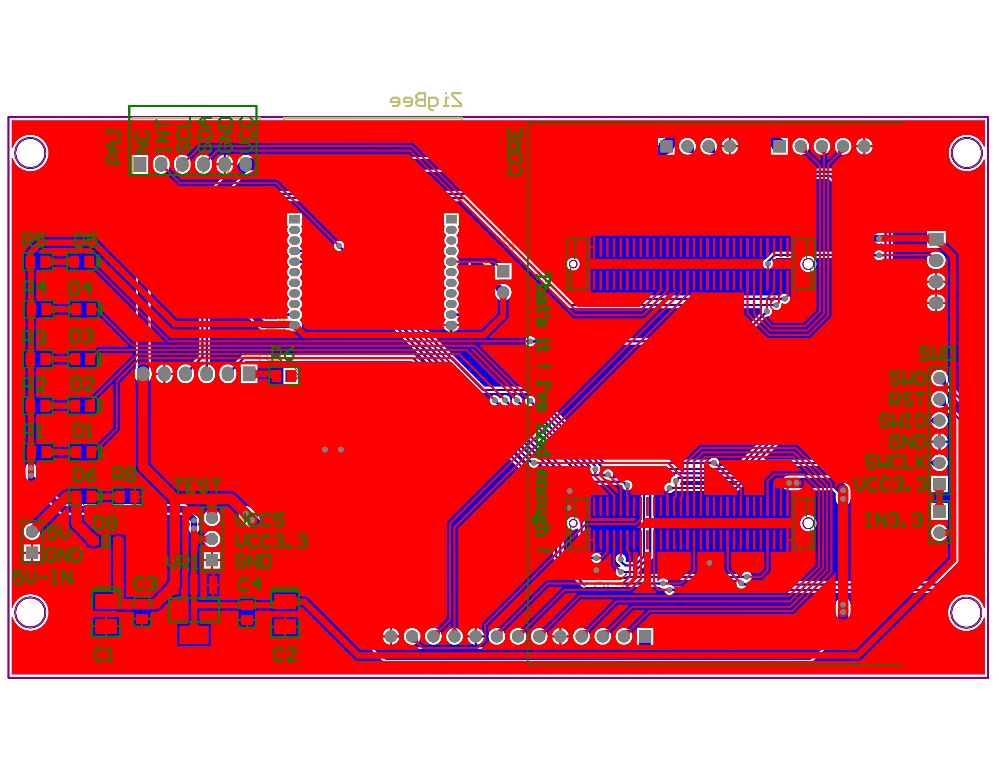
与监测节点不同，汇聚节点对大小并不敏感，然而，为了能够搭配屏幕，在设计中，需要对汇聚节点PCB进行开孔，使显示屏和PCB能够连接到一起，因此对汇聚节点的尺寸精度具有较高的要求。

在最初的设计中没有考虑到显示屏排线方向问题，导致连接显示屏后无法进行固定。PCB设计图如下：





在后续的设计中修复了该问题，并且对PCB进行了铺铜处理，增强了信号的稳定性







## 汇聚节点软件设计

### 汇聚节点程序架构

与监测节点类似，汇聚节点的程序架构也分为硬件驱动层，功能模块层和应用层三层。

1. 硬件驱动层的主要作用为初始化汇聚点点所需的各种硬件模块并为上层调用提供接口。如实现温湿度传感器模块DHT11和多种片内外设的初始化，提供ZigBee模块，SDRAM，LCD等外部模块的接口等

2. 功能模块层分为三大部分。第一部分为UCOSIII操作系统，为功能模块层的部分模块提供消息队列，延时，信号量等服务，第二部分为汇聚点通讯协议栈，图形显示模块，采集点数据表储存模块，他们使用了UCOSIII操作系统提供的部分接口，并向应用层提供对应的服务，第三部分为FreeModbus协议栈，该部分为汇集点提供了Modbus接口的功能。

