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，设计并实现了一套数据中心环境监测系统。该系统能够实时监控机房空间内各个点的温湿度情况，其收集到的数据不仅用于空调系统的控制，从而有效的降低数据中心的能耗，而且可以用于分析机房内空气的实际流动情况，以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。

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。

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. 设计了一款运行于电脑的上位机软件，实现了对机房内温湿度的实时监控以及数据可视化。

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。

关键字：数据中心；环境感知；监测系统

# 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，企业对数据中心服务的需求也日渐增长。

从国际上看，虽然由于大型化可显著的降低成本，全球数据中心数量在近几年成下降趋势，预计到2020年将减少至 42.2 万个，然而机架的总数量近几年一直平稳上升，同时单个机架功率提升快速，预计到2020年总机架数量将达到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%[1]。

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，由于我国数据中心的建设起步较晚，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，在很多数据中心中还存在仅仅由人员巡检来确定服务器运行状态的情况。一般来说，国内数据中心每个运维人员平均能够同时管理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运维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，在2019年，我国工业和信息化部，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[2]，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低到国际先进水平，并实现运维水平的大幅提升。

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，从而有效的降低新建数据中心的能耗，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[3]，亚马逊旗下数据中心的PUE值平均值为1.2[4]，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[5]。

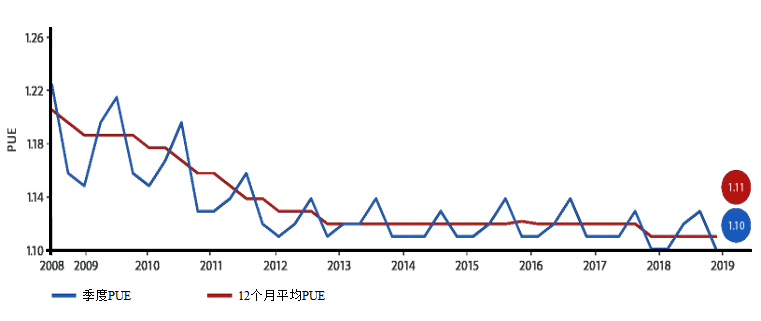


图 1.1Google公司近几年PUE值变化

在对数据中心房间内环境监控方面，Rodriguez等使用Arduino单片机套件设计了一种温湿度监控网络[6]，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。

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[7]，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。然而，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，也无法进行直接应用，且所有数据是先收集再建模，无法达到实时性监控的要求。

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[8]。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监测系统。

## 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。该系统分为分布式监测节点、汇聚节点和上位机三个部分。

对于检测节点和汇聚节点，本课题使用软件Altium Designer软件设计了其PCB，进行焊接及硬件调试后，通过Keil软件使用C/C++语言对其编程，实现了房间内温湿度的监测。对于上位机，本课题使用Python语言设计了一款软件，使其通过Modbus协议读取汇聚节点的数据，实现了人机交互界面和数据进行可视化。

此外，本课题还设计了两种汇聚节点与监测节点间的3层通讯协议，用于实现监测节点与汇聚节点之间的数据传输，并对他们的性能进行了比较。

## 论文结构安排

本文共9章，具体内容安排如下：

第1章首先分析国内外数据中心发展的趋势，指出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。之后分析目前已有的对数据中心环境监测工作，指出他们的优势和缺陷。最后总结本课题核心的研究工作以及论文的结构安排。

第2章阐明课题的总体设计方案，指出课题中每个部分设备的作用以及系统架构。

第3章阐明课题中监测节点的设计和制作过程。首先分析监测节点的功能需求，之后介绍监测节点的硬件和软件的实现过程。

第4章阐明课题中监测节点的设计和制作过程。首先分析汇聚节点的功能需求，之后介绍监测节点的硬件和软件的实现过程。

4.4.5.1阐明本课题为实现节点间通讯而设计的2种通讯协议。文章首先对节点间的通讯层次进行划分，之后阐述通讯协议的设计思路及实现过程，最后对这两种通讯协议的性能做了比较。

**Error! Reference source not found.**阐明汇聚节点与上位机通讯的实现方案。文章首先分析使用Modbus通讯协议和JSON语言的优点，之后简述汇聚节点与上位机实现了数据通讯的过程。

第6章叙述上位机的实现过程。首先对上位机使用的框架进行简要说明，之后介绍上位机对汇聚节点数据的接收，超限报警，数据可视化等功能的实现，

第7章对本课题的成果进行了测试和验证。实地测试的结果表明，本课题提出的方案能够有效的达成最初的设计目标，具有较强的应用价值。

第8章是对本课题所做工作的总结，和对本课题进行进一步深入研究进行的展望。

# 总体设计方案

对数据中心内温湿度监测系统进行需求分析可知，为实现房间内不同空间温湿度节点的实时监测，首先需要有一种无线微型设备分散在房间各个位置来采集温度，还应有设备对采集到的数据进行分析整理并上传至上位机，最后上位机应能够对数据进行分析整理和可视化。

根据以上需求，本课题提出了如图 2.1的系统架构：

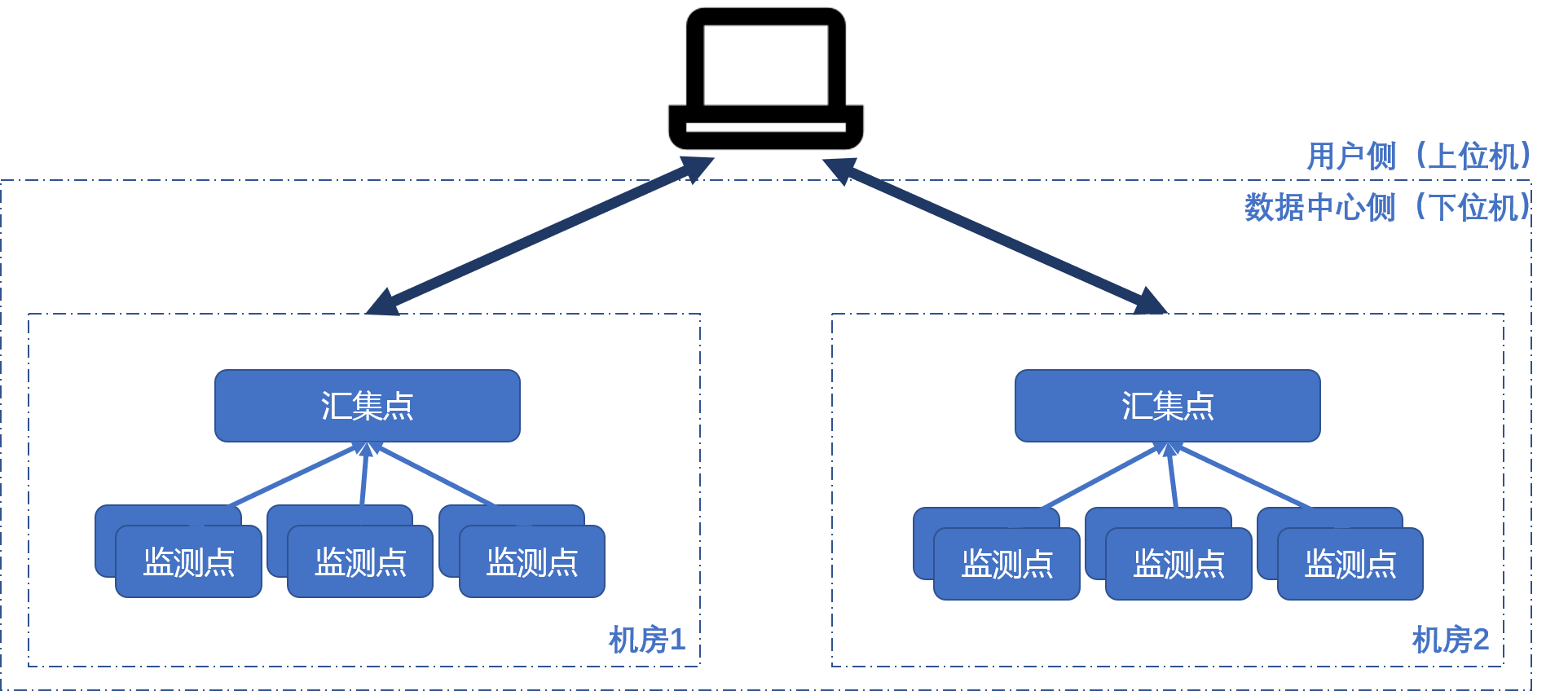


图 2.1本课题系统架构

由图 2.1可知，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。

下位机部分中存在两类设备，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接口供上位机设备查询。

运维人员可以通过上位机的可视化界面实现房间内温湿度状态的实时监测以及数据统计等功能[9]。

# 监测节点的设计

本章首先依据监测节点的功能需求提出了设计方案，按照方案进行硬件选型后，设计了PCB并完成了焊接及调硬件试，最后基于单片机完成了监测节点的软件设计。

## 监测节点的功能需求

在本课题的总体架构中，监测节点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。同时，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. 监测节点应有足够低的功耗[10]，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以无线传输的方式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，以便能够在房间内唯一的标识监测节点，并方便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

监测节点的硬件部分主要由以下5个模块组成：

1. 供电模块为整个监测节点提供稳定的电源。
2. 无线传输模块提供监测节点与汇聚节点进行数据通讯的功能。
3. 传感器模块用于采集环境的温湿度数据。
4. 节点号输入模块用于读取用户输入的节点的唯一编号。
5. 微控制器模块连接传输模块、传感器模块，编号输入模块，按照程序与各个模块进行交互，实现所有预定的功能。

最后，从微控制器模块引出调试接口，便于程序的下载与调试。

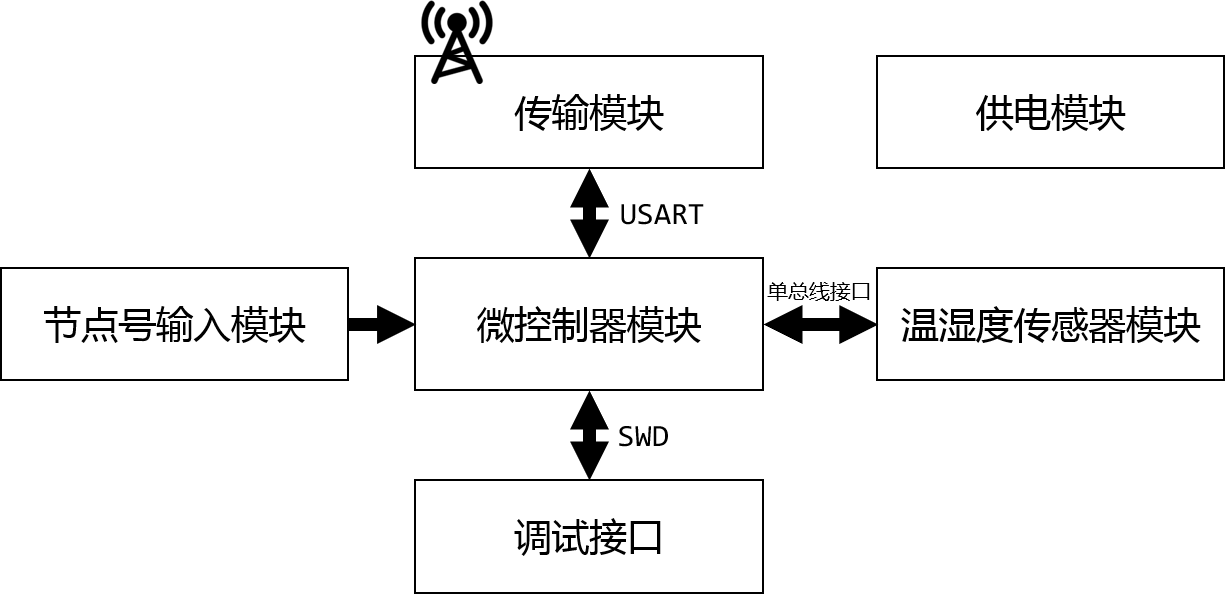


图 3.1监测节点硬件架构图

### 供电模块设计

由于监测节点需要较小的体积，以及不受外部供电的限制，所以采用电池进行供电。表 3.1为几种常用电池的比较：

表 3.1常见电池对比

Table 3.1 Common battery comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 电池型号 | 供电电压 | 电池容量 | 电池体积 | 是否可充电 | 价格 |
| 南孚5号电池 | 1.5v | 2100mA·h | 直径14×49mm | 否 | 2.4元 |
| 神火18650 | 3.7v | 2300mA·h | 直径18×65mm | 是 | 12.8元 |
| 402030 | 3.7v | 230mA·h | 4×20×30mm | 是 | 7.9元 |

