本科毕业设计规范格式

学号

年级

（黑体5号）



**本科毕业设计**

（1号宋体居中）

**智能物流监控系统的数据平台技术研究**

（2号黑体居中加粗，标题行间距为32磅）

|  |  |
| --- | --- |
| **专 业** |  |
| **姓 名** |  |
| **指导教师** |  |
| **评 阅 人** |  |

（宋体小3）

**ⅩⅩⅩⅩ年Ⅹ月**

**中国 南京**

英文扉页示例：

**BACHELOR'S DEGREE THESIS**

**OF HOHAI UNIVERSITY**

（Times New Roman 2号粗体居中）

**Writing the title of the paper in English here**

（Times New Roman 2号粗体居中）

College ：XXX XXX

Subject ：XXX XXX

Name ： X X X

Directed by ： XXX Professor

（Times New Roman 4号居中）

NANJING CHINA

（Times New Roman小2号居中）

学术声明：

**郑 重 声 明**

（宋体粗体2号居中）

本人呈交的毕业设计，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计（论文）的研究成果不包含他人享有著作权的内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计（论文）的知识产权归属于培养单位。

（宋体4号）

本人签名： 日期：

**第1章 引言**

**1.1 项目背景**

物联网技术是利用传感器嵌入各种末端设备，通过无线或有线网络接入互联网以实现人与物、物与物实时互联的新兴技术。

智能物流监控系统是在物流量越来越大、物流价值越来越贵重的背景下为了保证运输过程中的安全性以及记录运输中的状态，借助互联网时代的大背景下产生的物联网典型应用之一。

为了适应当前时代的物流需求，我们需要一套可靠的贴合自身需求的物流监控系统，以便让物流的运输成为透明、可控的过程。

依托于物联网技术的智能物流监控系统，需要解决一系列问题：

1. 实时获取设备的各项数据
2. 实时数据的实时保存
3. 数据的结构与相互关联
4. 不同角色的权限控制
5. 数据的实时显示

市面上现有的物联网数据平台大部分已经具备支撑某一部分物联网系统的基础需求，且在某些方面有各自的特色，互联网企业例如百度、阿里、京东等目标着眼于智能，包括智能家居、智慧城市等，他们都有一套成熟的部署与服务方式，有自己的app，对于第三方的扩展应用与开发友好度不够。而传统硬件厂商则更多地着眼于硬件的操作与数据平台提供，对于第三方的开发具备更大的灵活性，因此第三方开发者将利用数据平台做出更加贴合自身需求的应用或者系统。

借助于现如今良好的物联网平台的生态环境支撑，本系统将作为物联网的第三方应用，吸收各大物联网厂商平台的资源，完成更加贴合自身在物流上的需求的系统设计。

本系统将专注于物流数据管理软件方面的设计与实现，对于实时获取设备的各项数据与实时数据的实时保存等于硬件交互的方面，已经有比较多的相对成熟的解决方案与产品。我们将选定某一产品作为系统的硬件支撑，系统设计的主要工作是在于设计数据结构与关联、控制不同角色的权限、实时显示物流数据。另外的，作为本系统的数据平台研究，主要任务将集中在数据结构与关联和控制角色权限，对于前端显示将交付于Android端于web端开发。

**1.2 国内物联网平台的产业生态**

针对自身在物流物联方面的需要，我们需要收集并对比国内各大物联网平台的优势与劣势，选择更贴合自身需求的物联网平台，借助平台对于第三方开发者的支持完成自己的物流监控平台。

目前的物联网产业市场基本上都已形成各自的一套系统解决方案，通常涵盖了物联网的基础功能，有些会根据各自平台的优势致力于特色功能。根据物联网平台解决不同层次的问题以及在不同层次对下层提供服务可以对物联网平台分为三种：