3. 应用层通过调用功能模块层提供的接口实现监测节点数据接收，LCD屏幕的显示，处理Modbus请求等任务



### 汇聚节点操作系统的选择及移植

由于汇聚节点需要对上层设备提供接口以及接收监测节点的数据,需要较为复杂的调度操作,本课题使用UCOS3作为汇聚节点单片机的操作系统。UCOSiii是于2009年由Micrium提出的嵌入式实时操作系统，相对于ucosii，取消了任务数量，信号量的限制，支持时间片轮转，任务内建消息队列等功能。

移植UCOSIII与移植UCOSII过程相似，在此不再赘述

### 监测节点数据更新部分

检测节点数据接收部分的主要作用为通过调用汇聚点通讯协议栈，将接收到的监测节点的数据储存到监测点数据表中。

该部分由数据接收和中断服务程序任务组成。

数据接收任务的流程如下

1. 数据接收任务开始执行后，初始化任务内建消息队列并打开ZigBee模块的串口中断。

2. 轮询消息队列中是否有消息。

3. 如果消息队列非空，则取出消息，即一帧数据。

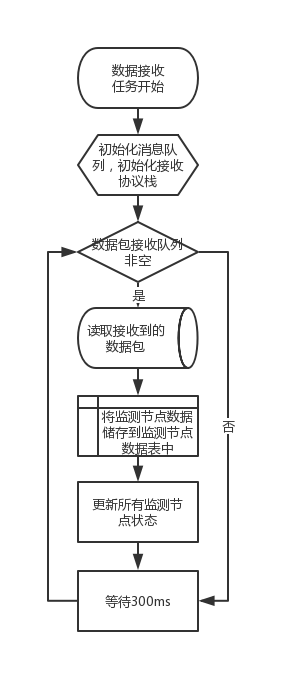
4. 解析数据帧，将接收到监测节点数据放到监测点数据结构体中

5. 将该结构体放置到监测点数据表中，以供其他程序进行查询。

在数据接收任务打开中断后，若接收缓冲区满，则调用串口接收缓冲区满回调函数，该函数运行流程如下

1. 若接收缓冲区慢，则调用一次汇聚点通讯协议栈接口。

2. 若汇聚点通讯协议栈识别到一帧接收完成，则向帧接受队列发送接收到的帧。



### 汇聚节点显示部分

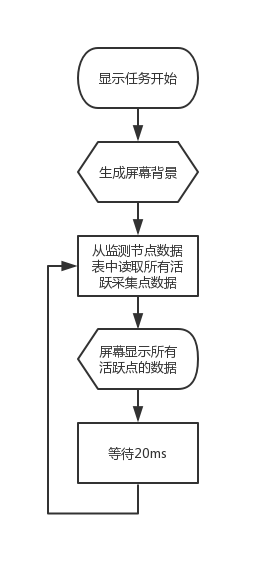
汇聚节点显示部分的主要作用为更新LCD显示屏的内容，由汇聚节点显示任务实现。

汇聚点显示任务的流程如下：