考虑到监测节点需要较小的体积，较高的供电电压以及较低的成本，本课题采用了型号为402030的小型锂电池进行供电，能够向监测节点提供电压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，本课题采用了RT9193-3.3超低压差, 超快速线性稳压芯片对锂电池进行从3.7v到3.3v稳压，测试表明，该芯片具有极低的噪声和功耗，能够满足监测节点各个模块的用电需求。此外，如果需要监测节点具有更长的续航时间，可以并联多个电池，由于单个电池较小，3个及以下的电池并不会对体积造成较大的影响。

在设计电路时，还加入了一个LED指示灯用于指示电池供电情况，在实际应用中，可以不焊接LED以节约电量。

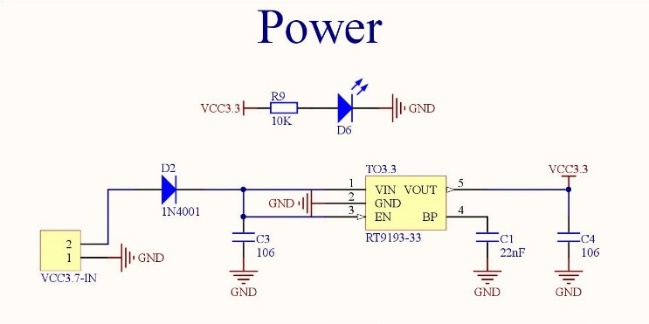


图 3.2监测节点供电模块原理图

### 传输模块设计

ZigBee是一种以IEEE802.15.4为基础的短距离，低功耗，低速率，低成本的无线传输协议[11]，由于其具有的自组网的特点，特别适用于较小空间内的无线传感网的部署。

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。在简单的配置后，可以在物理层上实现点对点传输和广播

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，接收控制字段和数据。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时闪烁。

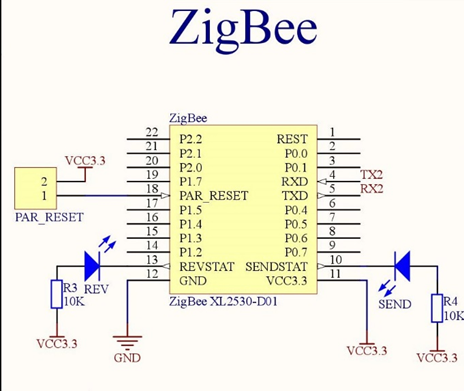


图 3.3传输模块原理图

### 传感器模块设计

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[12]。其温度测量精度为±2℃，湿度测量精度为5%。

传感器部分的其电路设计较为简单，读数据手册可知，在保证3.3v供电的前提下，仅需要在数据DQ引脚上接5KΩ左右的上拉电阻即可。

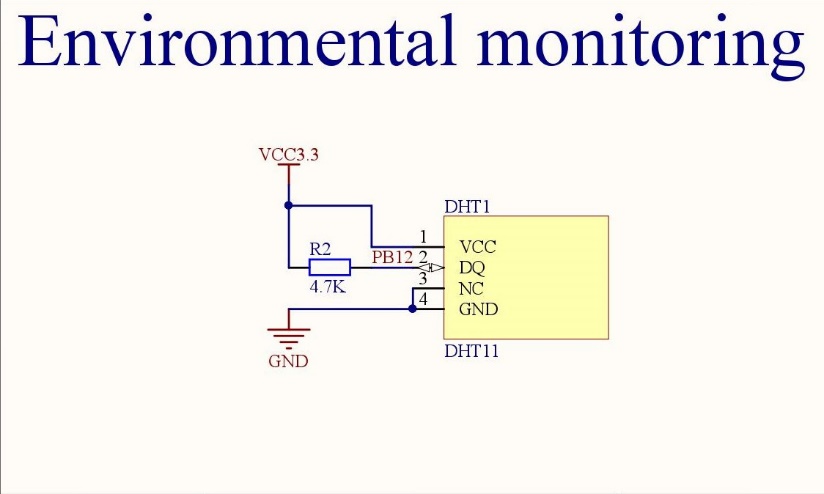


图 3.4传感器模块的原理图

### 微处理器模块设计

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,在低功耗模式下，其电流仅有20uA。

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。

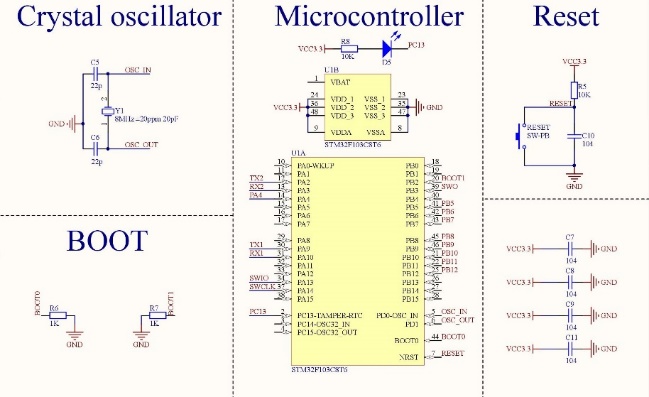


图 3.5微处理器模块原理图

### 编号输入模块的设计

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房间内唯一的编号。

拨码开关是一种内含多个键的微型开关，其中每一个键都可以控制一路电流的通断，常常被用作二进制编码，设定设备的地址等。在本课题中，单片机通过使用多个不同引脚与拨码开关连接，读取引脚的输入值确定该引脚的通断状态，从而确定监测节点的编号。

本课题在每个监测节点上放置2个4位拨码开关，即有8个二进制位可供选择，最多能够标识256个节点，能够满足监测节点的要求。

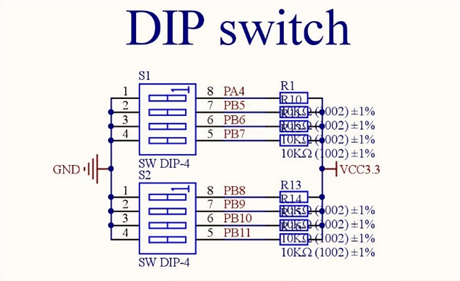


图 3.6编号输入模块拨码开关原理图

### 监测节点PCB设计方案

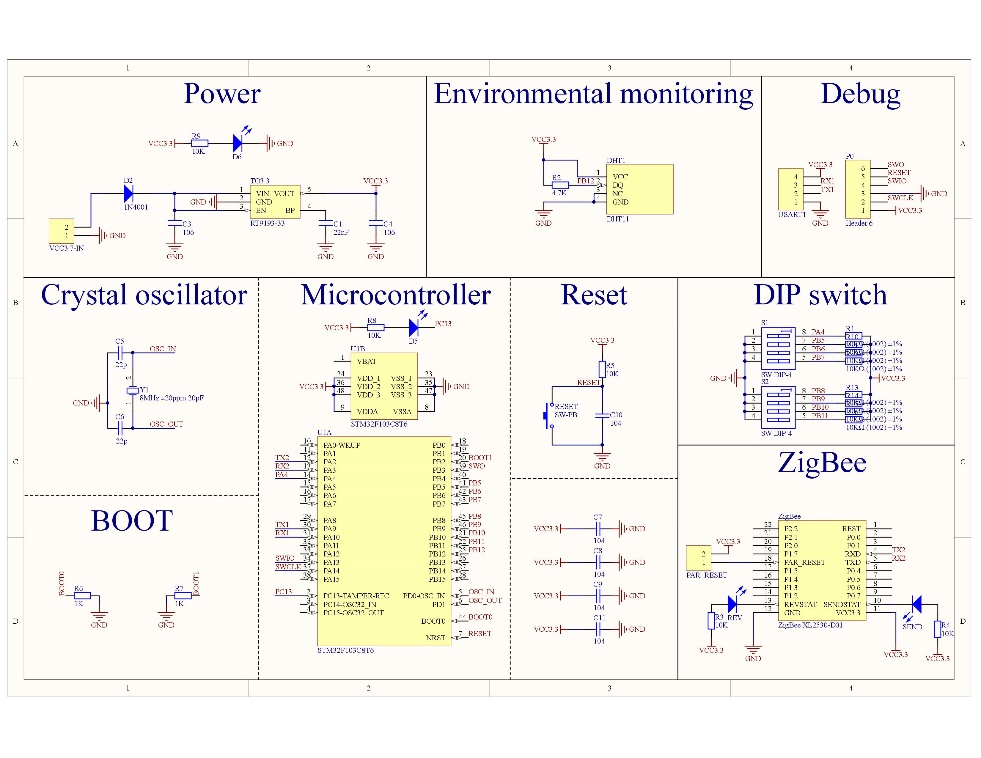


图 3.7监测节点原理图

对于监测节点，其PCB最重要的设计需求就是较小的体积。因此，在元件的选型上，本课题均使用了较小的封装，并采用双面板设计，有效的减小了监测节点PCB的大小。

在监测节点PCB的设计过程中，曾因考虑到DIP开关占用体积较大，故取消了加装该模块，最后设计成品大小为42mm×30mm。

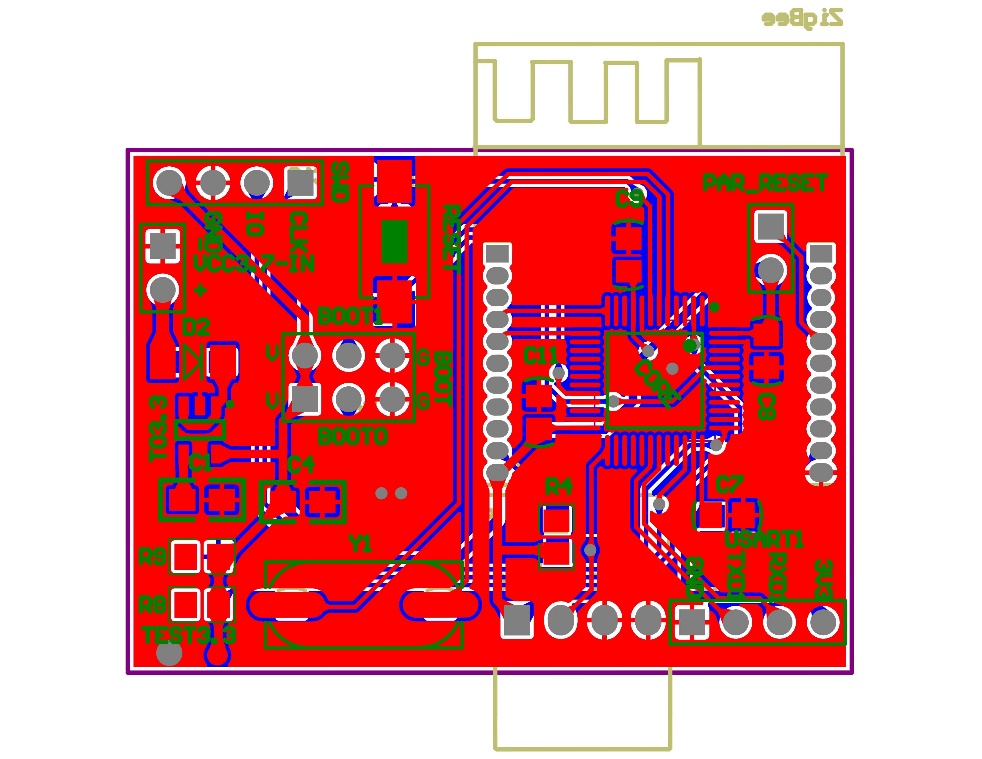


图 3.8未加装拨码开关的PCB

然而在之后的测试中发现，取消拨码开关后，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拨码开关部分的设计。