1. 基础设施类
2. 开发工具类
3. 运营服务类

第一类基础设施类，是以AzureIoT、阿里IoT、百度云等借助自身的云计算服务优势搭建的物联网平台。这一类平台提供最基层的环境与服务，支撑硬件数据提取与保存至云存储平台。而实际上这点基层服务无法运行起应用至用户端，所以，基本上这些平台也都有各自的针对物联网的PAAS工具，对第三方开发者提供数据接口，所以与用户对接的应用会有极大的灵活性。有些平台也会有各自的官方应用，针对自家的物联网平台需求，为了完善整个物联网系统生态。

第二类开发工具类，是以QQ物联、远景能源EnOS、GE Predix等为主的物联网平台。这些物联网平台要解决的核心问题是完善整个PaaS层，为了连接下层的IaaS与上层的SaaS，为物联网生态的末端设备与应用的交互提供开发工具。一方面面向硬件提供友好的环境与云计算平台，帮助硬件接入数据至云计算资源。另一方面向应用开发者提供可扩展且巧妙的存储结构以及友好的api，给开发者制定统一的开发标准，帮助物联网生态的最后一步交接至用户的使用上。而实际上，大部分这些物联网平台都不会放弃用户市场，为了抢占用户市场，这些物联网平台一般也都会研发各自的用户应用app作为官方应用，只是缺少了一些开发者自定义的功能，可以满足一些小成本物联网产品的需求。

第三类服务运营类,现有的大部分物联网平台都处于在服务运营类探索的阶段。平台众多，大家都想在物联网发展初期定义一套物联网时代的生活方式，这些生活方式正体现在用户与物联网的交互上面，所以正需要一整套物联网应用，完成用户生活的各方面渗透。只是现在正处于各界小厂商或个人第三方开发者依据PaaS层开发贴合自身需求的应用阶段，夹杂一些物联网平台已经完成的某个领域较完善的SaaS层的服务，譬如小米开放平台正在逐步构建智能家居领域的蓝图、飞凤平台对于智慧城市领域的一系列探索。正缺少对各个领域的整合，想要完善统一整个物联网应用生态需要一个实力足够的组织完成方方面面的应用开发以及应用与自家物联网平台的连接，或者完善一个类似应用商店，对接至自家物联网平台，统一各方应用以供用户使用，但是由于涉及到生活的方方面面，各种复杂场景正待解决，物联网平台仍然处于是一个急需发展以满足需求、构建万物互联的重要阶段。

我们本次设计的物流监控数据平台正是依据第二类开发工具类物联网平台，借助PaaS层连接IaaS与SaaS,完成自身需求的开发，也就是在第三类介绍中处于个人第三方开发者依据PaaS层开发应用。物流监控数据平台是物联网的其中一个典型应用，为了完成整个世界与物联网万物互联的生态，这还只是冰山一角。

我们需要找到一个可以更好的帮助我们构建应用，解决硬件方与软件方交互的PaaS平台。在对比了多家PaaS平台之后，权衡了各家平台生态与我们需求的契合度之后选定了中移动的OneNet平台作为本次系统的PaaS层支撑。

**1.3 OneNET平台介绍**

1.3.1 **OneNET平台概述**

OneNET平台是中国移动物联网有限公司为了解决物联网生态中硬件与软件连接、原始数据处理、简单的应用等通用且消耗沟通成本的问题而自主研发的开放云平台。

OneNET面向社会公共服务，着眼于开源的开发者环境，以开放态度以及互利共赢的理念，为各领域跨平台的物联网用户应用以及含物联网需求的各个行业的解决方案提供了更方便的连接与更稳定且灵活的云计算存储资源。帮助物联网开发者（包含企业组织与个人）降低运维和沟通成本、更专注于自身系统与应用的开发，依托于OneNET平台，共同构建统一核心的物联网生态环境。

