

**毕业设计**

**设计题目 基于物联网的建筑结构健康监测系统服务器软件设计**

**学生姓名 丁正超**

**学 号 20102685**

**专业班级 计算机科学与技术10-2班**

**指导教师 魏振春**

**院系名称 计算机与信息学院**

**2014 年 6 月 3 日**

目 录

[摘 要 1](#_Toc390194704)

[Abstract 2](#_Toc390194705)

[1 引 言 3](#_Toc390194706)

[2 概 述 4](#_Toc390194707)

[2.1 建筑结构健康监测的国内外现状 4](#_Toc390194708)

[2.1.1 国外研究现状 4](#_Toc390194709)

[2.1.2 国内研究现状 4](#_Toc390194710)

[2.1.3 物联网应用于建筑结构健康监测 5](#_Toc390194711)

[2.1.4 建筑结构健康监测技术发展 5](#_Toc390194712)

[2.2 建筑结构健康监测系统研究意义 6](#_Toc390194713)

[2.2.1 研究背景 6](#_Toc390194714)

[2.2.2 研究意义 7](#_Toc390194715)

[3 需求分析 8](#_Toc390194716)

[3.1 任务概述 8](#_Toc390194717)

[3.1.1 目标 8](#_Toc390194718)

[3.1.2 运行环境 8](#_Toc390194719)

[3.2 功能需求 9](#_Toc390194720)

[3.4 性能需求 9](#_Toc390194721)

[4 概要设计 10](#_Toc390194722)

[4.1 硬件环境 10](#_Toc390194723)

[4.2 软件环境 10](#_Toc390194724)

[4.3 开发环境 11](#_Toc390194725)

[4.4 总体设计 12](#_Toc390194726)

[4.4.1 硬件架构 12](#_Toc390194727)

[4.4.2 软件架构 13](#_Toc390194728)

[4.5 数据库设计 14](#_Toc390194729)

[4.6 通信协议设计 16](#_Toc390194730)

[4.6.1 整体说明 16](#_Toc390194731)

[4.6.2 地址编码方案 17](#_Toc390194732)

[4.6.3 通信包格式设计 18](#_Toc390194733)

[4.6.4 消息操作码编码方案 19](#_Toc390194734)

[5 详细设计 21](#_Toc390194735)

[5.1 软件流程描述 21](#_Toc390194736)

[5.2 关键模块详细设计 22](#_Toc390194737)

[5.2.1 监听模块 22](#_Toc390194738)

[5.2.2 数据解析模块 23](#_Toc390194739)

[5.2.3 命令处理模块 25](#_Toc390194740)

[5.3 难点和解决方案 27](#_Toc390194741)

[5.3.1 并行处理 27](#_Toc390194742)

[5.3.2 下载配置表 28](#_Toc390194743)

[5.3.3 解析采集时间 29](#_Toc390194744)

[5.3.4 解析采集数据 30](#_Toc390194745)

[6 系统测试 33](#_Toc390194746)

[6.1 命令解析测试 33](#_Toc390194747)

[6.1.1 入网请求测试 33](#_Toc390194748)

[6.1.2 全网对时测试 34](#_Toc390194749)

[6.1.3 通道重启测试 35](#_Toc390194750)

[6.2 数据解析测试 35](#_Toc390194751)

[6.3 压力测试 36](#_Toc390194752)

[结 论 37](#_Toc390194753)

[致 谢 38](#_Toc390194754)

[参考文献 39](#_Toc390194755)

基于物联网的建筑结构健康监测系统 服务器软件设计

摘 要：

近年来各种大型建筑坍塌事故频发，给人民的生命和财产造成巨大损失，并带来极坏的社会影响。然而，现有的监测手段无法适应海量数据的监测和实时在线监测的要求。物联网技术作为当今计算机应用领域的主流技术，将其应用于建筑结构健康监测可以实时、快捷、高效地对重大建筑工程进行监测，可以及时有效地发现待监测建筑的结构损伤，准确评估其安全性。

本论文着重于建筑结构健康监测系统服务器软件的设计与实现，通过该软件可以实时获取底层传感器采集到的建筑结构数据，经过解析处理后存储于MicrosoftSQlServer2008数据库中。同时，服务器软件还可以解析Excel配置表，将Excel数据解析为hex记录下发用于配置底层硬件工作模式。此外，服务器软件还能响应客户端软件的命令，根据具体命令提供相应的服务操作。

关键词：

建筑结构健康监测，服务器软件设计，数据解析。

Structural Health Monitoring System Based On IOT Technology Server Software Design

Abstract：

In recent years, the large building collapse accidents occur frequently, causing huge loss to lives and property of the people, and bring a very bad impact on society. However, the existing monitoring tools can not adapt to the requirements of real-time online monitoring of massive data. The IOT technology as the mainstream of today's computer applications technology will be applied to the construction of structural health monitoring. It can help to monitor the structural health of buildings in a real-time, fast, and efficient way. It can also find the structural damage of buildings timely and give out an accurate security assessment.

This thesis focuses on the design and implementation of structural health monitoring system server software. The server software can get the real-time data of building structural health collected by the underlying sensors. The data of building structural health will be stored in Microsoft SQL Server 2008 after parsing process. Meanwhile, the server software can also resolve the Excel table which is used to deploy the mode of underlying sensors.

Keywords**:**

Structural Health Monitoring，Server Software Design，Data Parse.

1 引 言

近年来建筑安全事故屡有发生，例如：上海闵行区在建13层住宅楼整体倒塌事故；重庆涪陵化工厂垮塌事故等，对经济财产和人身安全造成了巨大的损害。因此，对建筑结构的各种负载、环境因素，以及结构可靠度、耐久度的监测显得必要。

基于物联网技术的建筑结构监测系统，能够实时、快捷、高效地对重大建筑工程进行监测，可以及时有效地发现待监测建筑的结构损伤，准确评估其安全性，并预测其结构的性能变化、施工安全性和剩余寿命，帮助做出维护决策。

本论文设计的软件，是建筑结构健康监测系统的服务器软件，主要负责接收和解析采集节点通过无线传感网络上传的建筑结构数据，然后将解析后的数据存入服务器数据库。同时，还能够配置底层硬件的采集频率和采集通道号。

这篇论文包括以下几部分：

2 概述部分：介绍了建筑结构健康监测的国内外研究现状和研究的背景意义，并说明了本文设计的服务器软件在建筑结构健康监测中发挥的作用。

3需求分析部分：从功能上和性能上对建筑结构健康监测系统服务器软件进行了需求分析。

4 概要设计部分：重点说明了建筑结构健康监测系统服务器软件的架构，开发环境，数据库设计，通信协议设计。

5详细设计部分：具体描述了建筑结构健康监测系统服务器软件的工作流程，详细说明了重要功能模块的实现方法以及发开过程中遇到的难点和其解决方案。

6 软件测试部分：介绍了系统的测试与运行情况，并给出了相应的截图。

2 概 述

## 2.1 建筑结构健康监测的国内外现状

建筑结构健康监测是指运用部署在现场的传感器，通过长期监测建筑物荷载及环境变量，并分析结构安全性能、结构动力响应等一系列结构特性，经过评价分析得到检测结构是否安全与健康的结果。

### 2.1.1 国外研究现状

在土木工程领域中，结构健康监测概念早在1985年就已经提出，最初主要应用在桥梁健康监测方面，近十几年来，逐步推广到针对建筑结构的健康监测开展了大量的研究。随着土木工程结构设计的大型化、复杂化以及智能化，结构健康监测的内容逐渐变得丰富起来，不再是单纯的载荷应力监测，而要向结构损伤检测、快速损伤定位、结构剩余寿命预测等方向努力。