1. 显示任务开始后，调用图形显示模块接口，生成LCD背景

2. 从监测节点数据表中读取所有非关机的监测节点数据

3. 调用图形显示模块接口，将所有非关机的监测节点显示在LCD屏幕上。



### 汇聚节点Modbus部分

汇聚节点Modbus部分的作用主要为处理上位机的Modbus请求，由Modbus处理任务实现。

上位机通过串口向汇聚节点定时发送读取汇聚节点输入寄存器请求，汇聚节点接收到请求后，将最新接收到的监测节点数据放到输入寄存器中，供上位机读取。

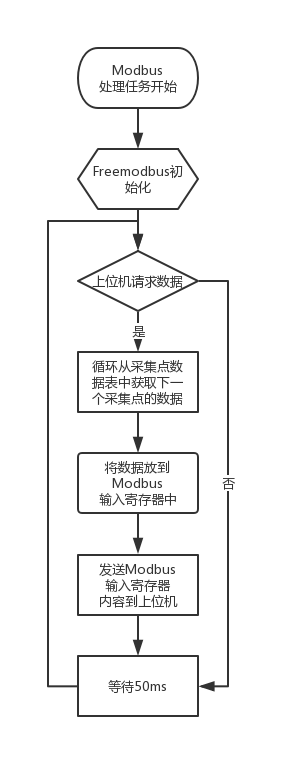
Modbus处理任务的流程如下：

1. 初始化并使能FreeMODBUS模块

2. 如果上位机发送了读取输入寄存器的请求，则调用监测节点数据表模块接口，读取最新接收到的一个监测节点的数据。

3. 将读取到的数据放到汇聚节点的输入寄存器中。

4. FreeMODBUS模块自动将更新后的输入寄存器值发送到上位机。



# 节点间通讯的设计与实现

在检测节点和汇聚节点的硬件设计中，本课题使用了基于ZigBee协议的XL2530-232AP1模块，该模块上电后可以进行自组网，实现了物理层传输的功能，本章主要讨论在物理层的基础上，实现监测节点与汇聚节点的通讯。

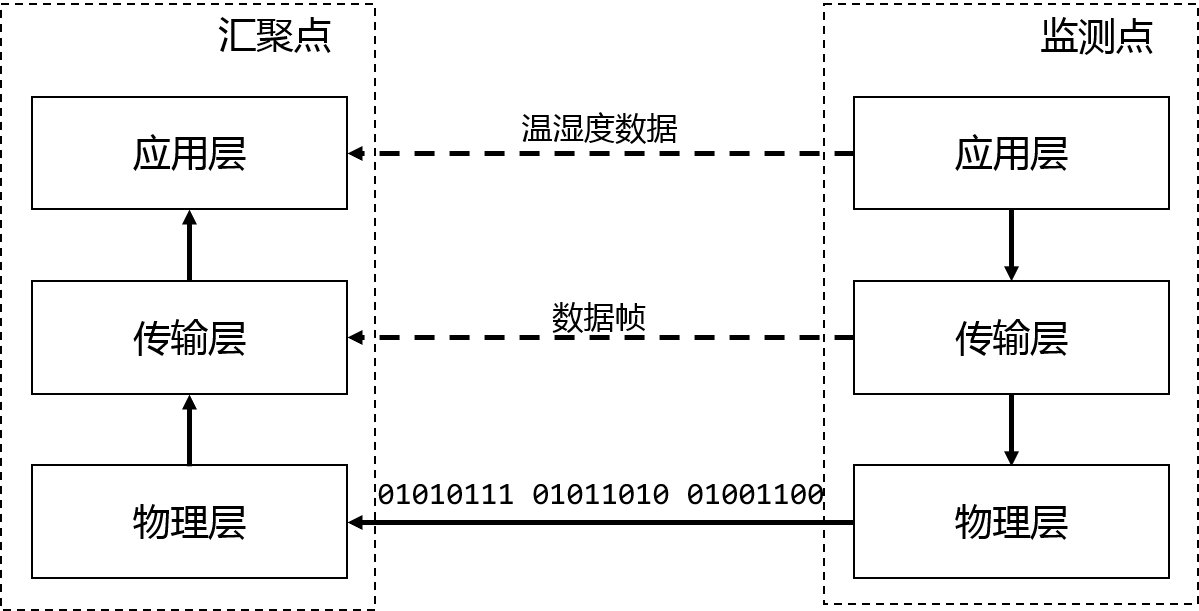
## 通讯协议层次划分

在本课题中，节点间的通讯协议分为三层。

最底层为物理层，该层通过XL2530-232AP1模块，实现了监测节点到汇聚节点比特流的透明传输。

中间层为传输层，该层传输的数据单位为数据帧，其作用为监测节点和汇聚节点提供可靠的端到端服务，向应用层屏蔽了物理层通信的细节，为应用层提供接口，接收应用层传递的不定长数据。