OneNET平台主要提供了PaaS层一套完整且成熟的技术支持服务，为第三方开发者提供开发工具与帮助解决末端设备连接问题。实际上最重要的，是解决了互联网软件工程师与硬件工程师之间的适配问题。原本要开发一套物联网系统，需要双方约定好协议与数据适配，且需要服务端完成后硬件才能投入测试，而此时双方只需要都按照OneNET约定的标准进行开发即可，可同步开发流程。硬件工程师的开发进度并不影响软件方的进度，从最开始互联网软件工程师就可以按部就班的借助OneNET的数据平台搭建自己的物联网能力服务器（当然也是可以没有自己的物联网能力服务器的，原始的OneNET数据平台与API支持可以满足基础功能开发）。

1.3.2 **OneNET平台能力与架构**

OneNET平台专注于PaaS层，为IaaS层与SaaS层提供连接的桥梁，分别向上层需求与下层需求提供中间层的整个系统核心能力。

OneNET目前平台提供的产品功能包含

1. 流分析：开发者自定义设备数据流类型和数据模板，让上传数据可视化展示
2. 设备云端管理：实时监控管理接入设备的状态与运行情况，并对设备进行远程操作
3. 多协议适配：支持多种网络接入协议，轻松接入各种物联网设备、智能家居、汽车、穿戴设备、行业终端等
4. 轻应用快速生成：提供最基本的通用应用模块的随意组合
5. API：开放、完善的API接口便于用户在SaaS层与PaaS层对接

架构示意图如下：

1.3.3 **OneNET平台产品案例**

OneNET平台自身并不参与SaaS层的各个行业的细化应用，将对于SaaS层的开发支持交付于第三方开发者（组织或个人），构建了友好的应用构建环境，这将成为各行业应用百花齐放的沃土。

同时OneNET平台也提供了轻应用快速生成的服务，可以将一些通用的应用模块随意组合，更简化了产品开发的流程，可以满足一些小应用的通用需求。

借助OneNET平台对于SaaS层应用开发的友好服务支撑，产生了各个领域对于物联网系统的比较成熟的用户服务应用，完成了用户与生活很多领域的对接工作。其中包括智慧停车解决方案、共享经济解决方案、城市消防监测解决方案、畜牧物联网解决方案、车联网管理平台解决方案等等领域的成果，已经渗透至用户生活的很多方面。

以车联网管理平台解决方案为例，可以很好地展现物联网企业借助OneNET平台完成定制某一特定领域的工作与设计流程，以及OneNET平台在此类产品中占据的地位与整个系统中发挥的作用。

车联网管理平台使用DTU获取车辆不同部件传感器数据与GPS信息，联网将获取到的数据上传至OneNET平台，OneNET平台以合理的结构将数据存储在云资源，车联网管理系统Web端以及app端会通过使用OnenET平台提供的API接口获取或者操作数据，并最终进行通知、控制、展示等，完成车辆各方面状态以及位置的远程监管的功能点。

已有的相对成熟的产品案例的系统架构与工作流程将对本系统有一个很好地指引作用。后续的物流监控系统将吸取前人成功的经验，参考既有的使用OneNET平台的系统架构，加以改进并融合自身需求，完成物流监控领域的解决方案。

1.3.4 **选择OneNET平台的理由**

**1.4 论文的主要工作和组织结构**

**第2章 springboot 技术概述**

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。很多框架的开发者与使用者都奉行“约定大于配置”，然而Spring的使用者想要构建一套完整的开发环境与正常运行起最基本的应用，都要在一些很基础的配置上面浪费时间，而这些配置往往会每一个项目的初始都要被重新配置一遍，哪怕是完全一样的配置。Springboot正是帮助解决了每次项目初始的约定配置，对于系统类库的设定会给出假定合理的默认值。该框架使用对于既往开发过程的总结后既定的默认值来进行配置，从而使开发人员不再需要定义样板化的配置。其实spring boot并不是什么新的框架，它默认配置了很多框架的使用方式，它通常被认为是Spring MVC的“接班人”。

