

杭州电子科技大学

硕 士 学 位 论 文

题 目： 螺栓预紧力实时监测系统设计

研 究 生 檀宇

专 业 电子与通信工程

指导教师 陈科明 副教授

完成日期 2021 年 6 月

杭州电子科技大学硕士学位论文

螺栓预紧力实时监测系统设计

研 究 生： 檀宇

指导教师： 陈科明 副教授

2021 年 6 月

**Dissertation Submitted to Hangzhou Dianzi University
for the Degree of Master**

Design of Real-time Monitoring System for Bolt Pre-tightening Force

Candidate: Tan Yu

Supervisor: Associate Prof. Chen Keming

June, 2021

杭州电子科技大学

学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：檀宇

日期：2021年6月2日

学位论文使用授权说明

本人完全了解杭州电子科技大学关于保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属杭州电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署单位名称仍然为杭州电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密论文在解密后遵守此规定）

论文作者签名：檀宇

日期：2021年6月2日

指导教师签名：陈利明

日期：2021年6月2日

摘要

随着使用螺栓联接的结构件越来越大,螺栓断裂后对生产带来的损失也越来越大。因此,对螺栓联接需要保证较高的要求。当前螺栓预紧力在线监测的方式正在逐渐替代传统的人工巡检方式,但仍然存在一定的不足:不能对螺栓松动程度进行精确预判;无效数据传输量大。

本文针对目前螺栓预紧力监测系统中存在以上的问题,设计了一套基于物联网技术的螺栓预紧力实时监测系统。通过自主设计的智能网关将螺栓预紧力监测设备的数据进行无线采集,通过 4G 将数据上报到云平台进行分析与处理。同时,设计 Web 客户端实现了设备状态的监测与远程控制、故障报警等可视化管理功能。

为了能够准确提前预判螺栓的松动时间,基于 BP 神经网络算法设计了故障预测方案。针对 BP 神经网络算法易陷入局部最优解的问题,提出以粒子群算法对其权值与阈值进行迭代优化。针对标准的粒子群算法存在的不足,设计随机的惯性权重与动态调整学习因子两种策略来优化。最后,通过测试表明改进后的 PSO-BP 神经网络算法预测精度的均方根误差相比于改进前降低了 47.9%,表明改进后的算法稳定性与预测精度更高。

为了减少远程运维系统中无效数据传输的问题,本文设计了分层式数据处理机制来优化数据传输链路。从设备层、平台层、应用层出发,根据每层间数据交互的特点设计相应的工作机制来优化数据传输。分层次设计了重复数据过滤、数据批量上报、对称监控点、断网续传等策略来降低无效数据的传输量。测试结果表明分层式数据处理机制相比于透传方式,数据传输量降低了 61%左右。

关键词: 螺栓预紧力、BP 神经网络、故障预测、数据传输

ABSTRACT

As the structural parts connected by bolts become larger and larger, the loss of production after the bolt breaks is also increasing. Therefore, higher requirements must be ensured for bolted connections. The traditional manual inspection method is gradually replaced by the current online monitoring of bolt pre-tightening force, but there are still certain shortcomings: the degree of bolt looseness cannot be accurately predicted; the amount of invalid data transmission is large.

Aiming at the above problems in the current bolt pre-tightening force monitoring system, this paper designs a set of real-time bolt pre-tightening force monitoring system based on the Internet of Things technology. Through the self-designed intelligent gateway, the data of the bolt pre-tightening force monitoring equipment is collected wirelessly, and the data is reported to the cloud platform for analysis and processing through 4G. At the same time, the equipment status monitoring, remote control, failure warning and some visual management functions can be realized by designing the Web client.

In order to accurately predict the bolt loosening time in advance, a fault prediction scheme is designed based on the BP neural network algorithm. Aiming at the problem that the BP neural network algorithm is easy to fall into the local optimal solution, a particle swarm algorithm is proposed to iteratively optimize its weights and thresholds. Aiming at the shortcomings of the standard particle swarm algorithm, two strategies are designed to optimize by designing random inertia weights and dynamically adjusting learning factors. Finally, the test shows that the root-mean-square error of the improved PSO-BP neural network algorithm prediction accuracy is reduced by 47.9% compared with that before the improvement, indicating that the improved algorithm has higher stability and prediction accuracy.

In order to reduce the problem of invalid data transmission in the remote operation and maintenance system, the layered data processing mechanism is designed to optimize data transmission links in this paper. Starting from the device layer, platform layer, and application layer, design corresponding working mechanisms to optimize data transmission according to the characteristics of data interaction between each layer. Strategies such as repeated data filtering, batch data reporting, symmetrical monitoring points, and resumable transmissions are designed to reduce the amount of invalid data transmission. The test result shows that compared with the transparent transmission method, the data transmission volume is reduced by about 61% through layered data

processing mechanism.

Keywords: Bolt pre-tightening force, BP neural network, Failure prediction, data transmission

目录

摘 要.....	I
ABSTRACT.....	II
第 1 章 绪论.....	1
1.1 研究背景和意义.....	1
1.2 国内外研究现状.....	2
1.2.1 螺栓预紧力监测技术研究现状.....	2
1.2.2 故障预警技术研究.....	2
1.3 本文主要研究内容及章节安排.....	3
第 2 章 螺栓预紧力实时监测系统设计.....	5
2.1 系统总体设计.....	5
2.1.1 需求分析.....	5
2.1.2 总体框架.....	5
2.2 硬件系统设计.....	6
2.2.1 智能网关软件设计.....	6
2.2.2 传感器.....	8
2.3 云平台设计.....	9
2.3.1 设备云.....	9
2.3.2 业务云.....	10
2.4 Web 端设计.....	14
2.4.1 技术架构设计.....	15
2.4.2 Web 功能设计.....	15
2.4.3 Web 功能实现.....	19
2.5 本章小结.....	25
第 3 章 基于改进的 BP 神经网络的故障预测方案设计.....	41
3.1 故障预测.....	41
3.2 BP 神经网络基本原理.....	41
3.3 PSO 算法及其改进策略.....	42
3.3.1 PSO 算法基本原理.....	42
3.3.2 PSO 算法改进策略.....	44
3.3.3 改进后算法性能测试和分析.....	46
3.4 基于 PSO-BP 神经网络的故障预测方案设计.....	47