国外尤其是北美、日本及欧洲研究的比较早，应用较广泛。最先在桥梁上应用，如Sunshine Skyway桥，对桥面、桥塔和桥墩截面的应变和温度及环境温度和湿度进行了长期健康监测。在应用于桥梁健康监测上的技术逐渐完善和成熟后，逐步发展到其他民用建筑上，如为德国的莱特火车站大楼而设计的健康监测系统，对屋顶支柱的垂直位移差进行了监控。意大利某安装了健康监测系统监测外部环境对某教堂的长期损害作用。

### 2.1.2 国内研究现状

在国内，近几年结构健康监测系统的应用逐渐增多，如杭州湾跨海大桥南航道桥健康监测工程、深圳湾公路大桥结构监测、北京地铁四号线南站施工监测、国家数字图书馆工程基坑监测、中央电视台新台址施工监测、天津津塔施工过程监测中钢大厦施工监测、国家体育场(鸟巢)钢结构施工监测、奥运会羽毛球馆新型预应力弦支穹顶结构全寿命健康监控、水立方钢膜结构安全监测、昆明新机场结构抗震及安全性监测。

### 2.1.3 物联网应用于建筑结构健康监测

全球信息技术产业分别在1965年、1980年、1995年前后发生了以大型机、个人计算机普及、互联网为标志的变革，人称“十五年周期定律”。在当下的“十五年”，一场新的变革正悄然兴起，那就是“物联网”（Internet of things，简称IOT）。在“物联网”这个领域中，我国的技术研发水平正处于世界前列。

物联网通过射频识别（RFID）、红外感应器、全球定位系统、激光扫描器等信息传感设备，按约定的协议，把物品与互联网连接起来，进行信息交换和通讯，以实现智能化识别、定位、跟踪、监控和管理的一种网络。物联网把物理世界与信息网络系统连接起来，把物理基础设施与IT基础设施整合在一起，让生产制造、经济运行、社会管理以至个人生活有了全新的基础设施，从而形成“智慧地球”。

将物联网技术应用于重大建筑结构健康监测与安全监控，可以解决目前建筑健康监测与全寿命安全监控的关键技术难题，实现监测数据采集实时化、传输无线化、评判智能化，满足重大建筑工程健康监测与安全监控的实际需要，将成为建筑结构健康监测的必然发展趋势。

### 2.1.4 建筑结构健康监测技术发展

总体来说，结构健康监测技术的发展大致经历了以下三个阶段：第一阶段，是以结构监测领域的专业人士的外部观察和专业经验等为基础，对诊断处的结构信息只能作简单的数据处理，得出大致的结论；第二阶段借助传感技术以及动态测试技术，以信号处理分析和建模计算为基础，分析结构的健康程度，由于现代传感技术和测试技术的发展，此阶段在工程中得到了大量的应用，其准确度也超过第一阶段；第三阶段，近些年为了对一些重点大型复杂结构进行健康诊断，进入了以数据处理为核心，信号处理与知识处理相融合的智能发展阶段。

传统的结构健康监测监控技术在工程实践中存在以下问题，例如：(1)采集速度慢、精度差、误差较大，而且布线较为复杂，在应用上存在较大的局限性；(2)信号调制、数据接口、系统整合等技术性问题还有待进一步研究，以提高系统的工作效率和抗干扰能力，降低成本；(3)如何在含噪声的环境中利用尽可能少的传感器获取全面、精确、灵敏的结构参数信息，并使所测结果具有良好的实用性和代表性；(4)数据采集系统在实时性、自动化、智能化等方面有待进一步提高；(5)已有的采集系统中，往往存在监测项目种类不足，而个别项目的规模又过于庞大等问题，尤其在数据的管理方面，海量数据的存储与处理还未得到有效解决。

物联网的应用使建筑健康监测技术更加完善，改善传统监控技术的不足。运用物联网进行重大建筑的监控，使得监控更加全面，采集速度快、精度高、实时性好、自动化程度高，实现在线监测，使得结构在进行安全评判时更具有科学性。所以，在施工阶段和使用阶段，这种监测技术可以达到及时有效的保证建筑的安全性，具有很好的实际意义。

2.2 建筑结构健康监测系统研究意义

### 2.2.1 研究背景

当前我国正处于城市化建设进程进一步推进、重大建筑工程和基础设施大规模建设的高峰时期，我国2000年完成建筑业总产值为1.25万亿元，2005年已跃升至3.46万亿元；2006年，我国建筑业实现增加值11856.5亿元，占当年GDP的5.66％；2009年全国共完成建筑业总产值75864亿元；2010年我国建筑业完成总产值95206亿元。

城市化建设过程中出现了一批大跨、高层、异型建筑物，如2008年北京奥运工程国家体育馆“鸟巢”、国家游泳中心“水立方”、国家大剧院、2010年上海世博会系列场馆、武汉体育馆、常州会展中心、首都机场航站楼、合肥新桥机场等。

重大建筑工程作为城市标志性建筑或与广大人民群众生活密切相关，社会影响力较大的建筑物，具有使用功能要求高、建筑造型新颖、结构受力性能复杂等特点，因此建设期的施工技术复杂、使用期健康与安全性能要求高。

由于重大建筑工程施工阶段的复杂性，结构不完整性和受力性能的时变性，以及运营使用过程环境载荷作用、疲劳效应、腐蚀效应和材料老化以及其它使用不当的人为因素的影响, 建筑结构将不可避免地产生影响安全使用的隐患, 如果不加以监测控制，可能导致重大事故的发生，给人民的生命和财产造成巨大损失，并带来极坏的社会影响。综合有关资料表明，影响重大建筑工程健康与安全的事故往往是由多方面的原因造成的，如施工阶段未进行施工监测或施工监测数据不可靠，施工过程安全性分析与控制不当，导致施工阶段缺陷；运营使用阶段火灾或地震等突发性灾害下结构失效，长期使用造成的累积损失或者结构使用不当等因素。

### 2.2.2 研究意义

本次毕业设计来源于国家物联网发展专项资金项目，工信部科〔2012〕583号，基于物联网的大型建筑结构健康监测与安全监控平台研发与产业化。着重于建筑结构健康监测与安全监控平台服务器软件的设计与实现。

建筑结构健康监测与安全监控平台，能够实时监测建筑的安全和健康状态，进行分析和评估，发现异常及时预警，以便做出必要的控制，提高建筑的安全性和使用寿命，最大限度地减少不必要的财产损失与生命伤亡。

基于物联网的建筑结构健康监测及安全评估，可以及时有效的发现其结构损伤，准确评估其安全性，并预测其结构的性能变化、施工安全性和剩余寿命来做出维护决定。对于提高建筑的运营效率，减少维修和检测的成本，保障建筑在建设和使用期的安全，保障人民生命财产安全具有重大的意义。

3 需求分析

## 3.1 任务概述

建筑结构健康监测系统服务器软件，需要完成对采集数据的解析，完成对客户端命令的解析，提供对底层硬件工作模式的配置功能，提供系统日志功能。为保证系统的安全可靠性，还需提供网络超时处理、数据重传机制。

### 3.1.1 目标

1. 设计服务器软件与底层采集节点之间的通信协议包括地址编码方案，通信包格式设计，网络同步方案，操作码编码方案等。
2. 接收底层采集节点实时上传的UDP数据包，根据目标1）中订好的通信协议对其解析处理，并将解析后的数据存入 MicrosoftSQLServer2008 R2数据库。
3. 解析客户端下发的命令，根据具体的命令提供相应的服务操作。
4. 通过解析Excel配置表，将Excel数据解析为hex记录下发实现对底层硬件工作模式的配置。
5. 设计异常处理机制以保证服务器软件的稳定性和可靠性。
6. 设计系统日志功能，用于错误数据的记录和异常情况的记录。
7. 用c#编写可视化界面软件，用于服务器软件的功能测试和性能测试。