经重新布线和改良，最终的监测节点PCB大小仅为40mm×35mm。

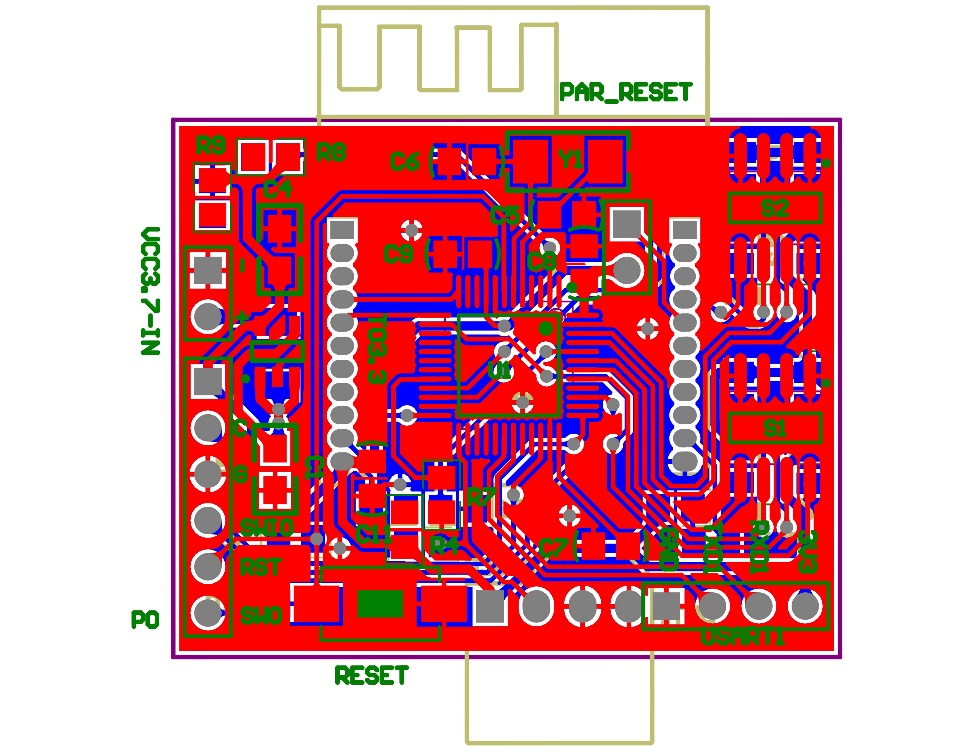


图 3.9监测节点PCB

在采购元件并完成焊接工作后，实物图如下：

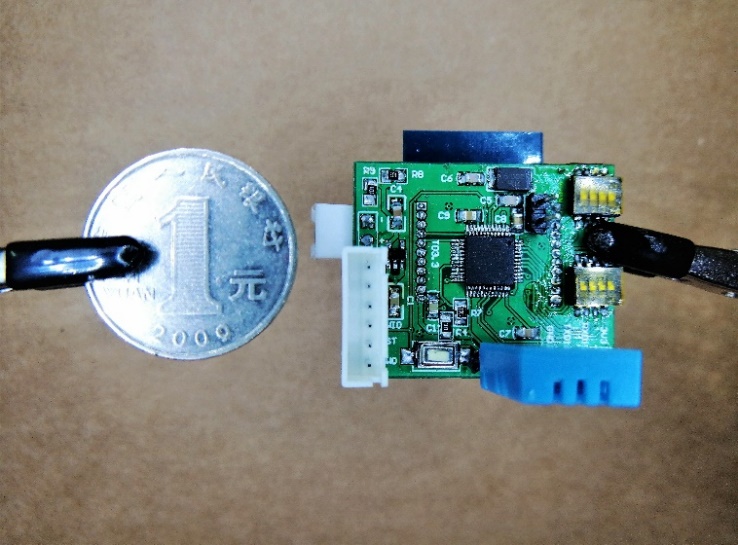


图 3.10监测节点正面实物图

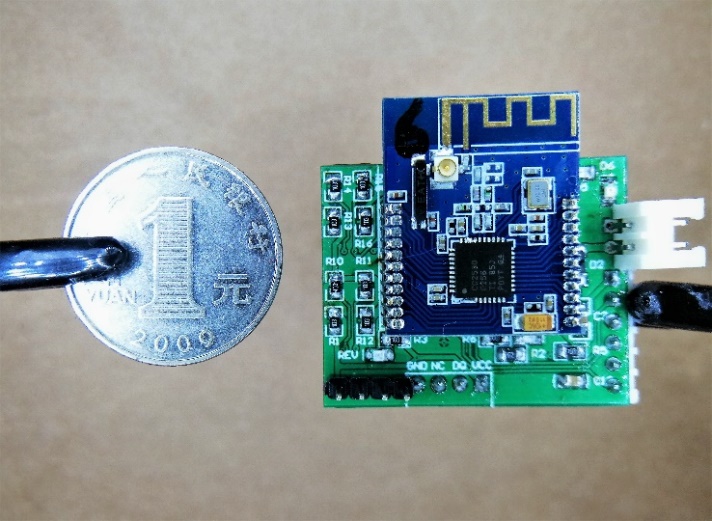


图 3.11监测节点反面实物图

## 监测节点软件设计

本节介绍监测节点上微处理器中运行程序的设计。

### 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[13]。

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，拨码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嵌入式操作系统[14]向应用层提供任务调度，消息队列，延时等功能，并且通过监测节点传输协议栈提供向汇聚节点发送数据功能的接口。
3. 应用层通过调用不同的功能模块层接口实现监测点的所有功能。

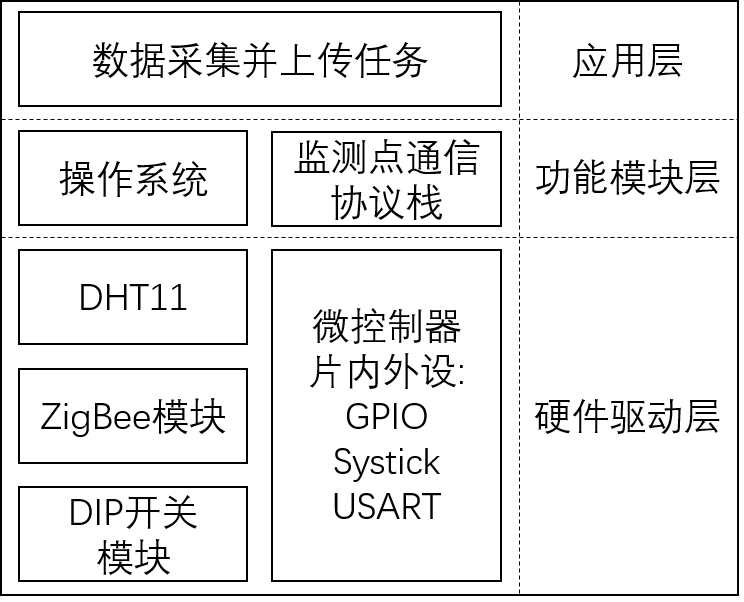


图 3.12监测节点程序架构

### 节点操作系统的移植

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，可以方便的增减功能，有效的提高软件的可拓展性。

本课题选用µC/OS-II作为监测节点的嵌入式操作系统，这是一款可裁剪的基于优先级调度的实时操作系统，提供了非常丰富的软件接口，在嵌入式领域使用非常广泛。

为了能够在STM32微处理器上使用该操作系统，需要对µC/OS-II源码的以下三个文件进行修改。

1. os\_cpu.h：包含数据类型的定义，以及处理器相关代码和几个函数原型。
2. os\_cpu\_a.asm：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。

移植过程较为简单不在此赘述。

### 监测节点程序流程图

监测节点的主要任务就是监测所在位置环境温湿度值并传输给汇聚节点。在设计中，整个流程作为一个任务通过操作系统进行调度。

任务主要流程如下：

(1) 对DHT11传感器进行初始化，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，调用监测节点通讯协议栈接口，将读取到的温湿度值放入传入发送函数进行数据发送。

(5) 进入低功耗模式。待数据发送完成后，监测节点进入低功耗模式，以节约电量。

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

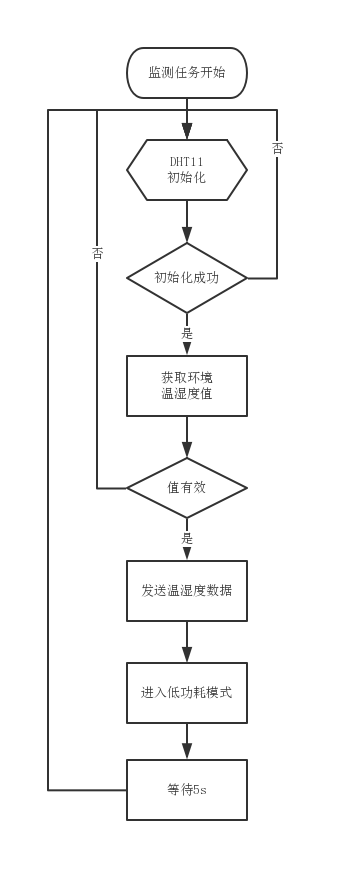


图 3.13监测节点程序流程图

# 汇聚节点的设计

本章首先依据汇聚节点的功能需求提出了设计方案，按照方案进行硬件选型后，设计了PCB并完成了焊接及调试，最后基于单片机完成了汇聚节点的软件设计。

## 汇聚节点功能需求

按照本课题的总体架构，汇聚节点应具有两大功能[15]。

首先，汇聚节点需提供人机交互界面，该界面不仅能够实时显示收集到的各个监测节点的温湿度值，还能应能够配置监测节点编号与其实际空间坐标的映射。

其次，监测节点需要向上位机提供接口，向上位机提供收集到的每个空间位置的环境数据。

## 汇聚节点硬件设计与实现

### 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：

1. 供电模块为整个汇聚节点提供稳定的电源。
2. 无线传输模块提供监测节点与汇聚节点进行数据通讯的功能。
3. 温湿度传感器模块使汇聚节点也具有监测温湿度的功能。
4. LCD触摸显示屏为监测节点提供人机交互界面。
5. 指示灯阵列，即多个可独立控制的LED，由于汇聚节点需要同时运行多个任务，使用多个LED指示灯可以直观的展示各个任务的运行情况。
6. 微处理器核心板，用以连接上述各个模块，按照程序与各个模块进行交互，实现所有预定的功能。
7. 考虑到汇聚节点的可拓展性，应为汇聚节点预留出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。