最顶层为应用层，该层的数据包可以直接储存温湿度数据，通过使用传输层提供的服务，实现监测节点向汇聚节点的数据传输。



## 节点间通讯协议的设计

本节主要讨论监测节点与汇聚节点之间的通讯协议的设计。在设计通讯协议的过程中，本课题实现了两种不同的设计思想，分别为一种定常协议方案和一种不定长协议方案。在对这两种协议进行性能对比后，本课题最终使用了不定长协议方案。

### 库函数的串口接收机制简介

为了便于对STM32型微处理器进行编程，ST公司提供了多种函数库，常见的有传统的STD库和较新的HAL库。

在传统的STD库中,其串口接收中断服务函数每接收完一个字节调用一次，需要使用特殊的字节标志或记录是否超时来判断一帧接收是否结束，实现较为复杂。

HAL库是ST公司于2014年推出的一款新的库函数，该函数库提供了一整套一致的中间件组件，可以更好的确保跨STM32产品的最大可移植性。在该库的串口部分中，可以动态的设置串口接收缓冲区长度，当缓冲区满时自动调用缓冲区满回调函数。因此，HAL库原生支持多个字节的接收，若使用一定策略，就可以实现每次调用缓冲区满回调函数都能接收1个数据帧，极大的简化了代码的复杂程度。

### 基于定长协议的方案

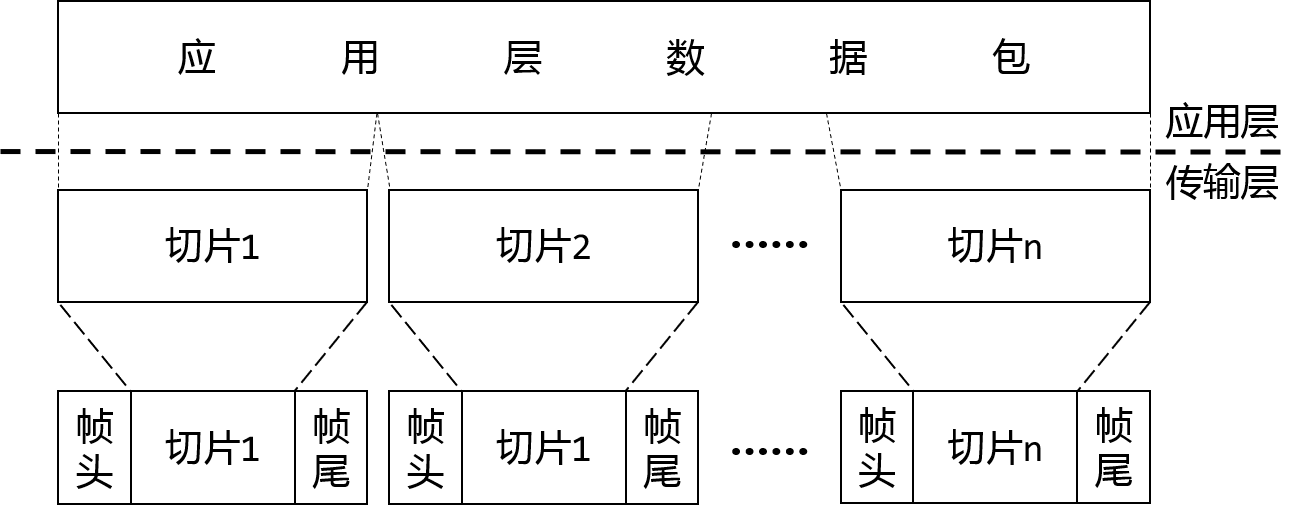
为使用HAL库的串口接收缓冲区满回调机制，该方案规定每帧的长度为定值，并将该值作为接收缓冲区长度，故在理想情况下，汇聚节点每执行一次缓冲区满回调函数，则可以在缓冲区中读取一帧数据。