### 3.1.2 运行环境

硬件环境：

* CPU：1.5GHz及以上
* 内存：1G及以上
* 显卡：128M及以上

软件环境：

* 操作系统： Windows 7/Windows 8
* 运行平台：Eclipse
* 数据库：SQL Server 2008 R2版本及以上

## 3.2 功能需求

1. 数据接收功能：接收底层采集节点上传的UDP数据包，并对数据进行初步的合法性判断，对不合法的数据进行丢弃处理，对合法数据进行备份处理。
2. 数据解析功能：完成对采集时间的解析和建筑结构数据的解析。
3. 数据库操作功能：完成对数据库的增、删、改、查操作，提供对数据库日志的查询。
4. 下载功能： 分为采集节点的配置表下载和程序下载。采集节点配置表下载，是将存储采集节点工作频率和通道号信息的Excel表解析成hex记录下发；程序下载，是将单片机/嵌入式程序解析成hex记录下发。
5. 全网对时功能：校准所有采集节点的系统时间。
6. 节点查询功能：查询具体节点的工作模式，网络地址，工作状态。
7. 系统日志功能：完成对非法数据的记录和异常情况的记录。

## 3.4 性能需求

1. 稳定性：在PC机正常工作的情况下，服务器程序能够稳定运行。

2） 可靠性：在网络超时等异常情况下保证数据的正确性。

3） 并发性：具有一定的并发处理能力，在同一时刻能够响应n个采集节点的请求，并对其上传的数据进行解析。

4 概要设计

## 4.1 硬件环境

建筑结构健康监测系统采用典型的物联网三层架构：采集层、传输层、应用层。用到的硬件环境如下：

采集层硬件环境：应力应变传感器、裂缝传感器、挠度传感器、风力风向传感器、沉降传感器、位移传感器、倾斜传感器、加速度传感器、温湿度传感器等传感器、多传感器融合模块、太阳能采电模块、供电模块。

传输层硬件环境：ZigBee模块、WiFi模块、交换机、嵌入式网关服务器、光纤/以太网接入。

应用层硬件环境：软件可在Windows 2000/XP/Vista/server 2003操作系统上运行，暂不支持UNIX/Linux环境以及Mac OS环境。

由于开发的系统为C/S架构，因而需要路由器或交换机进行局域网的组建。可采用华为、TP-LINK、D-LINK等公司的路由器或交换机产品，路由器或交换机的网线接入口的数目≥4个。

## 4.2 软件环境

本次毕业设计用到的软件环境如下：

Microsoft Windows 7/ Windows 8操作系统或Microsoft Windows Server2008服务器操作系统下，安装并配置Microsoft SQL Server 2008数据库管理系统。

## 4.3 开发环境



图4.1 开发环境示意图

建筑结构健康监测系统服务器软件采用Java语言编写，使用Eclipse开发，数据存储于Microsoft SQL Server 2008中。服务器软件的测试工具采用C#语言编写，使用Visual Studio 2010开发。

Eclipse（集成开发环境）：Eclipse 是一个开放源代码的、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。Eclipse 附带了一个标准的插件集，包括Java开发工具。

Visual Studio 2010：Visual Studio是微软公司推出的开发环境。是目前最流行的Windows平台应用程序开发环境。其集成开发环境（IDE）的界面被重新设计和组织，变得更加简单明了。Visual Studio 2010同时带来.NET Framework 4.0、Microsoft Visual Studio 2010 CTP(Community Technology Preview)，并且支持开发面向Windows 7的应用程序。除了Microsoft SQL Server，它还支持 IBM DB2和Oracle数据库。

SQL Server 2008 R2：是由美国微软公司所推出的关系数据库解决方案，近年来它的应用范围有所扩展，已经触及到大型、跨国企业的数据库管理。SQL Server数据库的内置语言是由美国标准局（ANSI）和国际标准组织 (ISO) 所定义的SQL语言，微软公司对它进行了部分扩充而成为作业用SQL (Transact-SQL)。

## 4.4 总体设计

### 4.4.1 硬件架构

建筑结构健康监测系统的物理架构采用的是传统的物联网三层架构进行设计，如图4.2所示的系统整体架构图，分布于建筑各个关键监测点的各类建筑专用传感器如沉降、倾斜、风力风向、温湿度、挠度、震动加速度、裂缝计等，通过多传感器集成模块以ZigBee短距离无线传感网络进行组网，因建筑距离和跨度较大，在传输部分采用的是三级布网方案，一级网络为ZigBee网，二级网络为长距离WiFi布网，三级网络通过网关与Internet或者光纤进行三级组网，使最终的数据汇聚到中心机房的建筑结构健康监测服务器端，在应用层设计了多种应用开发，其中包括基于B/S架构的web应用程序、基于C/S架构的WinForm桌面应用程序、基于安卓系统的手持平板电脑专用软件、短信报警模块等。

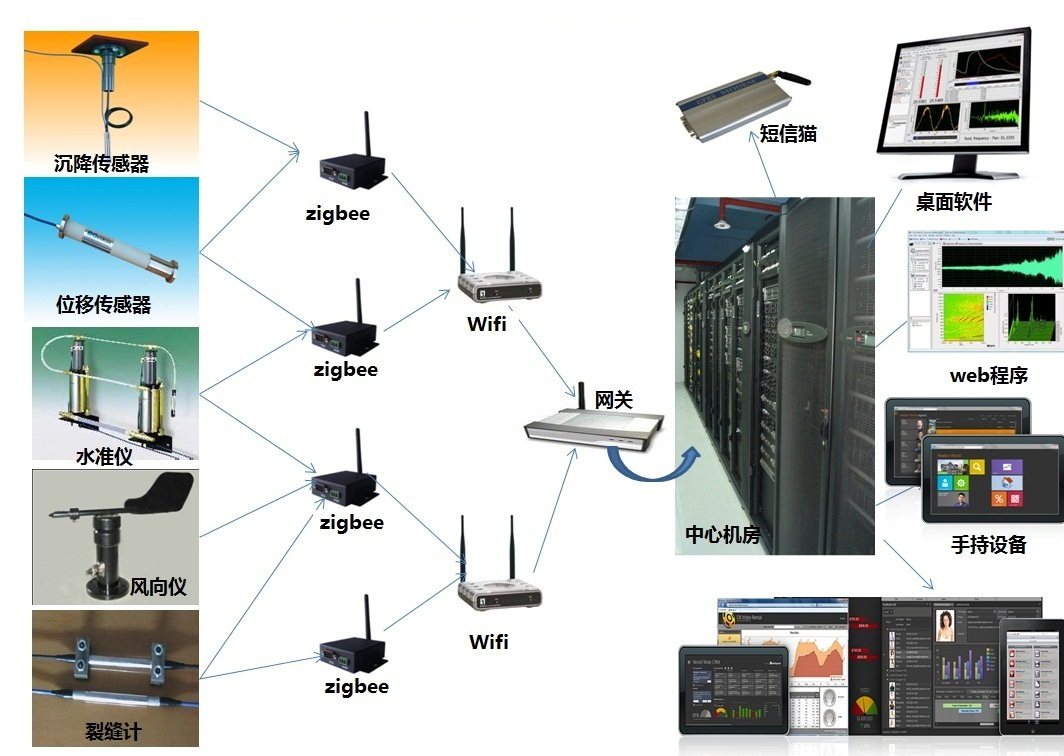


图4.2 建筑结构健康监测系统总体架构

### 4.4.2 软件架构

建筑结构健康监测系统服务器软件主要由初始化模块、监听模块、解析模块、数据库操作模块、系统配置模块、命令处理模块和系统日志这7个模块构成，如图4.3所示。

图4.3 服务器软件的模块结构图

初始化模块：完成服务器程序IP和端口的初始化以及全局静态变量的初始化。

监听模块：创建socket，循环监听端口，接收连接请求并获取UDP数据包。

解析模块：按照协议对UDP包进行解析，如果解析得到的是数据则将数据存入数

据库，如果解析到的为命令则提供相应的命令处理操作。

数据库操作模块：对数据库进行增、删、改、查等操作。

命令处理模块：完成对下载、全网对时、入网请求命令、节点查询命令的处理。

系统配置模块：解析Excel配置表，将Excel数据解析为hex记录下发实现对底层硬件工作模式的配置。

系统日志模块：完成异常处理的记录和错误数据的记录。

## 4.5 数据库设计

1. Mac地址与逻辑地址映射表 如表4.1

表4.1 Mac地址与逻辑地址映射表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名称** | **中文说明** | **键别** | **是否空** | **数据类型** | **长度** | **备注** |
| Id | 标识ID | PK | NO | INT |  | 自增 |
| Mac\_address | 物理地址 |  | NO | NVARCHAR | 20 |  |
| Logic\_address | 逻辑地址 |  | NO | NVARCHAR | 20 |  |
| Parent\_id | 父节点ID |  | NO | INT |  |  |
| Net\_in\_state | 在子网中的标号 |  | NO | INT |  |  |
| Version | 版本号 |  | NO | NVARCHAR | 20 |  |