3.4.1 改进的 PSO 算法优化 BP 神经网络流程	47
3.4.2 结合动态时间序列的预测方案设计	48
3.5 测试分析	50
3.5.1 测试初始参数设置	50
3.5.2 样本数据测试	50
3.5.3 实际数据测试	52
3.6 本章小结	53
第 4 章 分层式数据处理机制设计	26
4.1 分层式数据处理机制设计	26
4.1.1 分层式数据处理机制架构设计	26
4.1.2 分层式数据处理机制设备层设计	27
4.1.3 分层式数据处理机制平台层设计	31
4.1.4 分层式数据处理机制应用层设计	34
4.2 实验测试与结果分析	36
4.2.1 数据传输流量测试	36
4.2.2 固件远程更新测试	38
4.3 本章小结	40
第 5 章 总结与展望	54
5.1 本文工作总结	54
5.2 展望	55
参考文献	57
附录	60

第1章 绪论

1.1 研究背景和意义

近些年，我国基建工程的规模越来越大，在土木工程领域方面的建设尤其突出。在土木工程领域，钢结构因其抗震性强、强度高、高度工业化等优点而得到广泛应用^[1]。为了更好地满足工程建设的应用，一般会根据实际需求将钢结构材料加工后再使用。这种易于加工的特点使得钢结构在工程建筑中得到广泛使用，常用于海洋工程、工业厂房、大跨度桥梁工程和高层建筑物^[2]。在这种钢结构搭建的建筑中，需要使用螺栓进行部件联接。由于钢结构建筑的应用场景的不同，螺栓所受到的预紧力会受外界环境的影响会发生不同程度的变化，自然而然就会出现螺栓松动的情况。如果钢结构主要联接部位的螺栓过于松动的话就可能会导致设备损坏或人员伤亡。

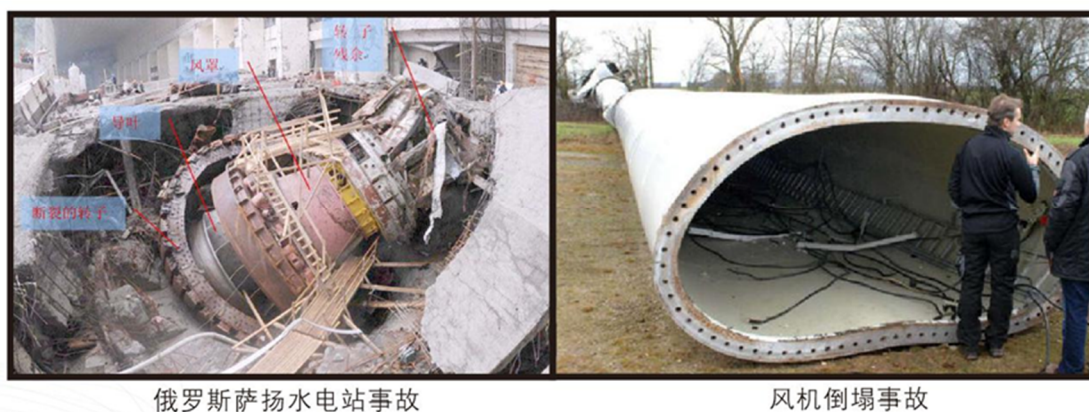


图 1.1 螺栓松动事故现场

2009 年 8 月 17 日，俄罗斯萨彦水电站发生重大事故^[3]，造成 75 人死亡，设备遭到严重受损，事故主要原因为一水轮机罩上的螺栓受到设备震动的影响而发生疲劳破坏，进而断裂后导致转子发生偏移运动。2020 年 7 月 17 日，河北某风力发电场的风机发生倒塌，经过现场勘察后发现事故的原因是该风机的两节塔筒之间的联接螺栓发生断裂导致的。

大型工程建设完成之后，需要定期检查结构的状态，提前发现问题以降低事故出现的概率。大型工程如果发生事故，造成损失是巨大的。使用螺栓联接的设备处于恶劣的环境中时，螺栓松动情况不好直接判断。如果在螺栓松动之后没有及时发现并且采取维护措施的话，将造成巨大的损失。为了避免螺栓松动而造成事故而引起的损失，实时监测螺栓的松动状况的重要性变得不言而喻。

目前市场上的螺栓预紧力监测系统的智能化程度不够，存在系统性能差、软件服务不完善、设备异常状态监测难的问题。因此，设计一种能够实时监测螺栓联接状况、系统功能完

善、设备异常状态可预测、用户交互体验好、便于维护的螺栓预紧力实时监测系统显得尤为重要。

1.2 国内外研究现状

1.2.1 螺栓预紧力监测技术研究现状

当设备处于振动的环境下时，设备上的螺栓会受其影响而慢慢变得松动。当螺栓松动到一定程度时，有可能会造成生产事故的发生。螺栓松动的传统监测方法是采用人工巡检，成本大且难以避免突发性事件的发生。通过螺栓预紧力在线监测系统，可准确判断螺栓是否发生松动或断裂，从而进行及时维护，避免重大事故的发生而导致的财产损失及人员受伤。

在国内外有不少学者根据不同的监测方式而设计了多种螺栓松动监测方法。在国内，文献^[4]利用干簧开关的特性对罐道内螺栓联接状态进行监测，判断螺栓是否松动。文献^[5]对 Lamb 波信号进行研究后，基于时间反演原理设计了螺栓松动监测的方法，并且研究了螺栓表面粗糙程度对监测结果准确性的影响。文献^[6]通过研究小波分析特征量与螺栓预紧力之间的变化关系，基于小波分析法对螺栓联接结构的状态进行监测。文献^[7]和文献^[8]利用压电阻阻抗技术，研究了压电阻抗谐振频率的变化量与结构受到外部压力变化量之间的线性关系，最终以压电阻抗谐振频率的变化量作为参考依据去监测螺栓的联接状态。文献^[9]通过分析法兰局部构件轴向荷载与位移的关系以及螺栓所受应力与轴向荷载的关系后建立起了螺栓松动监测模型，进而判断螺栓是否发生松动。文献^[10]利用多个传感器采集的数据，提取数据的特征信号输入到卷积神经网络中进行训练学习后再判断螺栓联接结构的紧固程度。文献^[11]利用螺母侧面的压电材料获取数据，再经过研究压电材料导纳峰值频率，建立了导纳峰值频率与预紧力之间的关系，实现了对螺栓预紧力的测量。文献^[12]首先基于“开合理论”提出了评定螺栓松紧程度的参数 L1，然后通过振动声调制技术完成了对松动螺栓的定位及松动程度的判别。文献^[13]首先建立了 CFRP 有限元分析模型与螺栓结合面连接刚度模型，通过这两个模型研究了预紧力与固有频率之间的影响关系，在此基础上结合 RBF 人工神经网络实现预紧力的量化与松动螺栓的定位。