以下将对于项目作为Spring应用，介绍Spring框架（包括工作方式以及发挥的作用）与项目中依赖与引入的各项技术。

**2.1 Spring框架**

Spring框架是一个JAVA框架，它是2003年兴起的Java EE轻量级、开源框架，是Java平台为了简化Java EE项目开发复杂度而开发的。

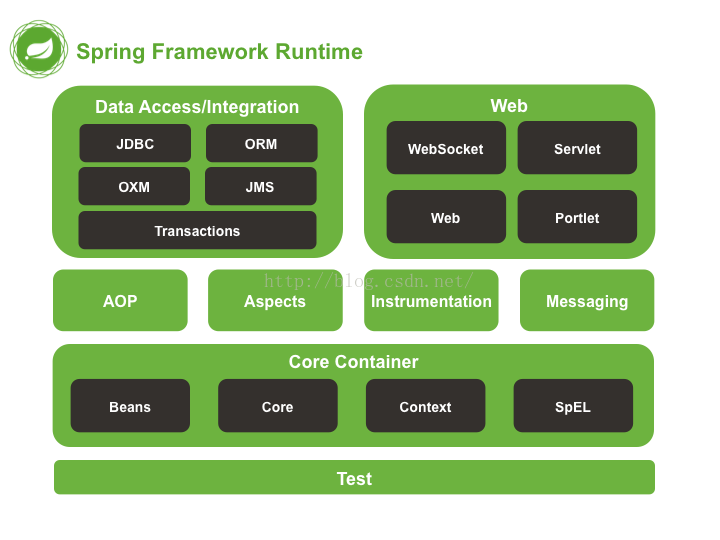
Spring框架对于解决企业应用开发复杂度具有很多方面的优势，其中之一的优势是它的分层架构，分层架构可以指示性的帮助开发者在系统开发的不同阶段为解决不同方面或模块的问题而使用不同的组件，同时整合了现有的一些优秀框架与常用技术，使现有的框架更加实用，为J2EE应用程序开发提供了集成之后的框架。Spring的目的并不是要取代那些现有的框架，而是要与它们无缝地整合。

依赖注入(Dependency Injection) 和控制反转(Inversion of control)是Spring的一个重要特性，它的基本概念是：不创建对象，但是描述创建它们的方式。对于代码中不应该出现直接与对象和服务交互连接的逻辑，只在特定的配置文件中描述某一组件需要某一项服务，以及连接服务的形式与参数。剩下的联系业务逻辑与服务的任务交由IoC容器来控制，通过那些开发的配置判断实例类型创建完成后依赖注入调用者，让开发人员专注于业务逻辑的实现，而不是浪费时间在重复的服务连接上面。

面向切面编程（Aspect Oriented Programming），即AOP，是一种编程技术，它倡导开发者对于某一特定方面问题或者分割其他依赖（例如拦截器和日志管理）的解决进行模块化。切面是AOP的核心构造，它将那些会对一部分类产生影响的行为封装起来，作为模块在需要的时候调用。

Spring被应用这么广泛的原因，是它具有紧密联系并巧妙配合的大约二十多个模块，这些模块被分组成Core Container，AOP (Aspect Oriented Programming), Instrumentation, Messaging, Test，Data Access/Integration, Web。

Spring各模块架构图如下：



核心容器：Core Container 由 Beans（spring-beans），Core（spring-core）,Context（ spring-context+ spring-context-support）, 和SpEL(Spring Expression Language) 模块组成。

Beans和Core作为框架的基础部分，提供了Spring的核心功能：依赖注入(Dependency Injection) 和控制反转(Inversion of control)。BeanFactory是工厂模式的一个优秀且复杂的实现，由于它的存在，对于可以编程的单例不再是必须的选项，并且帮助开发者将一些特定的配置（包括一些与其他服务的连接）与依赖从业务逻辑中脱离出来，真正实现解耦。