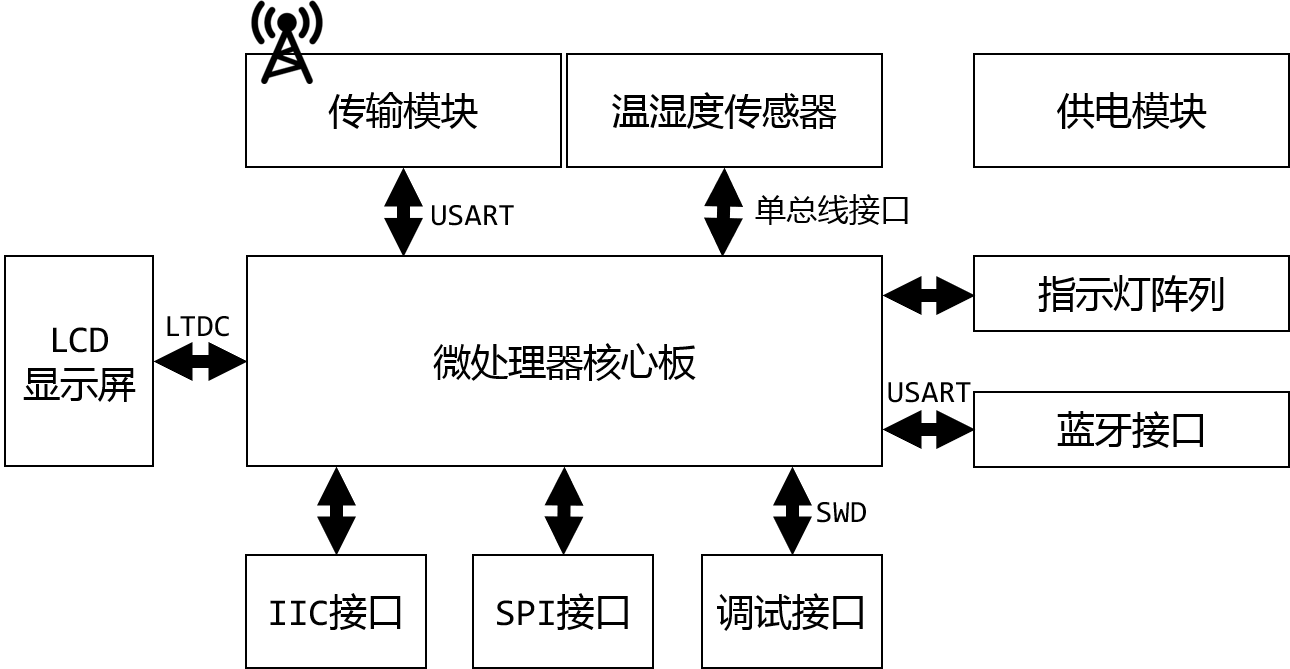


图 4.1汇聚节点硬件架构图

### 汇聚节点硬件选型

相对于监测节点，汇聚节点的硬件选型较为简单。在汇聚节点中，本课题使用了与监测节点相同的传输模块XL2530-232AP1，以及温湿度传感器模块DHT11。

其他模块的选型如下：

1. 微处理器核心板：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触摸显示屏：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：本课题选用了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. 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PCB设计方案

汇聚节点的电路设计中除稳压电源芯片、无线传输模块和传感器模块需要一些外围电路之外，其他接口均直接从核心板引出。

其中，无线传输模块和传感器模块的电路图与监测节点相同，见图 3.3和图 3.4。

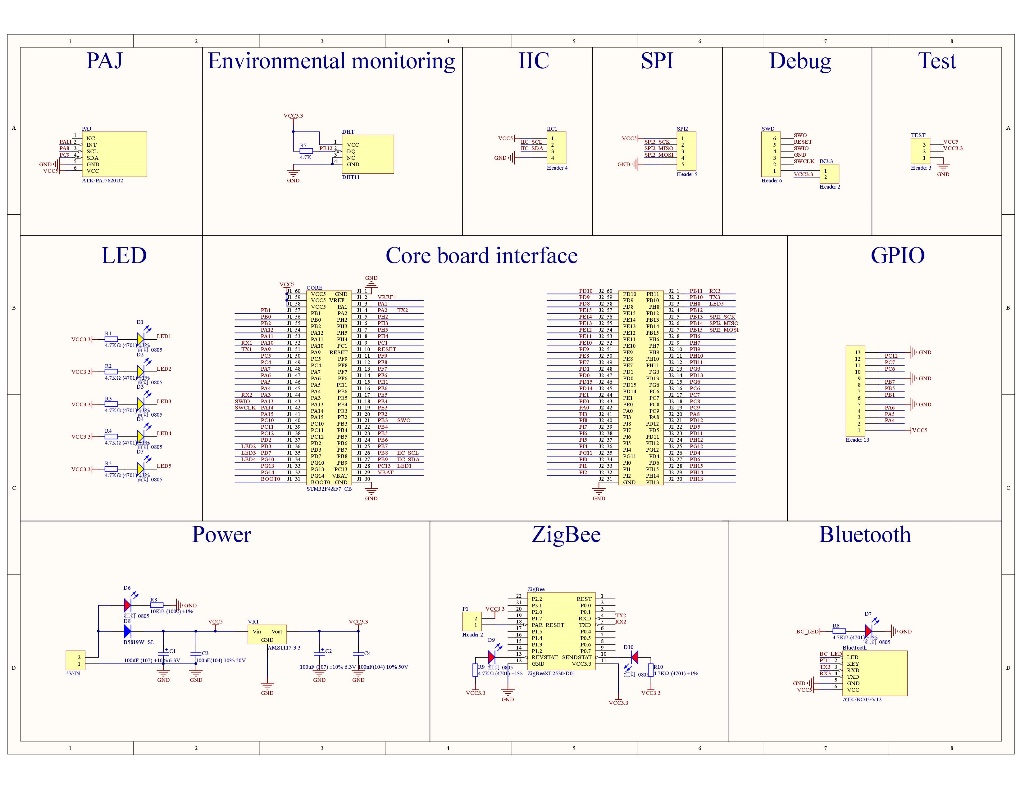


图 4.2汇聚节点原理图

与监测节点不同，汇聚节点并不需要较小的体积。然而，为了能够适配屏幕，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。

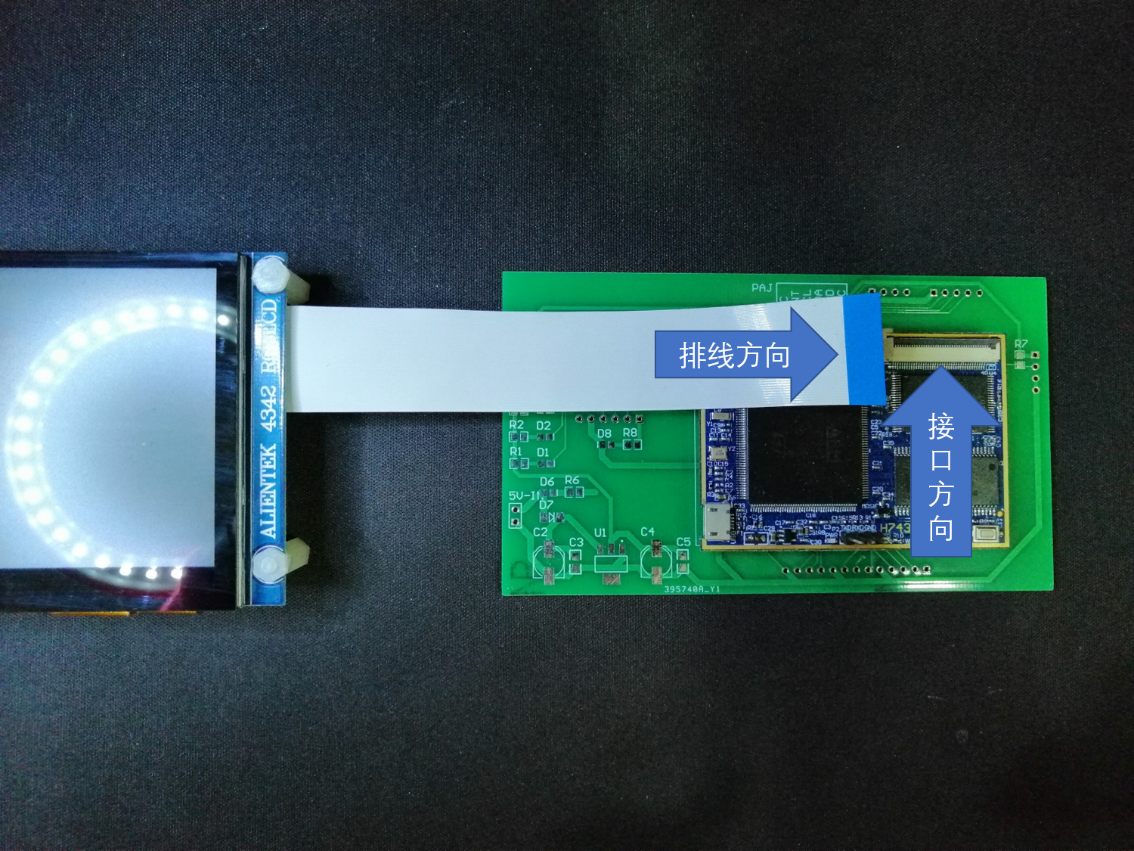


图 4.3排线方向不同导致无法固定的汇聚节点

在后续的设计中修复了该问题，并对PCB进行了铺铜处理，增强了信号的稳定性。

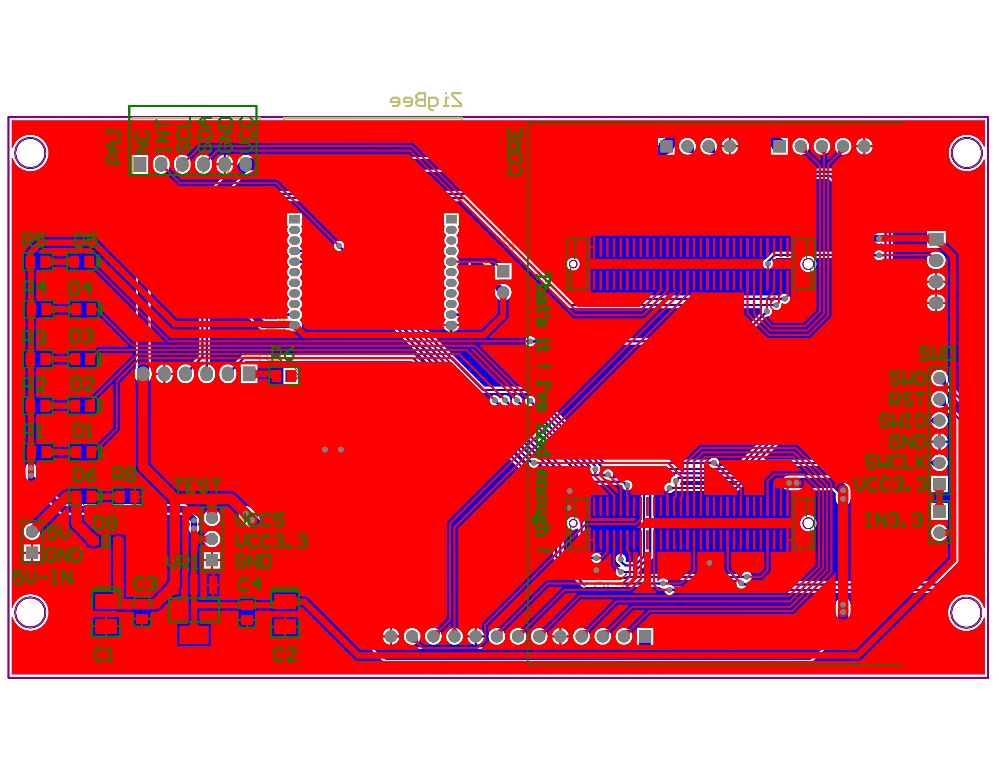


图 4.4汇聚节点PCB



图 4.5汇聚节点实物图



图 4.6汇聚节点拆解

## 与上位机通讯方案的确定

为了使汇聚节点具有更好的兼容性，以便能够和常见的上位机平台进行通讯，本课题采用了Modbus作为和上位机的通讯标准[16]。Modbus串行通讯协议标准自1979年发表以来普及迅速，目前，他不仅是工业电子设备之间常用的连接方式，也是工业领域通信协议的业界标准。