在国外，文献^[14]通过对螺栓联接结构进行实验后采集动态应变与结构加速度信号，再分析这两个信号与螺栓松动的关系，实现对螺栓紧固程度的监测。文献^[15]通过研究超声波穿过螺栓时所形成的透射波的非线性特性实现了对螺栓联接状态的监测。

本文中，设计利用螺栓预紧力传感器监测螺栓的松动程度，通过其受到螺栓的预紧力大小值反映出螺栓从紧变松的过程。将监测值与设定的松动指标对比后判断螺栓是否过于松动。

1.2.2 故障预警技术研究

故障预测是在设备工作时持续检查并判断设备未来一段时间的运行情况，可通过相关传感器采集的历史数据作为判断依据，再配合相关故障预测技术进行设备状态预测。近些年，

国内外有不少针对故障预测技术的研究。随着设备种类及实际应用场景需求的不断变化，衍生了多种故障预测的技术。常见的有以下两种：基于模型与基于数据驱动^[16]。

（1）基于模型

基于模型的故障预警技术是在建立设备的物理模型与数学模型后找出状态变化的规律再由此规律进行设备状态的判定。此方法对技术人员的要求比较高，需要了解设备结构的每一部分，包括机械、电路等部分。由于其前期复杂的数学建模工作，使得其故障预测的精度较高。较为常用的有粒子滤波法及卡尔曼滤波法。由于建立模型是基于某种特定设备进行分析的，这就造成了此方法的扩展性不强，无法快捷地应用在另一种设备上实现故障预测。

（2）基于数据驱动

以设备的历史监测数据作为输入，以观测值作为输出，通过分析处理建立起输入与输出之间的映射关系后实现故障预测。相比于上一种故障预测方法，此方法不需要构建复杂的数学模型，在减少了工作量的同时应用性与扩展性也得到提高。近几年的物联网技术与云计算技术的迅速发展而带来的大量数据为此技术奠定了研究的基础，使得其应用得最为广泛^[17]。

此种方式下的常用的故障预测方法有：神经网络算法、卡尔曼滤波算法、支持向量回归、贝叶斯网络、灰色理论等。其中卡尔曼滤波算法的运算量比较大，预测精度受当量故障率的线性特征影响；支持向量回归的预测精度受其自身的选择何种核函数及其函数的参数影响；贝叶斯网络预测依赖于先验概率；灰色理论的预测精度不高，适合于指数式增速的样本数据；神经网络算法拟合能力好，非线性映射能力强，对于具有非线性特性的监测数据，基于神经网络算法进行预测的方案较为突出。文献^[18]提出利用阶次分析法处理振动信号后提取其功率谱故障特征信息，再结合 BP 神经网络算法实现了风机轴承故障识别的方法。文献^[19]提出基于 BP 神经网络设计矿井主通风机故障预警的方法，使用小波包变换的方法提取故障特征值，并利用遗传算法优化 BP 神经网络。

综合上述的分析并结合实际的项目特点，本文基于数据驱动设计一套螺栓预紧力实时监测系统。通过监测螺栓预紧力传感器上报的历史数据，通过优化后的 BP 神经网络算法实现了对螺栓预紧力监测系统的故障预警。

1.3 本文主要研究内容及章节安排

本文针对实际项目——螺栓预紧力实时监测项目进行系统设计。基于物联网技术设计了一套螺栓预紧力实时监测系统。系统具备设备远程监控、故障报警、数据可视化管理等多个功能。将整个系统拆分成多个模块，详细分析并设计各个模块。同时，着重对系统性能进行了两方面的优化：提出一种基于 PSO-BP 神经网络算法设计了设备故障预警方案，实现了设备发生故障时的实时报警与预警，有效保证了企业生产的安全；提出一种分层式数据处理机制，降低了无效数据的传输，减少了企业数据通信的成本。

第一章：绪论。首先介绍了螺栓预紧力监测系统的发展趋势，指出当前系统中存在的不足。

足，明确了论文的研究意义。接着从螺栓预紧力监测系统、故障预测两个方面阐述了国内外的研究状况，为本文的设计指出方向。

第二章：螺栓预紧力实时监测系统设计。本章首先分析指出系统设计需求与整体框架。然后，将系统划分成硬件、云平台、客户端后对各个模块进行单独地分析和功能设计，并展示了各个模块的相关实物或者应用。

第三章：基于改进的 BP 神经网络的故障预测方案设计。本章首先介绍 BP 神经网络的结构及其工作原理，并且指出目前 BP 神经网络算法存在的问题，提出以粒子群算法进行优化。介绍粒子群算法的原理及其工作步骤后，分析其优缺点。并且提出在迭代的过程中动态地调整相关参数进行算法优化。最后，利用测试验证改进后的 PSO-BP 神经网络算法能够提高故障预测的精准度。

第四章：分层式数据处理机制设计。首先分析指出当前的物联网远程运维项目中存在的一些不足之处。然后结合本项目的特点提出一种分层式数据处理机制，从设备层、平台层、应用层出发进行数据传输优化。最后对整个系统进行相关测试，验证优化效果。

第五章：总结与展望。总结本文所设计的系统及相应的优化点，并对本系统中存在的不足之处做出分析，指出之后需要研究和改进的点，为后续研究提供指导。

第2章 螺栓预紧力实时监测系统设计

为了满足企业的实际需求，克服目前工业系统中设备问题发现不及时的重点。本文设计了一套基于物联网技术的螺栓预紧力实时监测系统，涵盖硬件系统设计、云平台设计、客户端设计，实现设备远程监控与可视化管理。