1. 数据记录表 如表4.2

表4.2 数据记录表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名称** | **中文说明** | **键别** | **是否空** | **数据类型** | **长度** | **备注** |
| Id | 标识ID | PK | NO | INT |  | 自增 |
| repeat\_adress | 回复地址 |  | NO | VARCHAR | 10 |  |
| collect\_adress | 采集地址 |  | NO | VARCHAR | 10 |  |
| data\_style | 数据类型 |  | NO | NVARCHAR | 20 |  |
| data | 数据 |  | NO | VARCHAR | 40 |  |
| self\_time | 采集时间 |  | NO | NVARCHAR | 20 |  |
| write\_time | 入库时间 |  | NO | DATETIME |  |  |

1. 操作记录表 如表4.3

表4.3 操作记录表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名称** | **中文说明** | **键别** | **是否空** | **数据类型** | **长度** | **备注** |
| Id | 标识ID | PK | NO | INT |  | 自增 |
| logal\_adress | 逻辑地址 |  | NO | VARCHAR | 10 |  |
| operate\_type | 操作类型 |  | NO | VARCHAR | 10 |  |
| operate\_para | 操作对象 |  | NO | NVARCHAR | 20 |  |
| operate\_results | 操作结果 |  | NO | VARCHAR | 40 |  |
| time | 时间 |  | NO | NVARCHAR | 20 |  |

1. 节点信息配置表

表4.4 节点信息配置表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名称** | **中文说明** | **键别** | **是否空** | **数据类型** | **长度** | **备注** |
| Id | 标识ID | PK | NO | INT |  | 自增 |
| port\_number | 端口号 |  | NO | VARCHAR | 10 |  |
| serial\_number | 流水号 |  | NO | VARCHAR | 10 |  |
| config | 配置信息 |  | NO | VARCHAR | 60 |  |
| Bl | 测量值低值 |  | NO | VARCHAR | 5 |  |
| N | 采样次数 |  | NO | VARCHAR | 5 |  |
| TD1 | 通道类型 |  | NO | VARCHAR | 5 |  |
| TD2 | 端口类型 |  | NO | VARCHAR | 5 |  |
| Tadd | 端口地址 |  | NO | VARCHAR | 5 |  |
| N1 | 采样字数长 |  | NO | VARCHAR | 5 |  |
| N2 | 无效低位 |  | NO | VARCHAR | 5 |  |
| Xmin | 基本底数 |  | NO | VARCHAR | 5 |  |
| Z1 | 上报警数 |  | NO | VARCHAR | 5 |  |
| Z2 | 下报警数 |  | NO | VARCHAR | 5 |  |
| Y1 | 最小记差 |  | NO | VARCHAR | 5 |  |
| Y2 | 最小发差 |  | NO | VARCHAR | 5 |  |
| Tn1 | 正常采集频数 |  | NO | VARCHAR | 5 |  |
| Tn2 | 报警采集频数 |  | NO | VARCHAR | 5 |  |
| Nsg | 采样间隙数 |  | NO | VARCHAR | 5 |  |
| Time | 采集时间 |  | NO | DATE | 5 |  |

## 4.6 通信协议设计

4.6.1 整体说明

建筑结构健康监测系统的整体数据包流向如图4.3所示，底层485传感器采集到数据后，为数据加上485包头和包尾组成485报文，上传至zigbee模块。zigbee为485报文加上zigbee包头和包尾组成zigbee报文，上传至中继节点。中继节点获取485报文后，进行第三次封装，上传至3G模块。

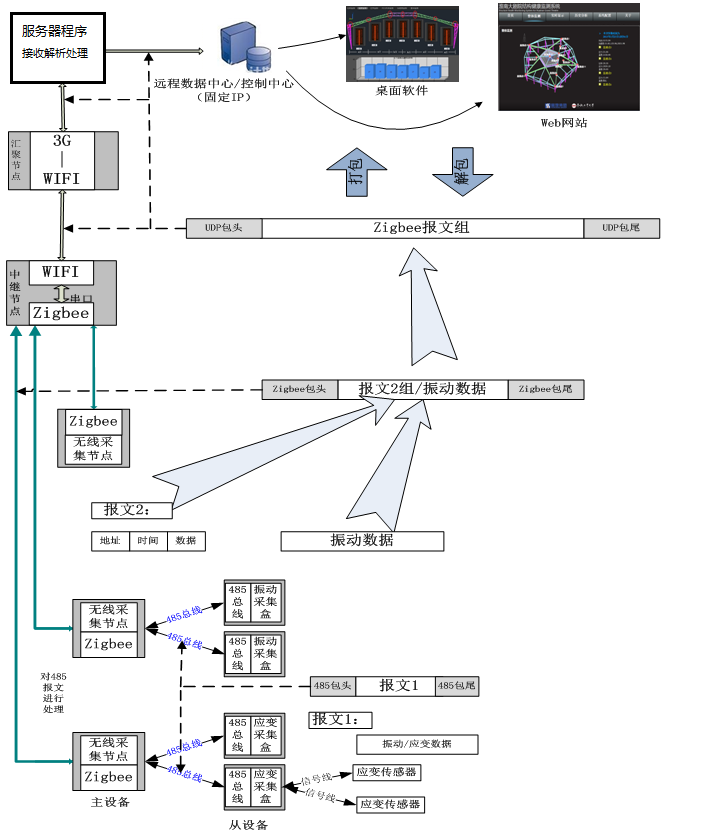


图4.4 数据包流向图

4.6.2 地址编码方案

表4.5 地址表示概述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 数据/控制中心 | 汇聚节点 | 中继节点 | 采集节点 | 485采集盒/通道 |
| 物理地址 | 设备ID | 路由器MAC | 固化CC2430 硬件ID | 固化CC2430 硬件ID | CC2430 硬件ID |
| 逻辑地址 | 固定IP | 汇聚节点编号 | 频段号 | 某个频段下节点号 | 通道号 |
| 物理地址大小（B） |  | 6B | 4B | 4B | 4B |
| 逻辑地址大小（B） | 1 | 1 | 4bit（低四位全0） | 4bit | 1 |

* 设备物理地址
* 汇聚节点：用路由器MAC地址标识，共48位，如：00 25 B3 46 68 C2
* 中继&采集节点&485采集盒：用四个字节标识其物理地址，全0全1暂且不用；高八位标识类型，共256种（现分别为中继、采集节点及485采集盒）；低24位标识每种类型节点的序号。
* 网络逻辑地址

表4.6 逻辑地址结构

23 15 11 7 0

|  |  |  |  |
| --- | --- | --- | --- |
| 汇聚节点ID | 中继节点ID | 采集节点ID | 通道号 |

* 详细说明
* 汇聚节点ID：用一个字节表示，表明属于哪个工程，采集数据上传时在中继节点中标识于数据包前端。
* 中继节点编码和采集节点编码组成节点的逻辑地址，由一个字节组成，中继节点编码有此字节的高四位表示，范围从0x10~0xE0进行编码，全网共有14个中继节点；采集节点编码有此字节的低四位表示，从0x01~0x0E进行编码，所以一个中继节点最多能够管辖14个采集节点。