标准的Modbus协议中一个数据的单位为一个比特或一个16位的寄存器，在此基础上，如果按照一定规则操作多个连续的寄存器，就可以实现如长整型，浮点型，字符串等类型的写入或读取。

JSON（JavaScript Object Notation）是一种于1999年提出的轻量级的数据交换语言，该语言以易于让人阅读的文字为基础，用来传输由属性值或者序列性的值组成的数据对象。由于目前大多数编程语言都原生支持对JSON的解析，所以该数据交换语言广泛的应用于面向用户的软件开发领域[17]。

为了使上位机能够更方便的解析收集到的数据，在本课题中，监测节点使用Modbus RTU模式直接向上位机传输JSON格式的字符串。通过使用这种方案，不仅利用了Modbus良好的兼容性，也大大方便了上位机对数据的处理[18]。

具体而言，上位机每次请求数据时，都向监测节点发送读取多个保持寄存器的请求，监测节点接收到请求后，将装填好数据的JSON字符串放置到连续的保持寄存器中，之后调用Modbus接口将寄存器数据返回，从而实现数据的传递。

## 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，提供传输模块，SDRAM，LCD等外部模块的接口等。
2. 功能模块层分为两大部分。第一部分为嵌入式操作系统，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。第二部分为汇聚节点的通讯协议栈、图形显示模块、监测点数据储存模块和Modbus协议栈，这些模块均使用操作系统提供的接口，并向应用层提供对应的服务。
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更新任务，LCD触屏人机交互任务和Modbus处理任务。

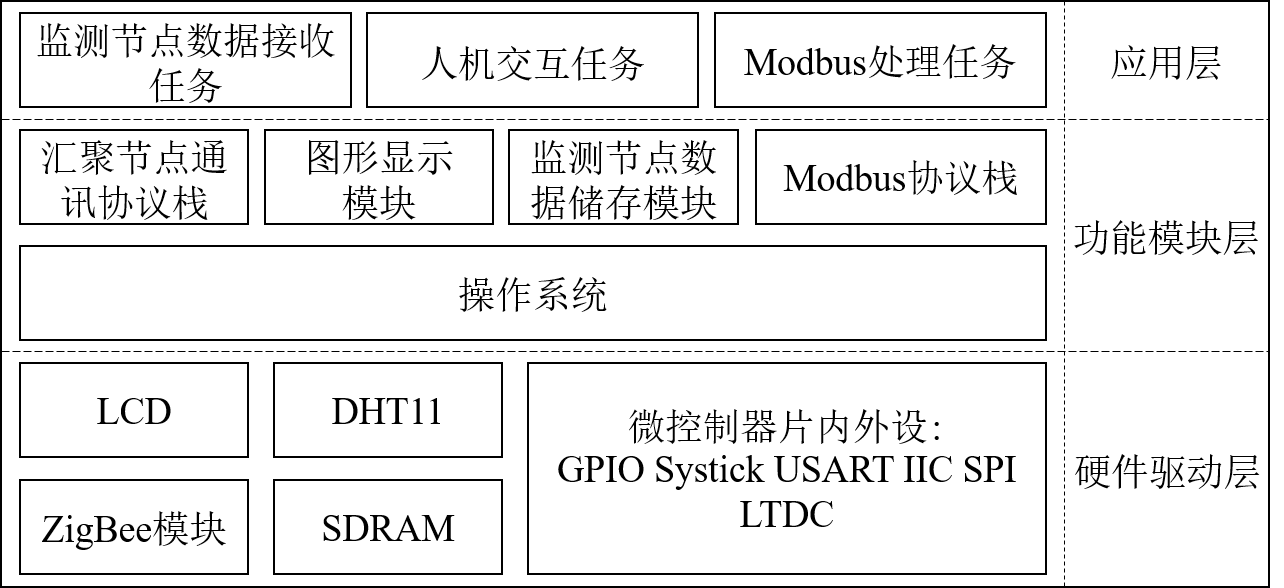


图 4.7汇聚节点程序架构

### 汇聚节点操作系统的选择及移植

由于汇聚节点需要对上位机提供接口并接收监测节点的数据，需要较为复杂的调度操作，本课题选用了µC/OS-III作为汇聚节点微处理器的操作系统。

µC/OS-III是2009年由Micrium推出的嵌入式实时操作系统[19]，相对于在监测节点使用的µC/OS-II，取消了任务数量和信号量的限制，新增了时间片轮转，任务内建消息队列等功能，性能有了较大的提高。

在汇聚节点移植µC/OS-III与在监测节点移植µC/OS-II过程相似，在此不再赘述。

### 监测节点数据更新任务

监测节点数据更新任务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储存模块中。

数据更新任务的流程如下：

1. 数据更新任务开始执行后，首先初始化汇聚节点通讯协议栈，之后将任务内建消息队列挂载到通讯协议栈中，使之可以获得通讯协议栈接收的数据。
2. 轮询任务内建消息队列中是否有消息。
3. 如果消息队列非空，则说明接收到了监测节点上传的数据包，取出数据包。
4. 解析数据，并将接收到监测节点数据放到监测节点数据储存模块和最新接收消息队列中。
5. 调用监测点数据储存模块的更新函数，更新汇聚节点中储存的所有监测节点的状态。

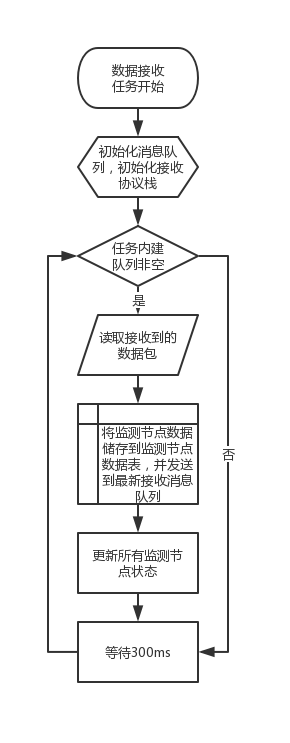


图 4.8监测节点数据更新流程图

### 汇聚节点人机交互任务

该任务的主要作用为提供人机交互界面逻辑，使之可以显示所有监测节点的温湿度数据，并可以配置监测节点坐标[20]。

人机交互任务主要流程如下：

1. 人机交互任务开始执行后，默认进入overview页面，在该页面中，程序通过读取监测节点数据储存模块获得节点数据，之后调用图形显示模块来显示所有非关机状态的监测节点数据。
2. 若检测到用户点击了页面切换按钮，则进入config页面，在该页面中，程序通过调用图形显示模块，读取用户点击的不同位置来实现相应的功能。
3. 若在config页面检测到用户点击了页面切换按钮，则首先保存config页面配置的节点坐标，再将页面切换回overview。

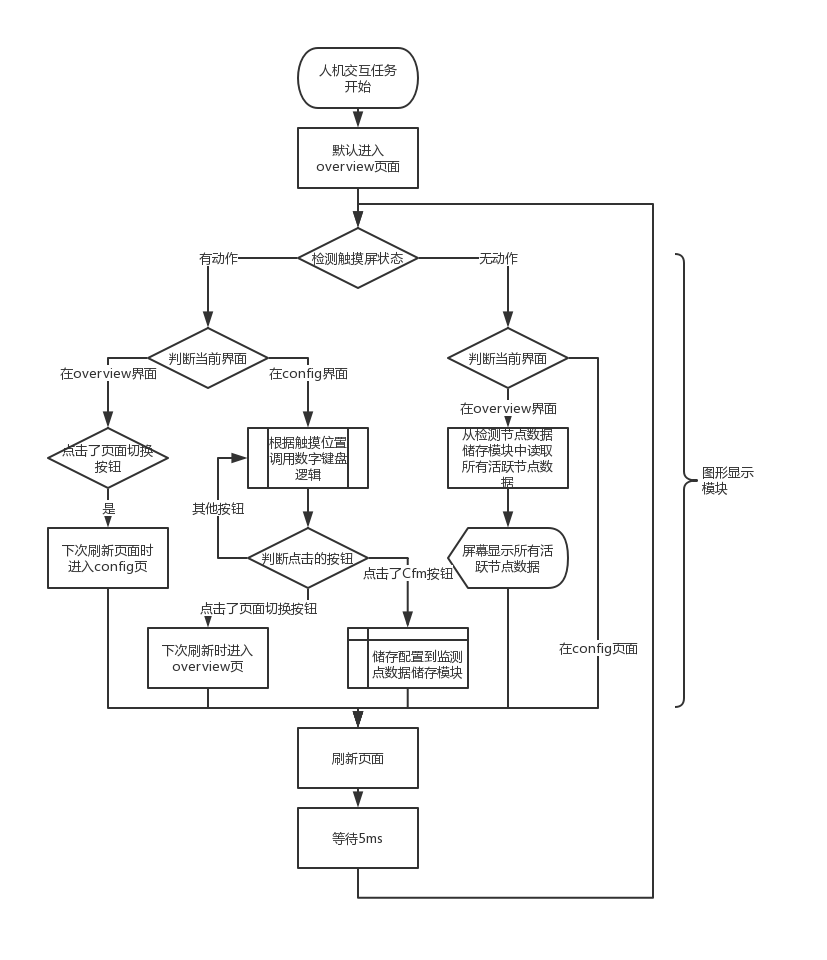


图 4.9汇集节点显示任务流程图

图 4.9、图 4.10和图 4.11展示了开机后初始的overview页面以及经config页面配置后的overview页面。



图 4.10 overview页面

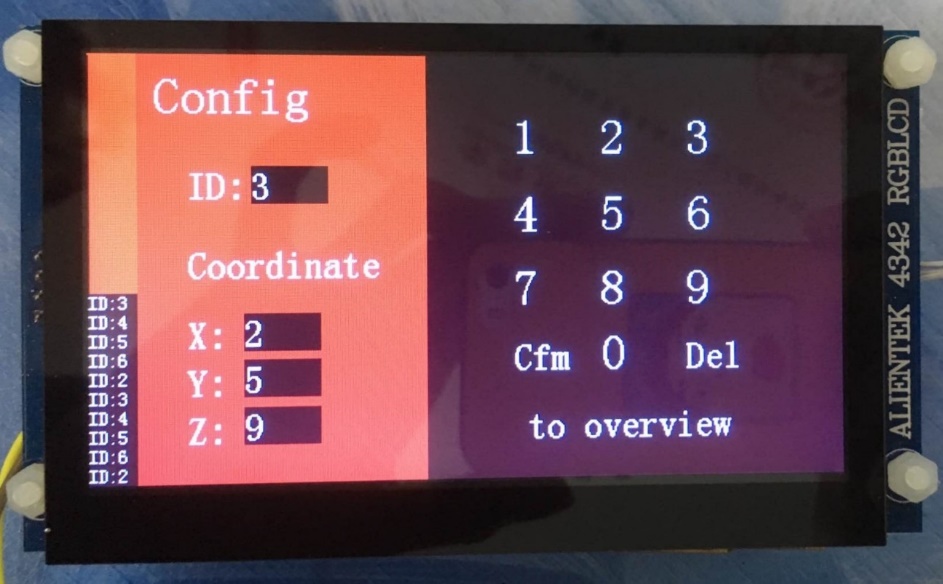


图 4.11 config页面



图 4.12经config页面配置后的overview页面

### 汇聚节点Modbus功能的实现

#### 实现软件接口

FreeMODBUS是一个专门为嵌入式领域而设计的Modbus协议组件[21]，本课题通过对该组件提供的接口及回调函数进行编程，使汇聚节点成为了Modbus RTU从机。