2.1 系统总体设计

2.1.1 需求分析

根据构建螺栓预紧力实时监测系统的关键要素以及目前监测系统的不足，对于螺栓预紧力实时监测系统，总结出以下几点需求：

（1）实现设备入网，实时上报螺栓预紧力传感器采集的数据到云端，并且能对设备进行远程控制。

（2）数据采集设备支持多种入网方式，包括 4G、以太网、WIFI 等方式，以便达到多环境可选择性地部署设备的目的。

（3）开发配套的软件服务，提供 Web 网页端数据查询、分析功能以及设备控制功能。提供设备故障报警功能，当设备发生故障时在网页端弹窗提醒。同时能够以短信或微信通知的方式发送给相关人员。同时，考虑到相关人员的职责划分，设置为用户分配权限的功能，使得不同职责的人员对于不同的功能模块具有不同的控制权。

（4）设备故障预警，提前预判螺栓在未来一段时间内是否会松动，减少故障带来的损失。

（5）优化数据传输，减少远程运维中无效数据的传输量，降低企业生产成本。

（6）整个系统性能要保证稳定可靠，具有容错能力。如果发生系统宕机的严重事故时，需要设置相应机制保障系统自动恢复上线。

2.1.2 总体框架

采用物联网技术，设计了螺栓预紧力实时监测系统。整个系统可分成为三个关键部分：硬件系统、云平台、客户端。系统整体架构如图 2.1 所示。

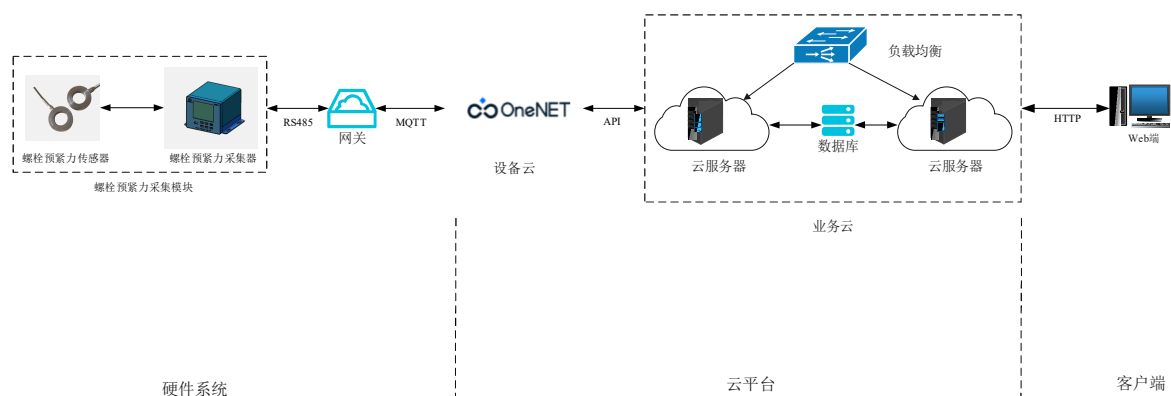


图 2.1 系统整体架构图

在上图 2.1 所示的结构中，硬件系统可分为三个部分，它们分别为：智能网关、螺栓预紧力传感器、螺栓预紧力采集器。其中，螺栓预紧力传感器与采集器二者共同组成了螺栓预紧力采集模块，用于采集与上报螺栓受到的预紧力给网关。网关为采集模块提供数据上云支持，其内部的网络模块为网关提供无线数据传输功能。云平台作为设备端与客户端连接的桥梁，主要用于处理业务逻辑以及保存相关数据。客户端设计为 Web 应用端，为用户提供数据展示与设备操控等功能。下面是对系统部分的详细描述。

硬件系统。智能网关通过 RS485 接口连接螺栓预紧力采集模块上，通过 Modbus 协议对设备进行数据采集。网关通过 4G 模块将数据上传到中国移动开发的 OneNET 云平台，实现设备的长连接功能。

云平台，即后端服务器。云平台由两个不同的部分组成，分为设备云与业务云两大模块。其中设备云采用的是中国移动开发的 OneNET 云平台，主要负责处理设备上传的数据及命令下发，提供设备数据存储。业务云由我们自己开发，根据具体的业务需求提供 API 服务接口给客户端进行调用。设备与云平台之间的数据交互只发生于设备云平台与设备之间，设备不与业务云平台发生直接地数据交互。

客户端，即 Web 端。客户端直接与用户关联，通过调用业务云提供的 API 接口完成具体的业务功能。Web 端为企业提供设备远程监控、故障实时报警与预警、数据管理等功能。