4.6.3 通信包格式设计

表4.7 通信包格式设计表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | | | | | 有效载荷 | | | 包尾 |
| \*0 | 长度1 | 汇聚ID2 | 消息地址3 | ACK标志会话结束标志4 | 流水号5 | 消息类型6 | 消息参数列表7 | # |
| 1B | 1B | 1B | 1B | 1B | 1B | 1B |  | 1B |

* 长度：以字节数为单位，表明数据包的总长度。（例如:\* 5 a b #）
* 汇聚节点ID：用一个字节表示，表明是哪个工程。
* 消息地址：用一个表示。
* 上传请求时，该字段表示消息发起方源地址；上传采集数据时，该字段为中继ID。
* 若为下发控制命令时，该字段表示消息的目的地址（命令的施予方）。

注：若为通道命令时，该字段表示采集节点的逻辑地址，具体通道在作用通道中指出。

* 流水号、消息类型：皆用一个字节表示。

4.6.4 消息操作码编码方案

消息操作码用8位表示，高四位表示消息类型编码，全0全1暂且不用；低四位表示消息类内序号编码，全0全1暂且不用，故此消息操作码可编码14\*14种消息。

表4.8 消息格式详细设计表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 消息类别 | 消息名称 | 功能含义 | 操作码 | 参数列表 | 备注 |
| 上 传 数 据 类 [0x10] | **TCD** | Transport Normal Data：上传正常采集数据 | **0x11** | 缓冲长度（UINT8）；缓冲区采集数据 |  |
| **TAD** | Transport Alarm Data：上传报警采集数据 | **0x12** | 缓冲长度（UINT8）；缓冲数据包 |  |
| **TVD** | Transport Vibration Data：上传振动数据 | **0x13** | 长度（UINT8）；振动数据 |  |
| **TDD** | Transport Debug Data：上传调试数据 | **0x14** | 长度（UINT8）；调试数据 |  |
| **THD** | Transport Hardware Error Data：上传硬件故障数据 | **0x15** | 缓冲长度（UINT8）；缓冲数据包 |  |
| 请 求 类 [0x20] | **RUP** | Request Update Program：请求更新程序 | **0x21** |  |  |
| **RUC** | Request Update Configure Table：请求更新节点配置表 | **0x23** |  |  |
| **RNR** | Request Net Repeater：节点入网请求（中继和采集节点 | **0x24** | 节点物理ID（UINT32）； |  |
| **TL** | Test Link：发送探测链路握手请求 | **0x25** |  |  |
| **TR** | Timestamp Request：时间戳请求 | **0x26** | Originate Timestamp（UINT16）：起始请求时间；Receive Timestamp（UINT16）：接收方接收到的时间；Transmit Timestamp（UINT16）：TRY要回复的时间； |  |
| **RP** | Reply Program回复更新程序请求 | **0x31** | 程序包 |  |
| **RTT**  请  求  回  复  类  [0x30] | Reply Topology Table：回复中继节点配置表 | **0x32** | 回复此请求的时间（UINT16）；该中继节点的网络拓扑设置表见 |  |
| **RCT** | Reply Configure Table：回复采集节点配置表请求 | **0x33** | 配置表包 |  |
| **RLA** | Reply Logical Address：回复采集节点其逻辑地址 | **0x34** | 时间（UINT16）；采集节点逻辑地址（UINT8）；采集节点物理地址（UINT32） |  |
| **RLS** | Reply Link Success：返回链路连接成功 | **0x36** |  |  |
| **TRY** | Timestamp Reply：时间戳回复 时间戳消息有以下功能：时间同步、测网络延时、询问时间 | **0x37** | Originate Timestamp（UINT16）：起始请求时间；Receive Timestamp（UINT16）：接收方接收到的时间；Transmit Timestamp（UINT16）：TRY要回复的时间； |  |

5 详细设计

## 5.1 软件流程描述

建筑结构健康监测系统服务器软件流程如图5.1所示。程序启动后，首先完成初始化工作，对本地IP和Port进行赋值。接着启动监听线程，循环监听端口；当有连接请求时，接收请求并获取UDP数据包；接着会对UDP包做初步的合法性判断，不合法则丢弃，如果合法则启动解析线程进行解析，如果解析为数据则将数据存入数据库，如果为命令则移交命令处理线程进行处理。



图5.1 建筑结构健康监测系统服务器软件流程示意图

## 5.2 关键模块详细设计

### 5.2.1 监听模块

监听模块的流程如图5.2所示。监听模块是一个死循环，循环监听端口9000，等待连接请求。为了不阻塞主线程，所以创建了一个MainListener监听线程。MainListener启动时会创建一个套接字receiveSocket，并绑定本地端口。当收到连接请求时，接受请求获取UDP包，并创建一个DatagramPacket用于备份UDP包。然后根据协议对UDP包的合法性做初步判断（包头是否为‘\*’，包尾是否为‘#’），假如不合法则直接丢弃并且记入系统日志，假如合法则移交解析模块。



图5.2 监听模块流程示意图

### 5.2.2 数据解析模块

数据解析的流程如图5.3所示。解析模块也是单独创建了线程。首先获取监听模块传递过来的UDP包，根据定好的协议，UDP包的第一个字节为begin起始位，第二字节为length包长，第三字节为汇聚ID，第四个字节消息地址，第五个字节为ACK标志位，第六个字节为流水号，第七个字节是消息类型，第八个字节开始是消息参数部分，最后一个字节是结束位，图5.4所示。



图5.3 数据解析流程示意图

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | | | | | 有效载荷 | | | 包尾 |
| \*0 | 长度1 | 汇聚ID2 | 消息地址3 | ACK标志会话结束标志4 | 流水号号5 | 消息类型6 | 消息参数列表7 | # |
| 1B | 1B | 1B | 1B | 1B | 1B | 1B |  | 1B |

图5.4通信包格式设计图

根据第七个字节消息类型可以判定，此次接收到的是数据还是命令，假如是数据，则第八个字节开始直至结束位是数据部分，这部分是由n个长度为6字节的数据包构成，最高两个字节为数据地址，中间两个字节为采集时间，最底两个字节为采集数据，如图 5.5所示。



图5.5 采集数据包格式图

2个字节数据地址是由1个字节的通道号和1个字节的逻辑地址构成。解析模块会根据逻辑地址查询数据库的地址映射表，得到物理Mac地址。再根据Mac地址和通道号查询节点配置表如图5.6所示，获取通道类型，通道类型表明了这个包是什么类型的数据。最后根据具体的数据类型，将最低两字节的采集数据解析为常规格式。



图 5.6 采集节点配置表

### 5.2.3 命令处理模块

命令处理模块流程如图5.7所示。



图5.7命令处理模块流程示意图

假设收到的指令为中继入网请求，那么将节点配置表打包下发，告诉中继所属网络的拓扑结构；并等待接收来自中继的ACK确认回复，如果一段时间内无回复，则继续下发节点配置表。

假设收到的指令为全网校时，那么获取系统当前时间，转换成16进制后组包下发，并且等待下位机的ACK确认回复，如果一段时间内无回复，则继续下发并将计数器加1，如果计数器的值大于10，则认为网络超时，响应失败。

假设收到的是下载指令，首先区分是下载程序还是下载节点配置表。假设是下载节点配置表，那么读取用于存储节点信息的Excel表，并将Excel表中的数据解析成hex记录下发；假设是下载程序，那么获取用于存储嵌入式程序的hex文件进行组包下发。下发时记录每个包的流水号，当接受到下位机的ACK确认回复后，组包下发下一个数据包并将流水号加1，否则重新发当前的数据包。