在FreeMODBUS官网下载源码后，根据移植手册，需要对portserial.c，porttimer.c，portevent.c，port.h这4个文件进行修改。

port.h文件中对FreeMODBUS使用的宏进行了定义。在本工程中，由于使用了µC/OS-III操作系统，所以需要对临界区代码进行补充定义：

porttimer.c文件为FreeMODBUS对定时器的相关实现。由于Modbus协议栈需要通过计算接收时间来检测一个帧的结束，所以需要对一个定时器进行配置，包括定时器的初始化函数，使能/失能函数和超时回调函数。

portserial.c文件包含了FreeMODBUS对串口的相关实现。在该文件中，需要实现实现串口的初始化函数，使能/失能发送或接收中断函数，发送或接收中断回调函数以及接收和发送单个字节函数。

portevent.c包含了FreeMODBUS中事件标志组的相关实现。由于在µC/OS-III操作系统中有对于事件标志组的支持，故将在该文件中使用操作系统提供的函数实现事件标志组即可。

以下展示了移植FreeMODBUS需要进行操作的文件及函数：

表 4.1 FreeMODBUS移植相关函数

Table 4.1 FreeMODBUS porting related functions

|  |  |  |
| --- | --- | --- |
| 文件 | 函数 | 作用 |
| port.h | ENTER\_CRITICAL\_SECTION( ) | 进入临界区 |
| EXIT\_CRITICAL\_SECTION( ) | 退出临界区 |
| porttimer.c | xMBPortTimersInit( ) | 定时器初始化 |
| vMBPortTimersEnable( ) | 使能定时器 |
| vMBPortTimersDisable( ) | 关闭定时器 |
| prvvTIMERExpiredISR( ) | 定时器溢出回调 |
| portserial.c | xMBPortSerialInit( ) | 串口初始化 |
| vMBPortSerialEnable( ) | 使能串口 |
| xMBPortSerialPutByte( ) | 读取1Byte数据 |
| xMBPortSerialGetByte( ) | 发送1Byte数据 |
| pxMBFrameCBByteReceived( ) | 接收完成中断回调 |
| pxMBFrameCBTransmitterEmpty( ) | 发送完成中断回调 |
| portevent.c | xMBPortEventInit( ) | 事件标志组初始化 |
| xMBPortEventPost( ) | 发送事件 |
| xMBPortEventGet( ) | 获得事件 |

#### 修改保持寄存器回调函数

实现FreeMODBUS的相关接口后，汇聚节点即可被视为一个支持Modbus协议的从机设备，然而，此时汇聚点虽然有完整的Modbus RTU从机协议栈，却无法对相应的主机命令做出正确的反应。

在FreeMODBUS中存在一种命令回调函数，每当接收到主机特定的命令时，对应的命令回调函数便被调用。在该类函数中，可以对相应的寄存器进行修改，更新等操作，而使从机返回合适的数据。本课题通过对保持寄存器命令回调函数进行修改，来实现返回JSON字符串的功能。

修改后的保持寄存器命令回调函数流程如下：

1. 读取寄存器命令回调函数被调用后，判断是否为读取操作，若为读取操作，则继续进行，否则不对寄存器进行操作。
2. 从最新接收数据消息队列中取出一条消息。
3. 提取消息中的监测点坐标，温度值和湿度值，并将其装入JSON字符串中。
4. 将JSON字符串写入连续的保持寄存器。

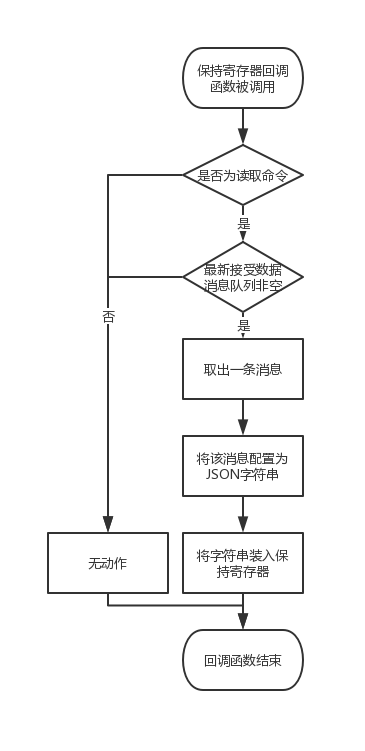


图 4.13保持寄存器命令回调函数流程图

#### 汇聚节点Modbus任务

汇聚节点Modbus任务的作用主要为处理上位机的Modbus请求。

在4.3节中，本课题确定了汇聚节点与上位机的通讯方案，上位机通过串口向汇聚节点周期性的发送读取汇聚节点保持寄存器请求，汇聚节点接收到请求后，将JSON字符串放到保持寄存器中，供上位机读取。

Modbus任务的主要流程如下：

1. 初始化并使能FreeMODBUS模块。
2. 如果上位机发送了读取保持寄存器的请求，则通过FreeMODBUS调用保持寄存器回调函数，实现将最新的数据转换为JSON字符串后写入保持寄存器。
3. 写入完成后，FreeMODBUS会依照主机命令，将对应的保持寄存器数据发送到主机，以实现JSON字符串的发送。
4. 等待50ms，以提供时间供其他任务调度。

# 节点间通讯的设计与实现

在监测节点和汇聚节点无线传输模块的选型中，本课题选用了基于ZigBee协议的XL2530-232AP1模块，该模块上电后可以进行自组网，能够实现物理层传输的功能。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

在无线传输模块已经实现了物理层的透明传输的基础上，由于监测节点需要向汇聚节点传输的数据包可能有多种类型，为了能够便于节点间通讯，就需要加入一个中间层作为物理层和顶层数据包的媒介，因此在本课题中，节点间的通讯协议分为三层[22]。

最底层为物理层，该层通过XL2530-232AP1模块，实现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，其作用为向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。

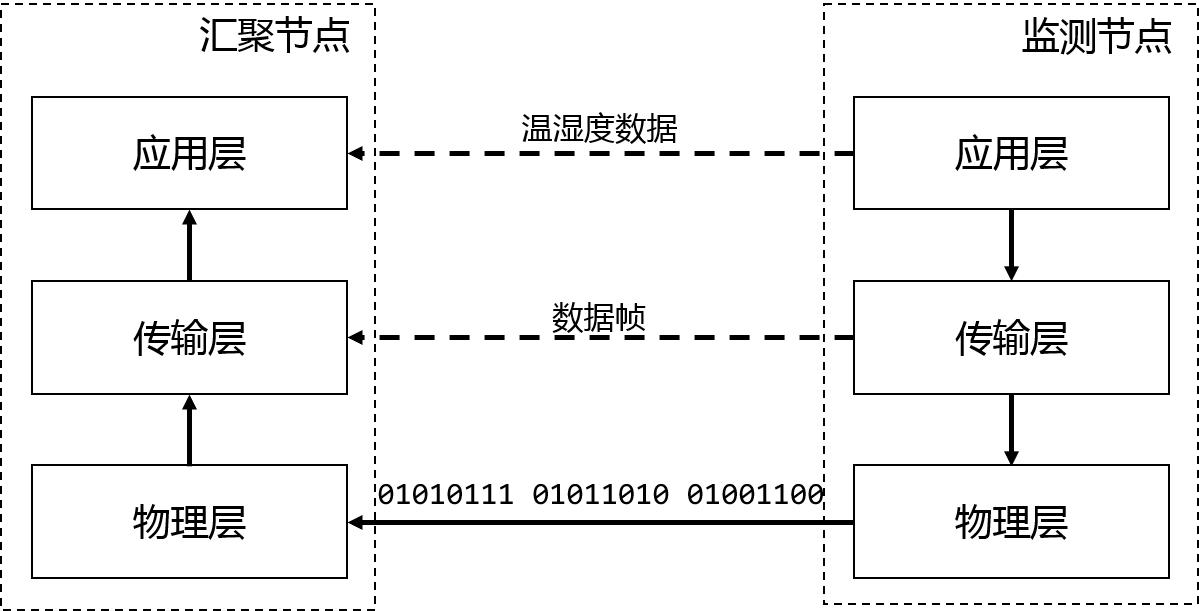


图 5.1节点间通讯协议层次划分

## 库函数的串口接收机制简介

由于节点间数据的通讯均是建立在串口基础之上，因此本节首先介绍在STM32微处理器上对串口编程使用的库函数。

ST公司提供了多种函数库对单片机上寄存器进行了封装，以便于对STM32型微处理器进行编程，常见的有传统的STD(Standard Peripheral)库和较新的HAL(hardware abstraction layer)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，一般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，以实现多个字节同时处理。因此，若使用一定策略，就可以实现调用缓冲区满回调函数时接收完成1个数据帧，极大的简化了代码的复杂程度。

本课题设计的两种传输层通讯协议均使用HAL库的串口接收缓冲区实现。

## 节点间传输层通讯协议的设计

本节主要介绍监测节点与汇聚节点之间的通讯协议的设计。在设计过程中，本课题使用了两种不同的设计思想，分别在传输层实现了一种定常协议方案和一种不定长协议方案。在本节的最后，对这两种通讯协议的性能进行了比较，最终使用了不定长协议方案。

### 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A Byte，传输层每帧承载应用层数据的空间为t Byte，则切片数量n为：

|  |  |  |
| --- | --- | --- |
|  |  | （5.1） |

传输层对数据包进行切片后，装到连续的数据帧中，待发送完所有帧，即实现了一个应用层数据包的传输。

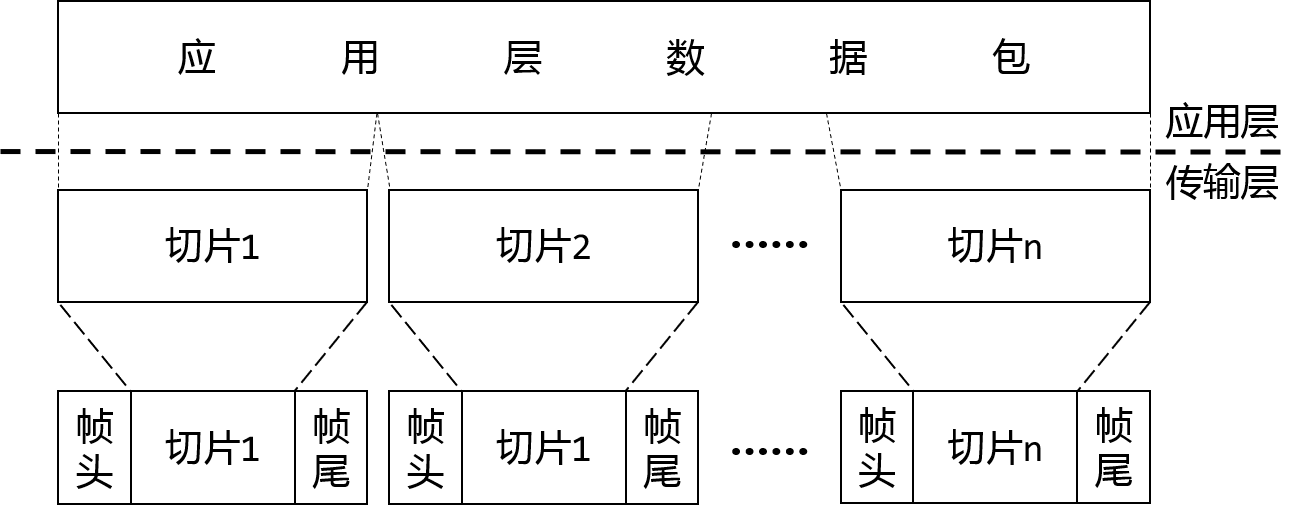


图 5.2对应用层数据包进行切片

#### 定常帧结构

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无线模块的控制命令。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），这两个字节仅在发送节点向无线模块通讯时需要，无线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字节用于承载应用层的数据。
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。