2.2 硬件系统设计

2.2.1 智能网关软件设计

网关向上与设备云平台相连，向下与工业设备相连。网关作为数据采集模块与云平台两者之间的通讯桥梁，在软件设计上需要实现以下功能：

- (1) 与设备云平台保持数据交互与长连接。
- (2) 支持监控点对应的设备数据的定时上报与查询。
- (3) 汇聚节点数据，分发控制指令给指定节点。

网关在系统中的工作流程如图 2.2 所示：

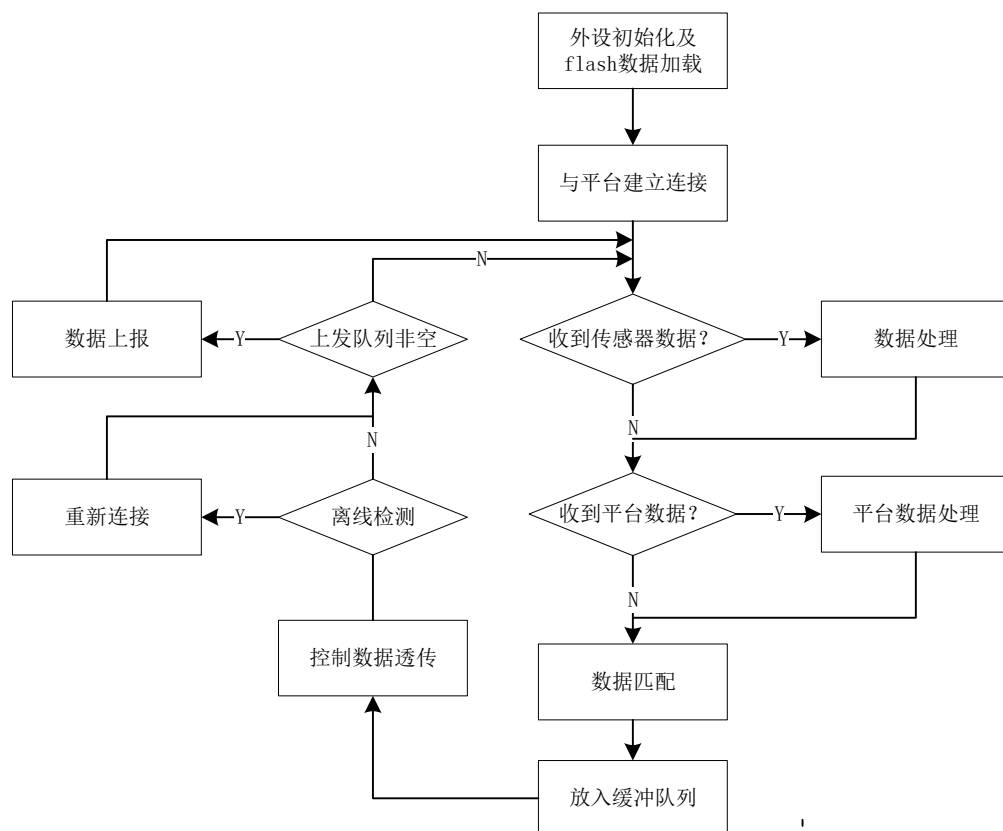


图 2.2 网关工作流程图

在传感器、设备与云平台进行交互的流程中，需要维护以下几个数据结构：

（1）数据流上发队列：传感器采集的数据存储在该队列中，等待发送给设备云平台。

（2）监控点列表：存储每个监控点的数据信息，提供掉电保存功能。设备云平台可通过下发命令帧进行数据点的增加、删除、修改、查询操作。智能网关定时向传感器发送查询命令，获取响应监控点数据。

（3）监控点缓冲队列：网关获取到传感器收集到的数据以后，将数据立即放入缓冲队列中，这些数据主要是由与网关相连接的设备上报的，这两者之间通过 RS485 或者 RS232 接口相连。

（4）数据重发队列：重发机制保证智能网关与设备、设备云平台数据发送不会丢失，根据规定好的数据帧协议中的 CRC 校验码进行确认，将待发送的数据加入到重发队列中，等到下个时间间隔后重新发送。

2.2.1.1 实物展示

智能网关主板及实物如下图 2.3 和图 2.4 所示所示。

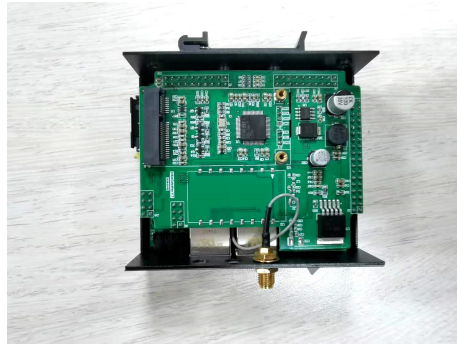


图 2.3 网关主板图



图 2.4 网关实物图

2.2.2 传感器

螺栓预紧力传感器设计用于监测螺栓联接的紧固程度。这种监测方式能够满足时刻监测螺栓预紧力的要求。利用相应的仪器可在安装和运行过程中的监测实际预紧力值，可以更好地调整螺栓松紧程度，使其处于一个正常的工作范围内。

螺栓预紧力传感器具有不同的系列，可适合多种规格下的螺栓预紧力监测需求。不同系列的螺栓预紧力传感器参数如下表 2.1 所示。

表 2.1 同系列的螺栓预紧力传感器参数

型号	内孔(mm) B	外径(mm) A	厚度(mm) D	重量(g)	最大负载(KN)
M20	20.2	37	14	73	166
M22	22.2	40	14	85	208
M24	24.2	44	14	104	239
M27	27.2	50	14	137	315
M30	30.3	56	17	214	385
M33	33.3	58	17	218	480
M36	36.3	66	17	295	560

通过监测预紧力传感器受到的预紧力值，从而判断螺栓是否松动。传感器实物与实地安装如下图 2.5 和图 2.6 所示。



图 2.5 螺栓预紧力传感器



图 2.6 传感器实地安装

螺栓预紧力传感器与螺栓预紧力采集器共同组成了螺栓预紧力采集模块。其中，螺栓预紧力传感器测量自身受到的螺栓预紧力后传输给采集器，而螺栓预紧力采集器将传感器采集的数据从模拟量转成数字量，再将数据上报到网关。

螺栓预紧力采集模块实物如下图 2.7 所示，设备安装调试如下图 2.8 所示。



图 2.7 螺栓预紧力采集模块



图 2.8 设备安装调试

2.3 云平台设计

云平台设计分成设备云与业务云两个部分进行开发，这两个平台发挥着不同的职能。其中，设备云负责与智能网关进行数据交互；业务云则主要负责与 Web 端进行数据交互。

2.3.1 设备云

在本文设计的螺栓预紧力实时监测系统中，设备云平台选择第三方公司开发的公有云平台。之所以这样选择的原因是因为第三方平台技术成熟、平台兼容性高，相较于自己部署设备云，使用第三方成本也更低。

利用中国移动开发的 OneNET 云平台^[20]作为设备云平台，实现设备通信、数据分析与处理、数据存储、设备监控、在线调试、事件报警、固件升级等功能，为螺栓预紧力在线监测提供技术保障。使用 OneNET 云平台的用户多，使得其经过了海量设备接入与多种类设备连接的考验，平台技术成熟，符合本系统的开发要求。开发者使用第三方平台后能够降低设备入网的技术壁垒，减少了开发的工作量，加快了系统研发的进度。

网关接入云平台之后，通过 4G/WIFI 模块以 MQTT 协议方式将设备采集的数据上报到 OneNET 云平台。同时，OneNET 云平台为客户端与业务云提供了 RESTful 规范的高可用接口，开发者可利用这些接口读取设备的相关数据以及远程控制设备。OneNET 云平台提供设备数据告警机制，当设备上报的数据达到设定的条件时可通过业务云提供的 API 接口将报警信息以短信或者邮件的方式推送给相关人员。设备云平台的管理页面如下图 2.9 所示，开发者可以根据实际需求进行设备接入监控、规则引擎设置、数据推送、触发器设置等操作。