假设收到的不是中继入网请求也不是校时命令也不是下载命令，则认为该命令非法不予处理。

## 5.3 难点和解决方案

5.3.1 并行处理

难点：在服务器程序运行的过程中会经常遇到这种情况：同时要与多个采集节点进行通信，有的是要接收采集节点上传的数据，有的是要向采集节点下发命令。那么如何对这种情况进行处理，并且保证处理结果的正确性？

解决方案：使用Java多线程的技术，首先创建监听线程，循环监听端口，等待连接请求。然后为每一个请求创建解析线程单独进行处理。

下面是解决该难点的关键代码：

while(true) {

//端口被关闭后也能开启

if(receiveSocket.isClosed()) {

try {

receiveSocket = new DatagramSocket(ConstValue.listenPort);

} catch (SocketException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

dataPacket = new DatagramPacket(buffer, bufferSize);

receiveSocket.receive(dataPacket);

//接收到的数据的格式定义，视图创建一个接收到的数据的副本

DataPacket myDataPacket = new DataPacket();

myDataPacket.dataPacketLength = buffer[1];

myDataPacket.projectID = buffer[2];

myDataPacket.messageAddr = buffer[3];

myDataPacket.ackAndDialogueflag = buffer[4];

myDataPacket.serialNumber = buffer[5];

myDataPacket.messageType = buffer[6];

//构建数据副本

int j = 0;

for(int i = 7; i < buffer.length-1; ++i) {/

j = i - 7;

myDataPacket.messageParam[j] = buffer[i];

}

//开启数据解析线程

PraseDataPacketRunnable praseDataPacketRunnable =

New PraseDataPacketRunnable(myDataPacket, receiveSocket, dataPacket );

Thread praseThread = new Thread(praseDataPacketRunnable); praseThread.setPriority(ConstValue.PraseDataPacketPriority); praseThread.start();}

5.3.2 下载配置表

难点：配置表是用于存储采集节点信息的若干张Excel表如图5.6所示，下载配置表就是要将这若干张Excel表中的节点信息解析成16进制的hex记录下发。首先如何提取这若干张Excel表中的节点信息，这是一个难点。此外，用什么数据结构对这些节点信息进行存储这又是一个难点。

解决方案：节点配置表主要由符号，含义，通道号这三部分构成。对于需要被提取的节点信息，在含义内容的末尾+‘\*’用于标识。所有的节点配置表按顺序存放在配置文件夹中，下载程序会依次进行读取。对于每一张Excel表，下载程序首先逐行读取含义这部分的内容，当找到带有‘\*’的含义时，将其对应通道号下的数值转换成16进制后存入byte[]数组中，当含义部分读取完毕后返回byte[]数组并插入ArrayList中，接着读取下一张Excel表直至遍历结束。

下面是解决该难点的关键代码：

public boolean downloadConfig(int index, String fileName) {

boolean downloadState = false;

System.out.println("开始下发配置表");

// 查询数据库获取MAC地址

this.MacAddrStr = getNodeMacAddrByLogicalAddr(MainProgram.requestDownloadList

.get(indexInDownloadList).logicalAddr);

DealDatatoCorrect dealDatatoCorrect = new DealDatatoCorrect();

// 构建要下发的配置表包

ArrayList<byte[]> confBytes = null;

try {

confBytes = dealDatatoCorrect.getDownloadConfigure(MacAddrStr,

fileName);

} catch (FileNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

System.out.println("config\_size:" + confBytes.size());

// 构建Map

for (int i = 0; i < confBytes.size(); ++i) {

// i为7、8的位置是包的序列号

int serialNum = DealString.twoHexStringToInteger(

confBytes.get(i)[7], confBytes.get(i)[8]);

indexSerialNumConfigHashMap.put(i, serialNum);}}

5.3.3 解析采集时间

难点：根据协议，采集时间只用两个字节表示如图5.5所示，最大为0xffff，转换成十进制为65535，而一天有3600\*24=86400秒。可见用两个字节无法表示全天时间。这就得区分采集时间是AM还是PM，假如是PM，还得区分是当天的PM还是前一天的PM。

解决方案：首先获取本地系统时间localHour（是24小时制的），然后根据系统时间localHour对采集时间hour进行判断，假设localHour>=12，并且localHour>hour+12那么采集时间=hour+12，是当天的AM；假设localHour<12并且localHour<hour,那么采集时间=hour+12,是前一天的PM；假设localHour<12并且localHour>hour,那么采集时间=hour,是当天的PM。

下面是解决该难点的关键代码：

int originalSecond = Integer.parseInt(originalSecondHexStr,16);

dataUp.self\_time = originalSecond+"";

int hour = originalSecond / 3600;

int min = (originalSecond - hour\*3600) / 60;

int sec = (originalSecond - hour\*3600 - min \*60);

int year = calendar.get(Calendar.YEAR);

int month = calendar.get(Calendar.MONTH)+1;

int date = calendar.get(Calendar.DAY\_OF\_MONTH);

int localHour = calendar.get(Calendar.HOUR\_OF\_DAY);

int localMin = calendar.get(Calendar.MINUTE);

int localSecond = calendar.get(Calendar.SECOND);

if(localHour>=12){

dataUp.half\_day = true;

if(localHour < (hour+12)) {

} else if(localHour >= (hour+12)){

hour += 12;

}

dataUp.date = year+"-"+month+"-"+date+" "+hour+":"+min+":"+sec;

} else {

dataUp.half\_day = false;

if(localHour < hour) {

hour += 12;

date--;

if(date == 0) {

month--;

if(month == 0) {

month = 12;

year--;

}

date = getLastDateInMonth(month, year);

}

}else if(localHour >= hour) {

}

dataUp.date = year+"-"+month+"-"+date+" "+hour+":"+min+":"+sec;

}

5.3.4 解析采集数据

难点：根据协议，采集数据是被封装在6个字节长的数据包中，如图4.5，最高两字节为地址（逻辑地址+通道号），中间两个字节为采集时间，最后两字节为采集数据。由于数据类型的不同，解析数据的方法也不同，这是解析时的难点。此外，如何获取数据类型这是另一个难点，数据类型是跟采集模块和该模块下的通道号有关，而服务器程序收到的只是通道号，不知道该通道号属于物理上的哪个采集模块，所以无法直接判断数据类型。

解决方案：在数据库中建立了一张Mac地址和逻辑地址的映射表，有了这张表后，可以根据数据包中的逻辑地址查询数据库得到物理Mac地址，也就知道了是物理上的哪一个采集模块。要知道数据类型还需要知道数据是从该采集模块的哪个通道上传的。为此，我在本地建立了一张用于记录通道号与数据类型对应关系的Excel表，Excel表就以物理Mac地址命名。当获取到Mac地址后，就去查询“Mac”.xls获取数据类型，根据数据类型得到转换公式进行解析处理。

下面是解决该难点的关键代码：

String MacAddrStr = getNodeMacAddrByLogicalAddr(logicalAddrStr);

//将16进制的mac中的小写变为大写

String MacAddrStrUpperCase = MacAddrStr.toUpperCase();

System.out.println("mac地址："+MacAddrStrUpperCase);

//获取通道类型码

int namePort = 0;

try {

namePort = getNamePort(MacAddrStrUpperCase, channelNum, MainProgram.noteConfigTableDirName);

} catch(FileNotFoundException fileEx) {

System.out.println("配置表文件未找到");

fileEx.printStackTrace();

break;

} catch(Exception ex) {

namePort = -1;

System.out.println("通道号不合法！");

ex.printStackTrace();

break;

}

System.out.println("通道类型："+MainProgram.NamePortHashMap.get(namePort));

dataUp.data\_style = (String)MainProgram.NamePortHashMap.get(namePort);

//获取数据部分的长度

subPacketLength = (Integer) MainProgram.NamePortSubPacketLengthHashMap.get(namePort);