图 5.3定常帧结构

#### 发送及接收协议栈

传输层发送协议栈负责接收应用层传输的数据包，进行处理后调用物理层将数据发送至汇聚节点。

其程序流程如下：

1. 接收到应用层数据包后，计算数据包长度等信息。
2. 按照每帧的长度对数据包进行切片，并将切片后的数据按顺序放到帧中，计算每一帧的控制信息。
3. 按照顺序发送所有帧。

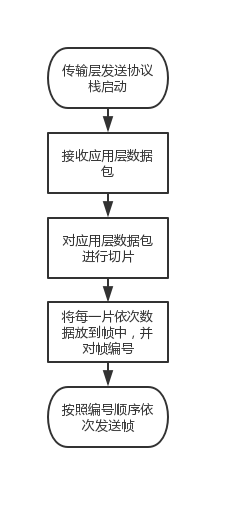


图 5.4定长帧发送流程图

传输层接收协议栈负责通过物理层接收数据，对数据进行整合后，还原为应用层数据包。

其程序流程如下：

1. 在接收缓冲区满后则认为一帧接收完成，取出消息。
2. 使用CRC16字段验证接收到的帧是否传输正确，如果校验正确则继续进行，否则丢弃该帧。
3. 查看是否存在数据包缓冲区。若不存在，说明接收到了一个新数据包，则新建一个数据包缓冲区，并将该帧放到缓冲区中。若存在，则说明上一个数据包尚未接收完成，此时需要验证本次接收到的帧是否与上一次接收到的帧连续，如果连续，则放入，如果不连续，则认为整个数据包接受失败，销毁数据包缓冲要去，等待接收新的数据包。
4. 待数据包缓冲区满后，取出数据包数据，校验整个数据包，若校验成功，则将数据包发送到数据包接受队列，等待其他任务进行处理。否则丢弃本次数据，等待接受新的数据包。

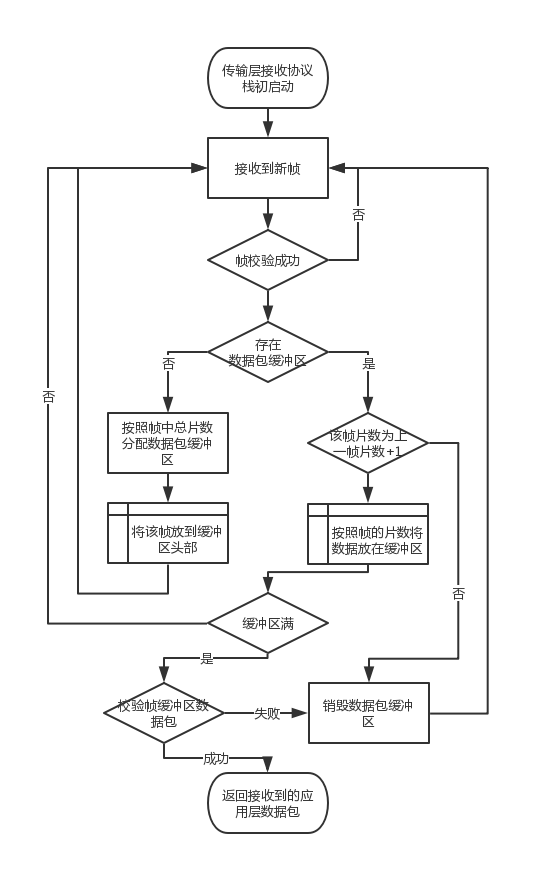


图 5.5定长帧接收流程图

### 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，相比于长度的固定缓冲区，该方案通过动态改变缓冲区大小，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，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立刻改变接收缓冲区长度，一次接收完帧的后半部分，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，帧数据部为不定长的应用层数据包，对其不做任何处理。

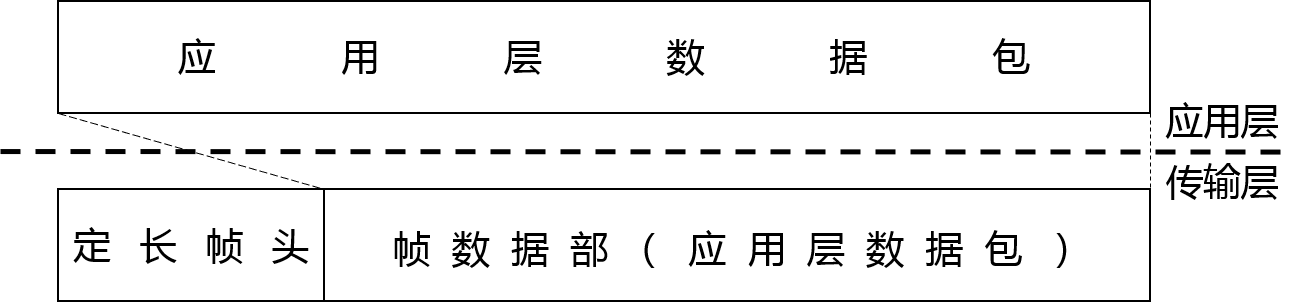


图 5.6应用层数据直接装入不定长帧的后部

#### 不定长帧结构

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。这两个字节仅在发送节点向无线模块通讯时需要，无线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，由于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。



图 5.7不定长帧结构

#### 发送及接收协议栈

在发送不定长帧时，首先通过传入的应用层数据包计算帧头参数，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。

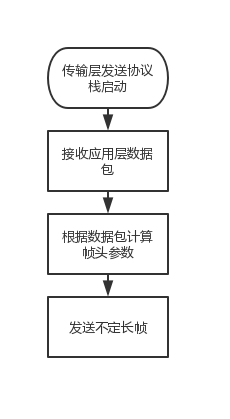


图 5.8不定长帧发送流程图

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1字节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1字节，就将该字节放到队尾，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，开始新一轮的接收。

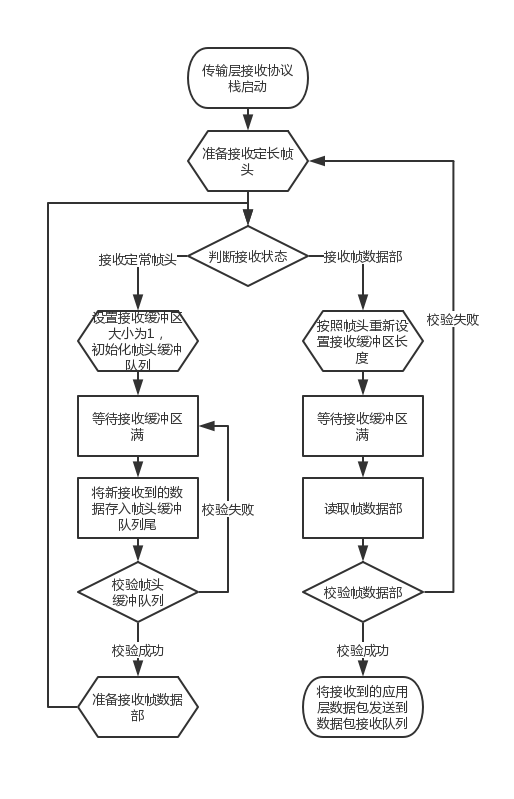


图 5.9不定长帧接收流程图

### 传输层通讯协议的选择

虽然定长帧和非定长帧两种方案都实现了传输层的通讯，但是在测试中也发现了一些问题。

对于定长帧协议，如果在一帧发送未完成时启动接收协议栈，就会造成1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1帧的后半部分和后1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。

测试表明，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，若应用层数据包过长，就会导致帧头中标识数据包长度的字节溢出，导致无法正常接收。因此，不定长帧对应用层数据包的长度存在较大的限制。但是由于监测节点仅需要向汇聚节点传输单次的温湿度数据，单个数据包长度较短，因此该问题并未出现。

测试结果表明，不定长帧协议能够较好的完成通讯需求，不仅利用了HAL库的缓冲区机制，而且帧中的信息密度较高，也减少了带宽的浪费。

## 节点间应用层的设计

应用层是节点间数据传输的最顶层，该层直接传输监测节点采集的温湿度数据，在程序中通过C语言的一个结构体进行实现。

该结构体声明如下:

|  |
| --- |
| typedef struct nodeData  {  u16 localShortAddress;//监测节点的短地址  float temperature;//采集到的温度值  float humidity;//采集到的湿度值  u8 controlWord;//保留的控制字  }NodeDataStruct; |

# 上位机的设计

## 上位机的功能需求

依据总体设计方案，上位机需负责接收汇聚点上传的数据，并提供超限报警，数据可视化等功能。

由于在之前的工作中，汇聚节点已经可以向上位机通过Modbus协议传输JSON字符串，因此需设计一款软件，该软件不仅能够解析接收到的数据，对异常的温湿度进行报警，还要具有记录数据，并对数据进行可视化的功能。

## 上位机的实现

由于Python语言语法简单，第三方模块非常丰富，可以很方便的开发产品原型，因此，本课题选用该语言作为上位机的编程语言[23]。

为实现上位机的功能需求，本课题调用了以下几个Python模块。

1. Tkinter是Python的标准Tk GUI工具包的接口。上位机使用该模块来实现图形界面
2. MinimalModbus是一个非常易用的Modbus协议组件。上位机使用该模块来实现Modbus主机的功能，用以和汇聚节点通讯。
3. json是Python内建的解析JSON数据的模块。上位机使用该模块来解析从汇聚节点接收到的数据。
4. pyexcel是一个可以读写多种excel格式的模块。考虑到本课题数据量较小，故上位机采用读写excel的方式来读取和储存数据。
5. matplotlib提供了一组类似MATLAB画图函数的程序接口，用以创建各种类型的图表。上位机使用该模块来对收集到的数据进行可视化。

以下为上位机具体的运行流程：

1. 软件打开后首先打开接收端口，若用户点击了开始接收按钮，则上位机以2s的周期向汇聚节点发送读取数据请求。
2. 待接收到汇聚节点的数据后，首先提取出其中的JSON字符串，然后对JSON进行解析，解析出最新接收到的坐标和温湿度数据。
3. 判断接收到的数据是否存在异常，如果出现温度或湿度大小异常，则启动报警。
4. 为防止数据丢失，每隔一小时就将接收到的数据存入excel中。此外，若用户点击停止按钮，则停止接收数据，并将新接收到的数据存入excel中。
5. 若用户点击出图按钮，则从excel中读取所有数据，并调用响应的作图接口绘制温湿度曲线。