#### 数据包切片

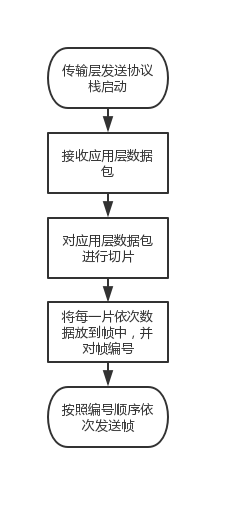
由于传输层并不能规定应用层数据包的长度，为了能够使用固定的帧长承载未知长度的应用层数据，传输层必须对传入的应用层数据包进行切片处理。

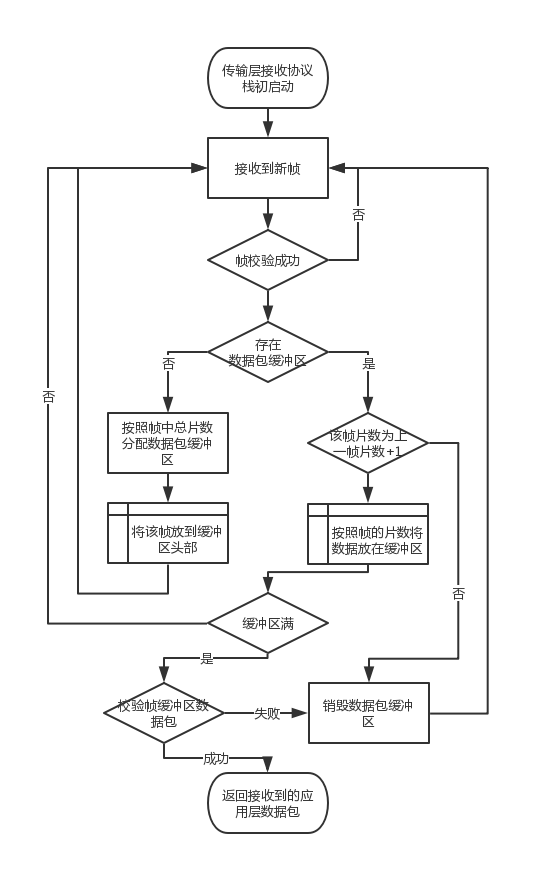
若应用层数据包长度为AByte，传输层每帧承载上层数据空间为tByte，则切片数量为：

N = ⌈A/t⌉片



#### 发送及接收协议栈





#### 帧实现

对于发送数据的节点，每一帧由帧前缀和帧主体组成。

帧前缀为2个字节，为XL2530-232AP1模块的控制命令。第1个字节为0x9E，该值规定了发送模式为点对点发送，第2个字节为0x12，代表本次向该模块传输的字节流流长度为12字节（不包含前2个字节），这两个字节仅在监测节点发送时需要，模块并不向接收节点发送这两个字节。