//需要解析的数据

byte[] data = new byte[subPacketLength];

for(int j = 0; j < subPacketLength; ++j) {

data[j] = dataPacket.messageParam[i];

++i;

}

if(subPacketLength == 2) {

double value = 0.0;

try {

dataUp.isZdData = false;

value = dealDatatoCorrect.getParseData(data, channelNum);

///////////////////////////输出还没有解析的16进制数////////////////////////////////

System.out.println("还没有解析的16进制数据：");

for(int k = 0; k < data.length; ++k) {

System.out.print(DealString.dealHexString(Integer.toHexString(data[k]))+ " ");

}

System.out.println();

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

System.out.println("检测值："+value);

dataUp.data = value+"";

//存入数据库

updateDataInDB(dataUp);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

} else if(subPacketLength == 10){

//振动数据

dataUp.isZdData = true;

ZdData zddata = null;

try {

zddata = dealDatatoCorrect.getParseDataZd(data, channelNum);

System.out.println("振动数据：");

System.out.println("x轴方向加速度:"+zddata.x);

System.out.println("y轴方向加速度:"+zddata.y);

System.out.println("z轴方向加速度:"+zddata.z);

System.out.println("振动频率:"+zddata.f);

System.out.println("振幅"+zddata.a);

dataUp.x = zddata.x;

dataUp.y = zddata.y;

dataUp.z = zddata.z;

dataUp.f = zddata.f;

dataUp.a = zddata.a;

//存入数据库

updateDataInDB(dataUp);

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

6 系统测试

本次毕业设计主要运用黑盒测试技术，对服务器软件进行了功能测试。为方便测试，还用c#编写了一个可视化的测试工具如图6.1所示。



图6.1测试软件

## 6.1 命令解析测试

完成对入网请求命令、全网对时命令、通道重启命令解析正确性的测试。

6.1.1 入网请求测试

测试用例：发送入网请求RNR 2A 09 00 00 00002400 23

测试结果：服务器程序收到请求并回复RTT，如图6.2所示。

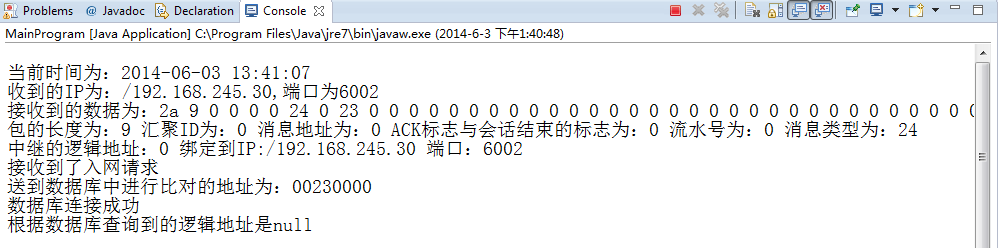
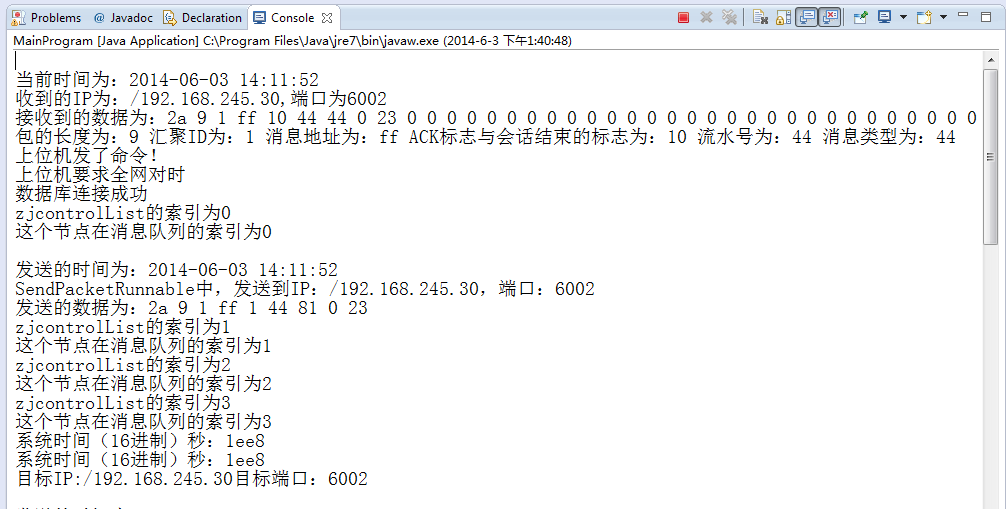