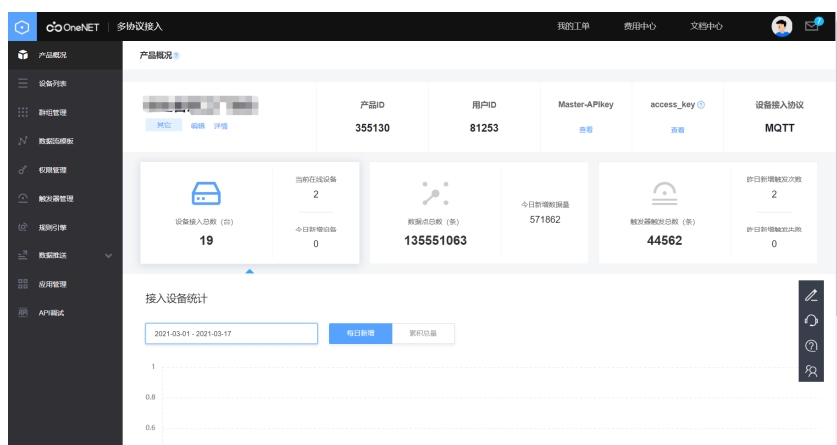


图 2.9 OneNET 云平台管理页面

2.3.2 业务云

在本文设计的螺栓预紧力实时监测系统中，选择在阿里云上采用云虚拟主机 ECS 来搭建业务云平台。

根据业务需求设计，业务云平台由负载均衡、云服务器和数据库组成一整套架构。其中，负载均衡负责将客户端的请求分发给多台云服务器，缓解单台服务器的压力，同时也能避免单点故障的问题。云服务器作为系统业务核心部分，提供业务逻辑处理功能，同时提供接口给其他应用软件接入服务。数据库提供数据持久化存储服务。本文根据实际的项目需求，使用一台服务器负责负载均衡转发器，两台服务器作为服务器集群节点。业务云整体架构如图 2.10 所示。

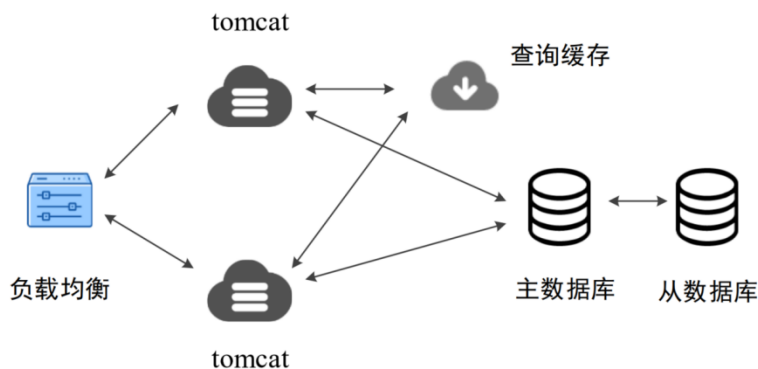


图 2.10 业务云整体架构

2.3.2.1 业务云技术架构

本文使用 Nginx 搭建高可靠负载均衡器。数据库采用阿里云上的 MySQL 数据库，同时还对数据库进行主从配置，一台为设置主库，一台设置为备库。当主库不能提供服务时，如主库发生宕机时，从库通过监听主库的状态发现主库宕机后，立马升级为主库，提供数据交互服务支撑，这样就能保障数据库的高可用性。

业务云采用 Spring + Spring Boot + JDBC 技术栈进行开发，架构如图 2.11 所示。Spring Boot 是一种轻量级的后端开发框架，具有很多开发模块^[21]。在开发过程中，通过应用不同的模块可以帮助进行开发工作，提高开发效率。只需要进行简单的配置以后，就可以轻松创建一个 Spring 应用。同时还内置了 Jetty、Tomcat 等 Web 容器^[22]，降低了开发难度，本文使用 Tomcat 进行辅助开发。

本文在结构上采用分层设计的准则，外层通过基于 HTTP 协议的 RESTful^[23]的 API 接口与内层交互，而内层按照 SpringMVC 开发模式设计分层。HTTP 请求首先会转发到 springMVC，然后 springMVC 根据请求信息中的包含的 url、header、body 等信息找到对应的业务 API，然后进行业务处理，处理完成后返回响应结果。如果在这个过程中需要与数据库交互，则通过 JDBC^[24]技术来实现。

最后，整个项目通过 Docker 容器部署到多台云服务器上。由于业务层需求变化频繁，导致功能迭代频繁，因此代码需要经常修改更新，采用 Docker 部署能够降低运维成本^[25]。

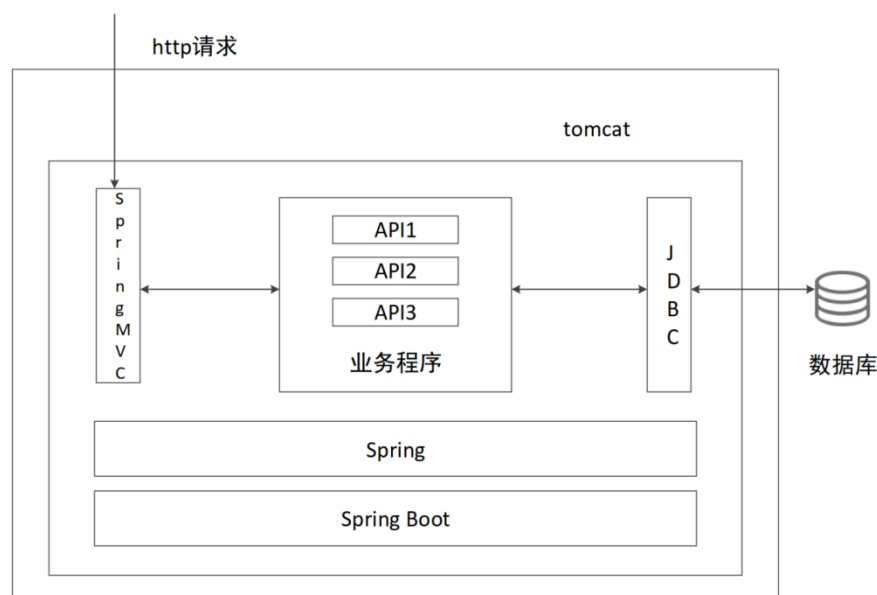


图 2.11 业务服务器技术架构

2.3.2.2 接口调用安全认证机制

API 接口认证机制一般分为有状态、无状态两种^[26]。有状态的认证机制需要在后端服务器中保存状态信息，在服务器部署集群的时候，每台服务器都要保存状态信息，不利于集群扩展。本文采取无状态的 API 接口认证机制，利用 JWT(Json Web Token)机制进行无状态安全校验。JWT 是一种 token 认证方案，利用密钥加密后存储相关信息，解密后可以得到相关信息^[27]。JWT 的基本原理是当用户信息发送到服务器后，经过其授权之后生成一个 JSON 格式的加密信息再返回给用户。当用户下次访问服务器时需要带上这个加密信息，经过服务器认证为合法请求后才被允许继续与服务器通信。为了避免出现用户故意修改数据内容的事情发生，服务器会在加密信息中额外加上签名。这样服务器就不用保存状态信息而变成无状态了，