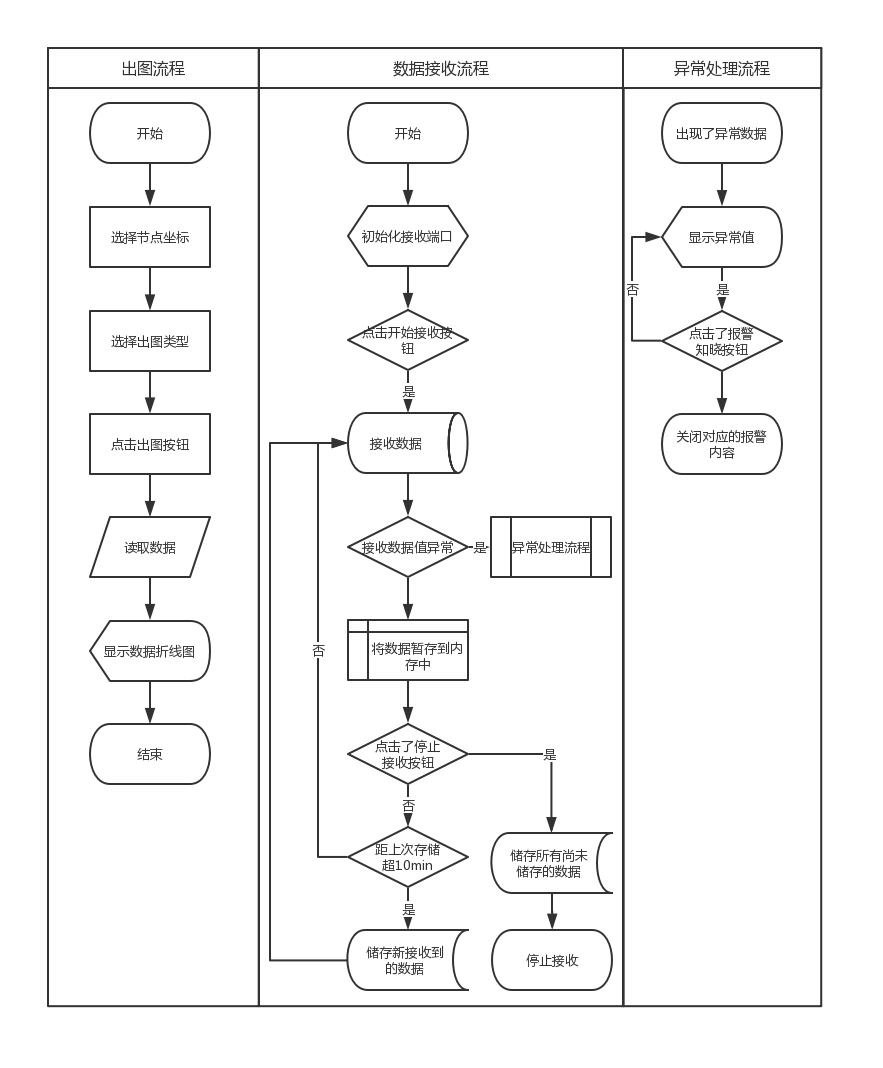


图 6.1上位机程序流程图

以下为本课题设计的上位机湿度过高报警示例，若点击对应的报警按钮，则关闭报警。

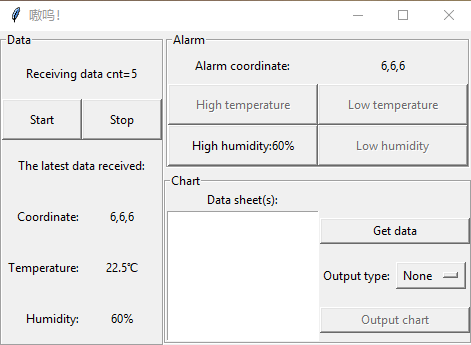
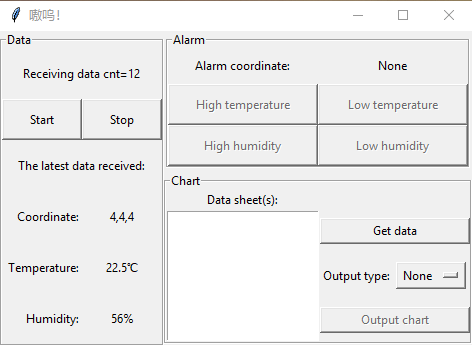
 

图 6.2低湿度报警（左：激活，右：关闭）

若希望查看历史数据折线图，则需首先停止数据接收，点击Get data按钮获取数据后选择输出类型，最后点击Output chart按钮，即可得到绘制的图像。

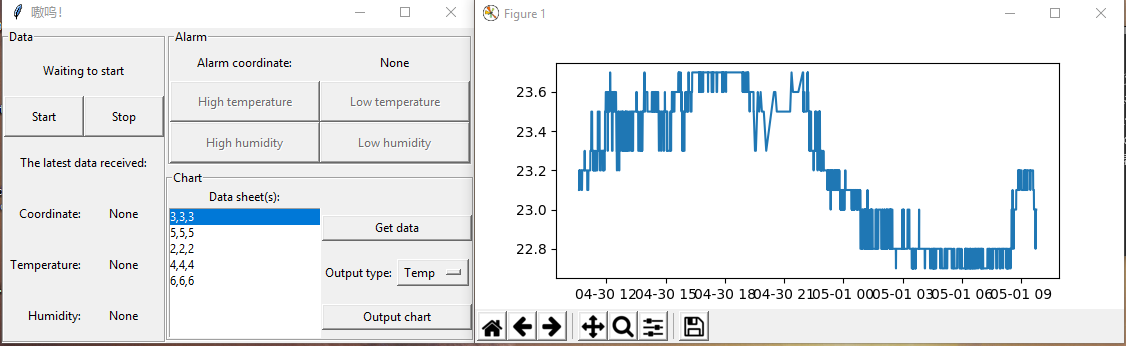


图 6.3输出历史数据折线图

# 调试与验证

# 总结与展望

本章主要内容为对课题研究内容的总结，以及对本课题未来更进一步的深化做出展望。

## 论文工作总结

随着21世纪以来信息革命的进一步深化，对数据中心运维要求不断提高，如何更及时的监控数据中心环境，如何进一步降低数据中心能耗等问题愈发受到重视。本课题正是着眼于数据中心机房内温湿度这一关键点，设计了一整套可以实时反映机房内不同空间位置温湿度的监控系统，该系统不仅可以对数据进行采集和可视化，而且可以对异常情况做出实时报警处理，有效的降低了服务器机房的运维难度。本课题完成的具体工作如下：

1. 为了能够采集房间内环境数据，本课题经过硬件设计，PCB焊接及软件调试，实现了一种基于ZigBee通讯协议的微型无线低功耗温湿度监测节点，经测试，这种节点能够较为真实的上传所处空间位置的温湿度数据。
2. 为了便于汇集节点数据，并提供与上位机的接口，本课题通过软硬件设计，实现了一种能够收集监测节点数据，并使用Modbus协议向其他设备传输数据的汇聚节点设备，此外，该设备本身也具有LCD触摸屏，可以实时显示收集到的数据，以及对监测节点坐标进行配置。
3. 为实现监测节点和汇聚节点的通讯，本课题设计了两种传输协议，并分别对这两种协议进行了测试，最后选用了更加高效的非定长传输协议作为传输标准。
4. 在电脑上实现了一个能够通过COM口进行Modbus通讯的上位机，使用该上位机与汇聚节点连接后，可以实时显示节点监测最新的温湿度数据，还具有超限报警，数据可视化等功能。
5. 对上述系统进行了性能与功能测试。

## 未来展望

在对本课题所设计的系统进行测试后，虽然能够满足预定的要求，但是还有一些不足可供探索和优化。

1. 本课题所设计的汇聚节点理论上最多可以支持255个监测节点接入，但是由于经费及时间原因，仅使用了5个节点进行测试，在未来可以接入更多的设备来测试汇聚节点的稳定性。
2. 考虑到成本，本课题监测节点所使用的温湿度传感器精度较低，微处理器功耗较高，如果条件允许，可以为监测节点配备精度更高的传感器以及能耗更低的微处理器，以进一步延长监测节点的运行时间。
3. 由于本课题的侧重点在数据的收集和可视化，在此基础上，通过收集分析这些数据，还可以更好的了解机房内的分湿度变化情况，进而可以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。

参考文献

1. 中国信息通信研究院开放数据中心委员会.《数据中心白皮书》 2018
2. 工信部. 工业和信息化部 国家机关事务管理局 国家能源局关于加强绿色数据中心建设的指导意见. http://www.gov.cn/xinwen/2019-02/14/content\_5365516.htm. [EB/OL]. 2019
3. Google.Efficiency: How we do it. [EB/OL]. https://www.google.com/about/datacenters/efficiency/internal, 2019.
4. Amazon Web Services. Cloud Computing, Server Utilization, & the Environment.[EB/OL]. https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment, 2019.
5. 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].
6. Rodriguez, M.G., Ortiz Uriarte, L.E., Yi Jia, Yoshii, K., Ross, R., Beckman, P.H.. Wireless sensor network for data-center environmental monitoring[P]. Sensing Technology (ICST), 2011 Fifth International Conference on,2011.
7. 沈华峰. 基于阵列传感器的数据中心温度场可视化系统研究[D].浙江大学,2013.
8. 尤海鹏. 基于Ganglia的数据中心监控平台设计[D].山东大学,2014.
9. 王雪.智慧农业远程监控系统设计[J].农业与技术,2019(08):29-30+146.
10. 张铮,曹守启,朱建平,陈佳品.面向实时监测的无线传感网低功耗通信策略[J].仪器仪表学报,2019,40(02):257-264.
11. 杨瑞峰,王雄,郭晨霞,张鹏.基于ZigBee无线传感网络环境监测系统设计与应用[J].电子器件,2017,40(03):760-765.
12. 廖建尚.基于物联网的温室大棚环境监控系统设计方法[J].农业工程学报,2016,32(11):233-243.
13. 袁仲达,马昱春,边计年,赵康.基于运行时基本代码块的可重构软硬件协同设计[J].清华大学学报(自然科学版),2013,53(09):1347-1355.
14. 彭安妮,周威,贾岩,张玉清.物联网操作系统安全研究综述[J].通信学报,2018,39(03):22-34.
15. 关新平,陈彩莲,杨博,华长春,吕玲,朱善迎.工业网络系统的感知-传输-控制一体化:挑战和进展[J].自动化学报,2019,45(01):25-36.
16. Xu Fangmin,Ye Huanyu,Cui Shaohua,Zhao Chenglin,Yao Haipeng.Software defined industrial network architecture for edge computing offloading[J].The Journal of China Universities of Posts and Telecommunications,2019,26(01):49-58.
17. 信俊昌,王国仁,李国徽,高云君,张志强.数据模型及其发展历程[J].软件学报,2019,30(01):142-163.
18. 高静,段会川.JSON数据传输效率研究[J].计算机工程与设计,2011,32(07):2267-2270.
19. Micrium. µC/OS-III Documentation. [EB/OL]. https://doc.micrium.com/pages/viewpage.action?pageId=10753180, 2018.
20. 李海波,赵晓峰,付琛,刘昌珍.基于Web技术的农田恒压灌溉远程监控系统测试研究[J].排灌机械工程学报,2016,34(01):86-92.
21. 李东东,崔龙龙,林顺富,刘庆强,覃子珊,任婧玮.家庭智能用电系统研究及智能控制器开发[J].电力系统保护与控制,2013,41(04):123-129.
22. 张波,罗锡文,兰玉彬,黄志宏,曾鸣,李继宇.基于无线传感器网络的无人机农田信息监测系统[J].农业工程学报,2015,31(17):176-182.
23. 王伟珍,李荐民,王小兵,花正东,许浔江.基于EPICS的上海光源固定辐射监测系统[J].核电子学与探测技术,2008,28(06):1156-1162.