帧主体为12字节。其中每字节功能如下：

1. 第1个和第2个字节规定了该帧目标的短地址，即汇聚点的短地址。

2. 第3个到第8字节共6字节用于承载上层的数据。

3. 第9个字节代表应用层数据包切片后的总片数。

4. 第10个字节代表该帧在多个切片中的序列号

5. 第11个到第12个字节为对从帧主体第3个字节到第10个字节进行CRC16校验后的结果。

对于接收数据的节点，由于并不会接收到帧前缀，所以其传输层协议栈中每帧长度为12字节，其字节含义与发送数据节点帧主体相同。



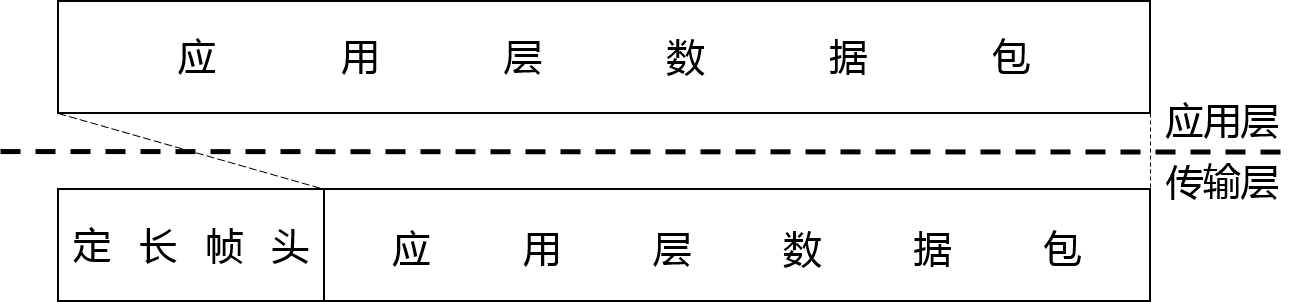
### 基于不定长协议的方案

本节介绍基于HAL库接收缓冲区满回调机制的另一种方案，相比于固定缓冲区长度，该方案通过动态改变缓冲区长度，更加灵活的使用了该机制。

#### 方案简介

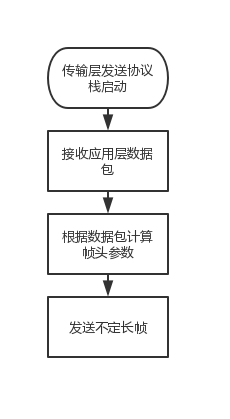
由于HAL库中接收缓冲区长度可以动态改变，因此，如果接收到的一帧内容的前半部分中包含该帧后半部分的长度，就可以在接收到前半部分内容后动态的改变接收缓冲区长度，从而实现不定长帧的接收。

在本方案中，每一帧包括帧头和帧数据部两部分，其中帧头定长，帧数据部变长。帧数据部为不定长的应用层数据包，对其不做任何处理。

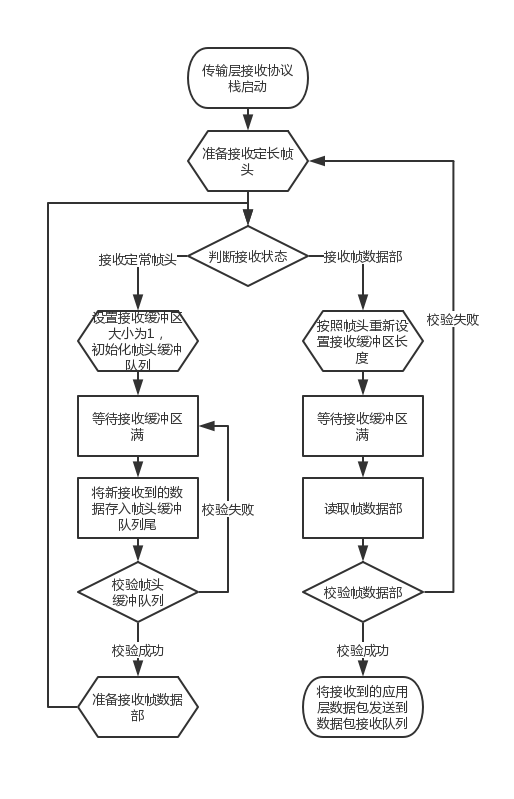


#### 发送及接收协议栈

在发送不定长帧时，首先计算帧头参数，之后将应用层数据全部作为帧数据部，合并成一帧直接发送。



在接收一个不定长帧时,分两步进行。首先，接收协议栈在接收到帧头后提取其中的帧数据部长度，之后重新设定接收缓冲区长度为帧数据部长度，从而一次性接收整个帧数据部。其具体过程如下：