服务器与客户端之间就不用保持长连接了，降低带宽损耗。**token** 信息总共由三部分组成，分别为：头部 **header**、载荷 **payload**、签名 **signature**^[28]。

本文设计的 **token** 认证方案流程是：用户登录成功后，基于用户 **userId** 生成 **token**，再将其与 **userId** 一同返回给请求者。在这之后，用户发起的所有接口调用请求时都要在 **http** 请求头中带上用户 **userId** 与 **token**。服务器用私钥解密 **token** 之后可得到里面包含的 **token** 过期时间和 **userId** 信息。如果 **token** 未过期的话，那么将它与传入的 **userId** 进行比较，结果相同的话则代表认证通过，认证通过之后才能进行具体的业务逻辑操作。不过，在登录成功之前的用户注册与登录 **API** 调用不需要认证。**API** 安全认证的流程如图 2.12 所示。

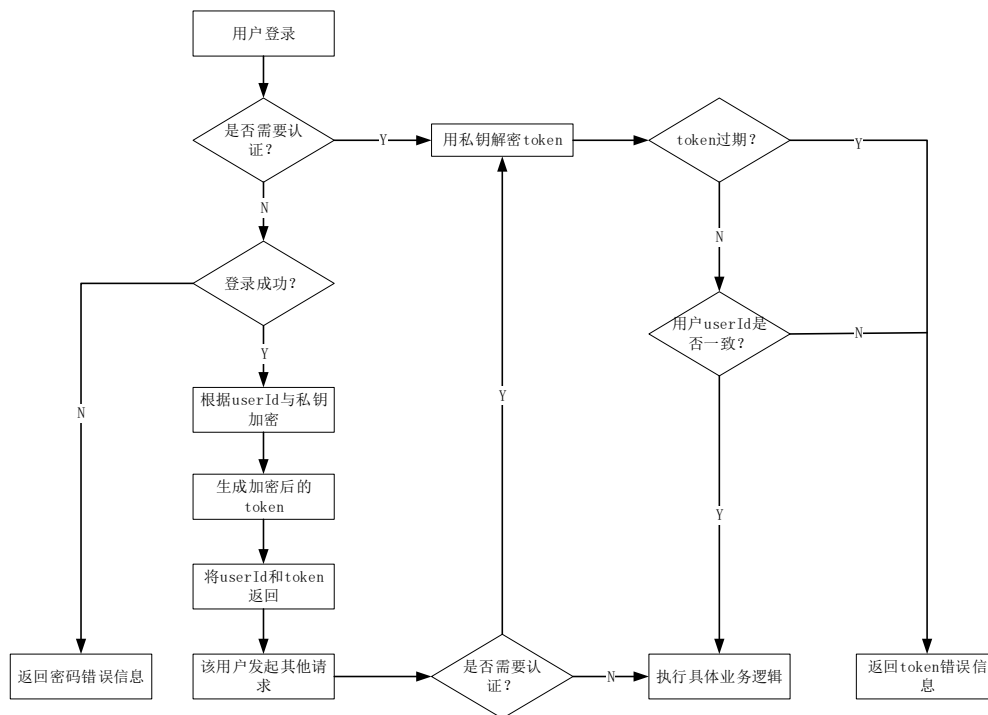


图 2.12 API 安全认证流程

2.3.2.3 用户请求信息加密传输设计

用户注册后的身份信息如果以明文的方式存储在数据库以及在网络间传输时，那么就会存在数据泄露的风险。因此需要对用户信息进行加密操作，在用户注册时的数据使用加密算法进行加密操作后存储在数据库中，数据传输时使用加密算法对传输信息进行加密^[29]。

整体信息传输加密、解密流程如图 2.13 所示。具体流程可分为以下几步：

(1) 用户注册时

- a、首先验证用户手机号码是否合法
- b、验证通过后，登录密码经由 MD5 算法加密后保存在数据库中。

(2) 用户登录时

- a、前端发送登录请求。

- b、后端服务器利用 RSA 算法生成公钥、私钥，并且将公钥返回给前端。
 - c、前端利用 RSA 公钥加密请求数据，后台利用对应的 RSA 私钥进行数据解密。
 - d、服务器去数据库中获取用户加密后的密码，用 MD5 解密之后。验证用户是否为合法请求用户，如果是的话生成验证 token。
 - d、登录成功，返回由 RSA 私钥加密过的数据，其中包含有 token 等信息。
- (3) 登录成功后
- a、前端发送业务请求，携带 token 参数。
 - b、服务器校验 token 是否正确以及是否过期，验证通过后处理业务请求。
 - c、向前端返回业务数据。