Context模块基于Beans与Core模块而建立，它是一种在框架类型下实现对象存储操作的手段。

SpEL模块提供了一个强大的Expression Language（表达式语言）用于运行时对于使用对象的各种行为（包括查询和一些修改的操作）。这种语言支持对属性值、属性参数、方法调用、数组内容存储、收集器和索引、逻辑和算数操作及命名变量，并且通过名称从 Spring 的控制反转容器中取回对象。

AOP 及 Instrumentation：AOP 模块提供依赖 AOP Alliance-compliant（联盟兼容）标准的面向切面的编程实现，支持开发者自定义某一模块的功能实现。比如，日志的记录打印和切入点完全分离业务逻辑方面的代码。

消息：

数据访问和集成：

Web：

Test：

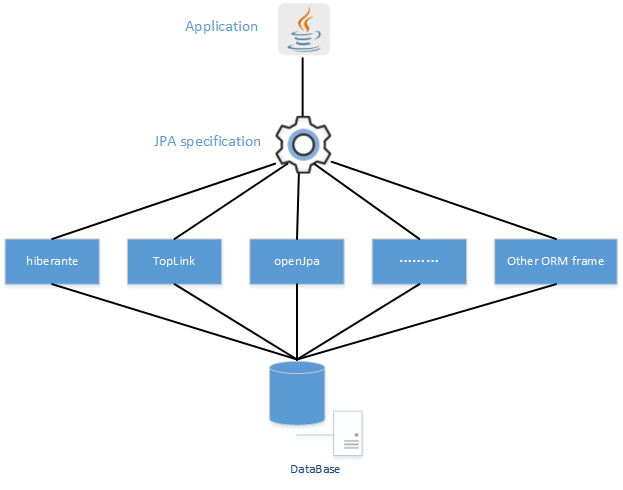
**2.2 JPA规范与在Spring中的应用**

提到JPA规范，需要先介绍对象关系映射（Object Relational Mapping，以下称ORM）。ORM是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单来说，ORM是一种使用元数据持久化数据的技术，对于数据库的数据用面向对象的代码中体现，使用对象描述元数据，建立程序与数据库之间的映射关系。由于开发者更偏向于面向对象的思考，所以将数据库领域映射至面向对象更贴合开发者的习惯。因此，可以用“one to many”描述表间关系，而不是用“row”、“column”、“foreign key”。

JPA规范是一种为统一所有遵守ORM模式的框架而为各框架编写者制定的行为规范。它是一种ORM规范，而不是ORM框架，用于指导ORM框架的实现标准。它制定规范，提供一些API接口，而不是去实现一些API接口，具体的实现则由各个包含ORM框架的服务厂商来完成（譬如JBoss应用服务器底层的JPA实现就是由hibernate完成的）。

这样一来，JPA规范定义并公布了所有需要有的接口，而且这些接口会被所有人（包括使用者与框架开发者）所熟知。对于使用者来说只需要根据文档组织自己的代码、调用自己需要的某些接口，这样很符合面向接口编程的思想，也呼应了对于分层分模块解耦的倡导。而对于框架开发者来说，不需要再自己重新设计一套框架的使用流程，直接按部就班一步步实现JPA规范规定好的一些接口（甚至接口名都不需要自己定义）。它帮助统一了ORM框架的开发与使用方式，如果没有它，对于市面上不同的ORM框架，比如hibernate与TopLink，在一个项目中使用hibernate时需要开发者学习使用hibernate框架的JPA实现，当另一个项目使用TopLink又要重新学习TopLink框架的JPA实现。

以下是JPA和不同种ORM框架之间的关系：



Spring中使用的Spring Data JPA 是为了在JPA的基础下，简化对于数据持久存储的访问层（包含查询与修改CURD）而努力。也就是说它在JPA提供的规范之下，提供了Repository层的实现，使得Domain classes的持久性方面的开发变得轻松了很多。

**2.3 RESTful API**

REST的英文全称是REpresentational State Transfer，中文意思是表现层状态转移。它在2000年首次出现在Roy Fielding（HTTP规范的主要编写者之一）的博士论文当中，他在论文中提到：“我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。REST指的是一组架构约束条件和原则”。而RESTful API就是符合REST约束条件和原则的API服务。