1. 接收协议栈启动后，首先进入准备接收定长帧头模式。

2. 在接收定常帧头模式下，需要配置接收缓冲区长度为1，即配置为每接收1Byte就执行一次回调函数，此外，还需初始化一个与帧头长度相等的帧头缓冲队列

3. 每接收1Byte，就将该Byte放到队尾，如果队列满，则删除队头值再将该Byte放到队尾

4. 校验整个帧头缓冲队列，若校验成功，则认为接收到了完整的帧头，进入接收帧数据部模式，若校验失败，则继续接收。

5. 进入帧数据部接收模式后，读取帧头中帧数据部长度，重新设定接收缓冲区为该长度。

6. 若接收缓冲区满，则认为帧数据部接收完成，校验帧数据部，若校验成功，则将帧数据部发送至数据包接收队列，若校验失败，则舍弃该帧，开始接收新帧。

#### 帧实现

对于发送数据的节点，每一帧由帧前缀、帧头和帧数据部组成。

帧前缀为2个字节，为XL2530-232AP1模块的控制命令。第1个字节为0x9E，该值规定了发送模式为点对点发送，第2个字节为本次向该模块传输的字节流流长度（不包含前2个字节），该值在接收到应用层数据包后计算得来。这两个字节仅在监测节点发送时需要，模块并不向接收节点发送这两个字节。

帧头为6字节，其中每字节功能如下：

1. 第1个和第2个字节规定了该帧目标的短地址，即汇聚点的短地址。

2. 第3个和第4个字节为对应用层数据包进行CRC16校验后的结果

3. 第5个字节为承载的应用层数据包的长度

4. 第6个字节为帧头部分所有字节的和校验结果，高位截断

帧数据部长度不定，为接收到的应用层数据包。

对于接收数据的节点，并不会接收到帧前缀，所以其传输层协议栈中仅有帧头和帧数据部两部分，其字节含义与发送数据节点相同。



### 传输层通讯协议的选择

虽然前两节提到了两种方案都可以实现传输层的通讯，但是在测试中也发现了种种问题。

对于定长帧协议，如果在一帧发送未完成时打开接收协议栈，就会造成一帧发送完毕但是接收缓冲区未满的情况，一旦产生这种情况，则继续接收数据导致缓冲区满时，其中的内容并不是一整个帧，而是前一帧的后半部分和后一帧的前半部分，因此每次均会校验失败，导致无法正常接收。

此外，如果接收到的连续两个帧并不是来自同一个应用层数据包，或者丢失了个别帧均会导致整个应用层数据包接收失败。

综上，该方案理论上能够实现数据传输，但是传输错误率较高。

对于不定长帧协议，由于在接收定常帧头时采用了逐字节接收后放到帧头缓冲队列的方案，所以避免了类似定长帧的帧截断问题，并且由于每帧都能包含一整个数据包，所以也避免了定常协议第二个问题。

然而，由于不定长帧协议并不对应用层数据包进行处理

测试结果表明,该协议能够较好的完成通讯需求,不仅利用了HAL缓冲区机制,而且帧中的信息密度较高,减少了带宽的浪费,。在使用不定长协议后,就可以对应用层数据不做任何操作整个装入帧。

结构体伪代码

在实践中发现,进行切片后,物理层需要以较高的频率发送长度较短的帧, 由于物理层处理能力的限制, 主机接收到的相邻两个帧很可能不是出自同一个包的现象(加接收到不同节点的头帧图),产生这种情况理论上可以增加包缓冲区,但是实现过于复杂,或者增大帧的长度,但是对造成带宽的浪费, 故考虑后放弃了此种通讯方式。

### 应用层的设计

在该传输层的基础上，本课题也设计了对应的应用层协议，协议中每个字节功能如下：

1. 第1个到第4个字节共4字节单精度浮点型的温度数据.

2. 第5个到第8个字节为4字节单精度浮点型的湿度数据

3. 第9个到第10个字节共2字节储存short型的监测点编号

4. 第11字节为保留的控制字

5. 第12字节储存前11字节的和校验值，高位截断。



# 汇聚节点与上位机通讯的设计与实现

## 通讯协议的确定

本设计使用modbus作为汇聚点和上位机进行通讯的协议

## 实现Modbus通信

### FreeModbus的移植

# 上位机的设计

## Niagara框架简介

## 数据接收机制的设计

## 人机交互界面的设计

# 调试与验证

# 总结与展望