图6.2服务器程序回复RTT

6.1.2 全网对时测试

测试用例：发送全网校时命令TSO 2A 0A 01 10 00 00 44 24 25 23

测试结果：服务器程序收到命令下发16进制时间，如图6.3所示。

图6.3服务器程序解析对时命令结果

6.1.3 通道重启测试

测试用例：发送通道重启命令 2A 0A 01 ff 10 72 72 01 70 23

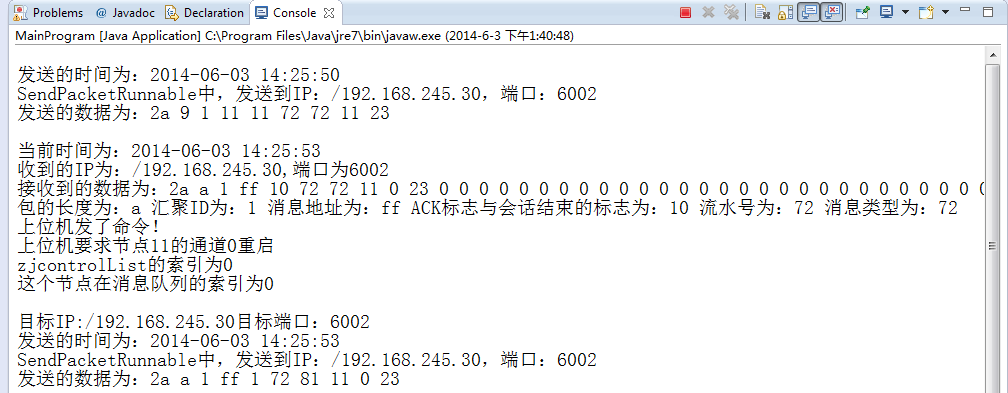
测试结果：服务器接收到重启命令并进行封装下发到相应通道如图6.4所示。

图6.4服务器通道重启命令解析结果

## 6.2 数据解析测试

测试用例：上传充电电压数据2A 09 01 10 0000 11 00 23

测试结果：服务器程序接收到通信包后进行数据解析，并将解析后的数据存入数据库如图6.5所示。

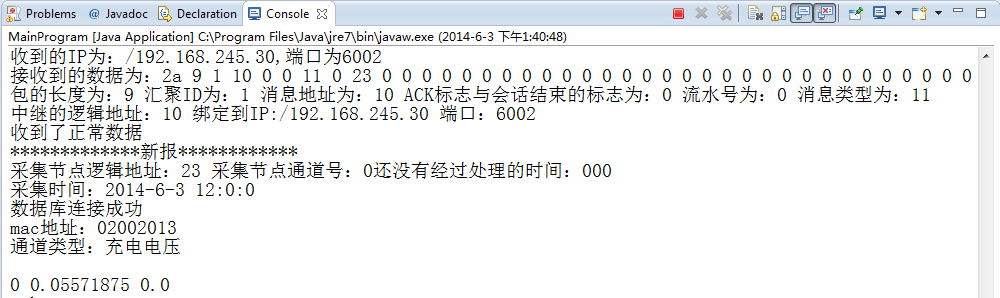


图6.5服务器数据解析结果

## 6.3 压力测试

完成对服务器程序的并发处理能力进行测试。

测试用例：多个中继节点同时向服务器发请求如图6.6所示。

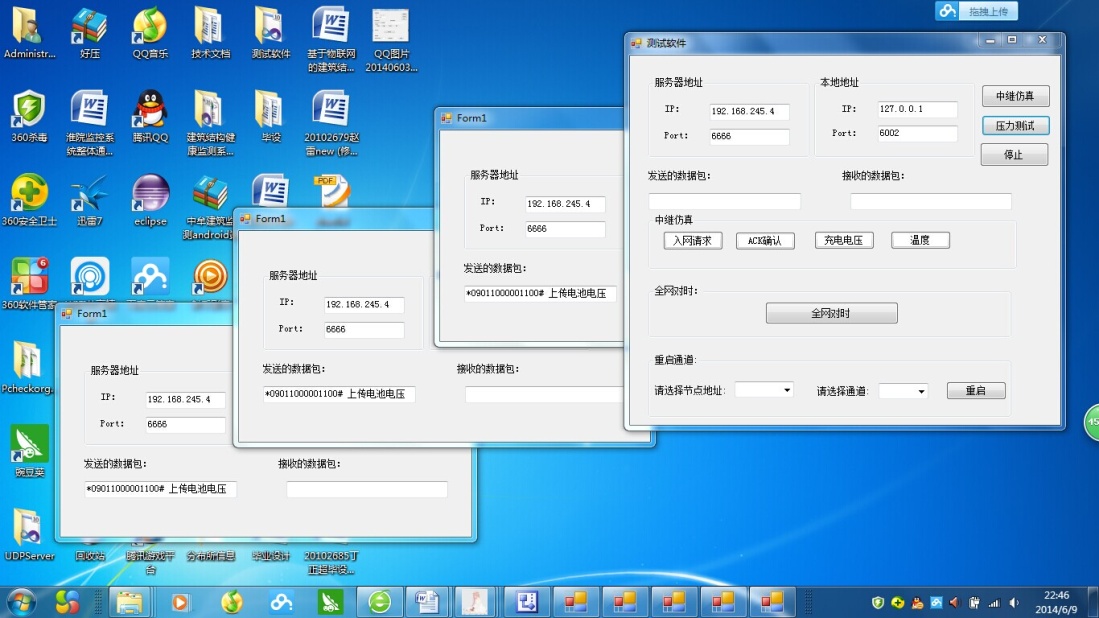


图6.6多个中继同时上传数据

测试结果：服务器程序通过多线程的技术分别处理如图6.7所示

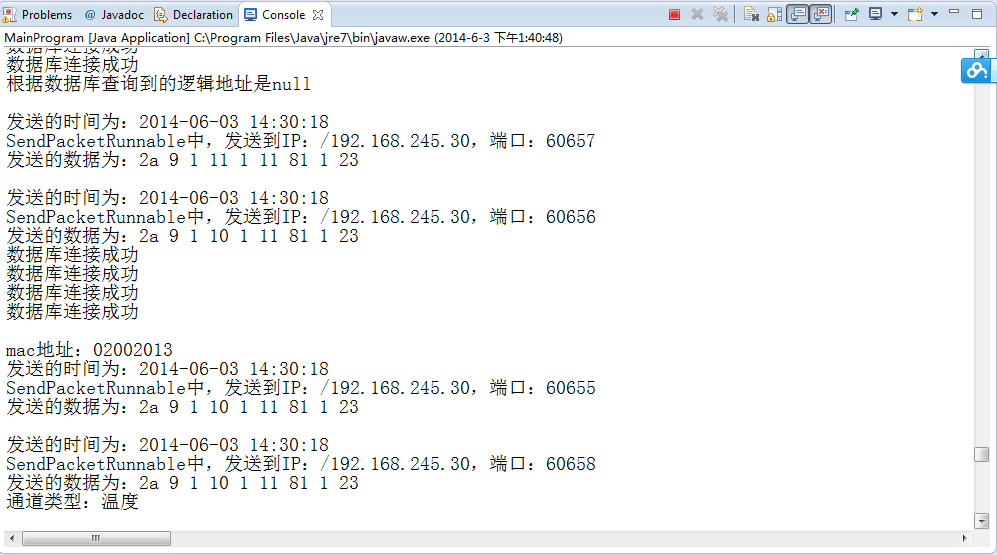


图6.7服务器压力测试结果

结 论

至此，基于物联网的建筑结构健康监测系统服务器软件已经实现。服务器软件能够实时获取底层传感器采集到的建筑结构数据，并将解析处理后的数据存储于数据库中。还可以解析用于存储节点信息的Excel配置表，将Excel数据解析成hex记录下对底层硬件工作模式进行配置。此外，服务器软件还能响应客户端软件的下载命令、校时命令、重启通道命令，根据具体命令提供相应的服务操作。当下载过程中发生断网异常时，服务器程序能够进行数据重传，从而保证数据的正确性。服务器软件的系统日志模块还能对错误数据和异常情况进行记录。

致 谢

四年的大学本科生活就要画上一个句号，回望四年的学旅生活，我的内心久久不能平静，四年的学习充满了辛酸和快乐，尽管和我们共同走过的老师教授我们知识的时间并不长，可是他们作为教师为我营造了一种良好的学习氛围，刚入大学的我，非常天真无知，四年的学习，使我不仅接受了全新的学习方式，学习到了专业知识，培养了专业技能，还使我确定了远大的人生目标。我感激曾经悉心教导过我的老师，帮助过我的同学，因为你们，我可以自信的走出工大的大学校园。

本毕业设计从选题、系统实现的过程和论文的书写都是在我的指导老师魏振春老师的悉心指导和帮助下完成的，在此我衷心地感谢魏振春老师。魏老师平易近人，在我们毕业设计期间给予了我们无微不至的关怀，每次我们遇到问题时，魏老师都会与我们进行讨论并引导我们去自己提出问题解决问题，即使老师在外出差也不时通过网络或是手机的方式询问我们毕业设计的情况并给予帮助。

毕业在即，四年的同窗让我依依不舍，计算机科学与技术10-2班的所有同学，是因为你们的帮助和支持，我才能克服一个一个的困难和疑惑，直至毕业，在这里请接受我诚挚的谢意。最后，我还要感谢一直辛辛苦苦养育着我的父母，谢谢你们！

参考文献

1. 梅茨克尔.Java设计模式[M].北京：电子工业出版社, 2012.45-102.
2. 王华,张震宇.大并发高性能用户应用服务器程序设计[J].浙江科技学院学报,2006,18(4):268-273.
3. 王娟,杨娜,杨庆山等.适用于遗产建筑的结构健康监测系统[J].北京交通大学学报,2010,34(1):100-104.
4. 马建，黄河清等．物联网技术概论[M]．北京：机械工业出版社，2011.56-98.
5. 陈国君.Java2程序设计基础[M].北京：清华大学出版社，2006.74-152.
6. 彭力.无线传感器网络技术[M].北京：冶金工业出版社，2011.102-145.
7. 杨刚，沈沛意，郑春红等．物联网理论与技术[M]．北京：科学出版社，2010.25-67.
8. 杜晓通.无线传感器网络技术与工程应用[M].北京:机械工业出版社，2010.64-90.
9. Eric. Java编程思想[M].北京：机械工业出版社，2011.202-254.
10. 王珊，陈红.数据库系统原理教程[M].北京:清华大学出版社，2004.13-156.
11. 耿祥义，张跃平.JAVA2实用教程（第二版）[M].北京:清华大学出版社，2004.78-124.
12. Joshua Bloch. Effective Java[M].Piscataway N.J:IEEE Press, 2009.203-256.
13. Bruce Eckel. Thinking in Java[M].Upper Saddle River, New Jersey, USA: Prentice Hall, 2006.92-127.
14. 尹立宏.Visual C#应用编程150例[M].北京：电子工业出社,2009.74-102.
15. 朱冰.C#语言实用教程[M].北京：中国铁道出版社, 2005.56-98.