RESTful API遵循一些设计原则与规范：

1. 资源与URI。资源这个概念包含范围比较广泛，我们可以称所有出现在Web应用中可供用户查看与操作的“事物”为资源。资源可以体现在经过持久化的操作之后数据库某个数据表中的某一条数据或者被保存在磁盘的某个文件，也可以是Web服务器端应用接收到某一请求之后经过一系列代码运算操作之后得出的运算结果。资源可以被看做具体的物理方面的实体对象，也可以被看做抽象的流程。

一个资源需要有标记用于标识它，这个标记可以是唯一的也可以是多个。对于Web Service的API，我们很自然的选择URI作为资源的标识。我们认定，一个好的作为资源标识的URI应该具有良好的“可读性”。因为具有良好可读性的URI对于使用者可以对于要访问的内容一目了然。譬如以下这类URI：

[http://www.\*\*\*.com/order/{order\_id}](http://www.***.com/order/{order_id})（编号为order\_id的订单）

http://www.\*\*\*.com/user/{user\_name}（名字是user\_name的用户）

[http://www.\*\*\*.com/order/{order\_id}/device/{device\_id}（在编号为order\_id的订单下，设备编号为device\_id的设备）](http://www.***.com/order/{order_id}/device/{device_id}（在编号为order_id的订单下，设备编号为device_id的设备）)

1. 使用统一的接口与标准的HTTP方法。在面向资源的基础概念下，一个Web API是为了实现对某一特定资源的操作的服务。针对某一资源的操作，无非就是CURD操作，而这些CURD操作并不会因资源不同而存在不同。比较合理的体现应该是定义一致的接口来统一这些操作。而对于这些一致的CURD操作，我们在Web服务中又很容易可以把它们与HTTP请求建立联系（或者甚至可以说一一对应）。我们可以直接使用HTTP方法名（GET、POST、PUT、DELETE、PATCH等），用于组合Web API。这样可以使用URI与HTTP方法的配合，结果是URI尽情描述资源而不用管是何操作，由HTTP方法来描述当前API要完成的动作。
2. 无状态性。RESTful API应该像HTTP请求那样，作为无状态请求，它应该只关心资源会发生什么，而不需要关心请求的发出者的状态（即客户端状态）。每一次的请求都应该被视为一个新来的用户发出的，只需要针对本次请求做出相应的操作，不需要识别本次请求是否是第一次请求或者考虑用户后续访问的情况（本次不考虑Session技术，只针对Web API的设计）。

RESTful API是一个优秀的设计规范，它指导API设计者应该怎么设计出更规范合理的API。不是说在做项目时一定要依赖这个规范，只是如果我们引入并严格遵守了这个规范之后，可以帮助我们解决一些不必要的麻烦。至少在本次系统设计中，这个规范可以帮助我们前后端配合开发时更好的理解某一API接口在服务器做了哪些事。所以，本系统的设计将以RESTful API的规范为标准，设计出一套符合规范的API。

**2.4 Session技术与配置**

**第3章 智能物流监控系统需求分析与概要设计**

**3.1 智能物流监控系统整体概述**

智能物流监控系统的目的是为了解决在物流运输的工作中，被运输物体的实时状况无法被实时查看、运输过程无法重现的问题。它最终是为了服务于高端精密设备或物品的物流运输工作中，帮助保障运输物品的安全。

为了完成发配物流任务（便于监管物流的整个流程）和物流运输中对于物品的实时监控，我们对于智能物流监控系统的设计分为了两大部分：对于获取的系统基础数据平台的搭建，和监管物流流程用于接入物品监控的管理系统的设计。

**3.1.1系统基础数据平台架构**

**3.1.2 应用于物流的管理系统设计**

**3.2 智能物流监控系统需求分析**

**3.3 智能物流监控系统概要设计**