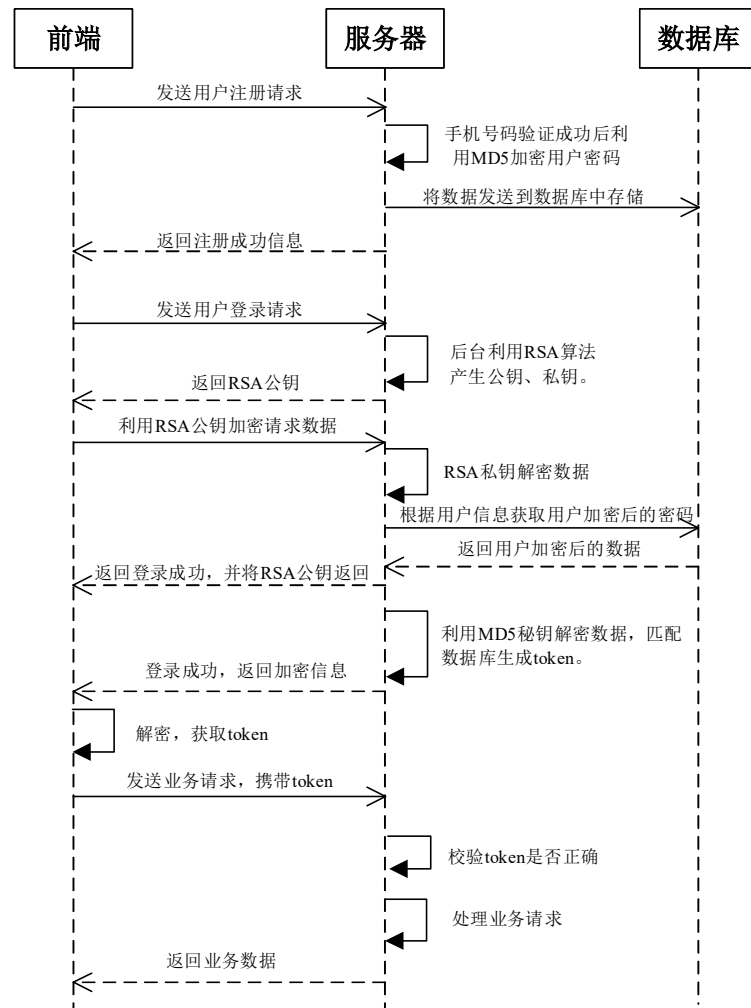


图 2.13 用户信息传输加密流程

2.3.2.4 用户权限管理机制

随着物联网技术的快速发展，需求迭代的速度越来越快，企业的管理系统构造日渐复杂。物联网管理系统主要涉及到人员管理、设备管理、业务管理等不同的功能模块。其中，在用

户管理方面，由于业务场景需求快速变化导致用户管理功能越来越复杂。以前的 ACL 单一模型无法满足企业发展的需求。复杂的系统设计存在用户非法访问及数据库设计混乱的问题，对系统带来严重的数据安全隐患。

传统的 ACL（Access Control List，权限控制列表）模型中^[30]，用户与权限直接关联。每一项资源都拥有一个对应的权限列表，用于记录哪些用户拥有执行该资源的权限。当用户访问该资源时，系统先会先去列表中查询该用户能否操作该资源。但是，此模型随着系统规模的扩展，其结构日渐复杂。

RBAC（Role-Based Access Control）基于角色的访问控制^[31]，此模型中增加了角色的概念，一个角色对应一种或多种权限。通过用户关联角色，角色关联权限的间接关系实现了用户与权限之间的直接联系。一个用户可赋予其多个角色，并且每一个角色具有不同的权限。由此构成了“用户-角色-权限”的结构来间接达到赋予用户多个权限的目的。间接的设计方案使得用户与权限之间形成了低耦合的关系，减少了 ACL 中高耦合带来的复杂结构问题。此模型如下图 2.14 所示。

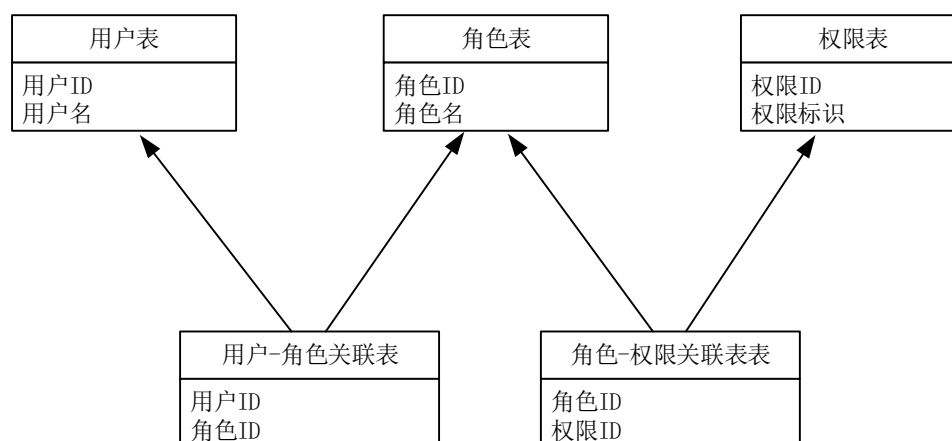


图 2.14 RBAC 权限模型图

在 ACL 的直接的权限设计方案使得用户权限的调整变得很困难，角色关联的所有用户在这个角色被赋予的权限受到修改时都会受到一定的影响，牵一发而动全身，这种情况在管理员的身上显得更加明显。间接的权限设计方案更利于权限管理。特别是对于多权限调整的难题，现在只需要调整用户已关联的角色权限即可，而不用一个个地修改。这样的设计方案在提高权限调整的效率的同时也降低了系统权限方面出现设计漏洞的概率。

2.4 Web 端设计

整个系统会涉及到不同职责的人员，主要分为：（1）超级管理员：开发本系统的企业人员，拥有最高级别的权限，可管理所有购买使用此系统的人员，给他们分配权限。（2）管理员：即从企业购买系统用于服务自己公司设备的客户，购买后用于管理自己的设备。（3）普

通用户：可对公司设备进行编辑修改，不能进行增加、删除操作，可查看设备数据。(4)：查看者：拥有最低权限，只能查看已添加设备数据，不能进行增删改操作。超级管理员拥有系统的所有权限，其他职员根据自己职能的不同拥有不同的权限。

Web 端开发管理平台，提供企业生产管理需要的功能，为企业管理人员提供设备管理、人员管理、可视化服务和数据分析服务。

2.4.1 技术架构设计

本系统采用前后端分离的形式进行开发，后端进行业务逻辑处理，提供前端所需的数据。而前端通过后端提供的访问接口进行功能交互与数据传递。整个前端在 Vue 框架的基础上进行开发。Vue 是一个轻量级、渐进式的框架，它本身包含数据绑定和组件系统这两个核心的部分^[32]。在开发的时候，我们可以根据实际的需求添加需要的组件，如路由管理、状态管理、第三方组件等。

前端技术架构如图 2.15 所示，大致可分为组件系统、网络请求、路由管理、状态管理、第三方组件五部分。

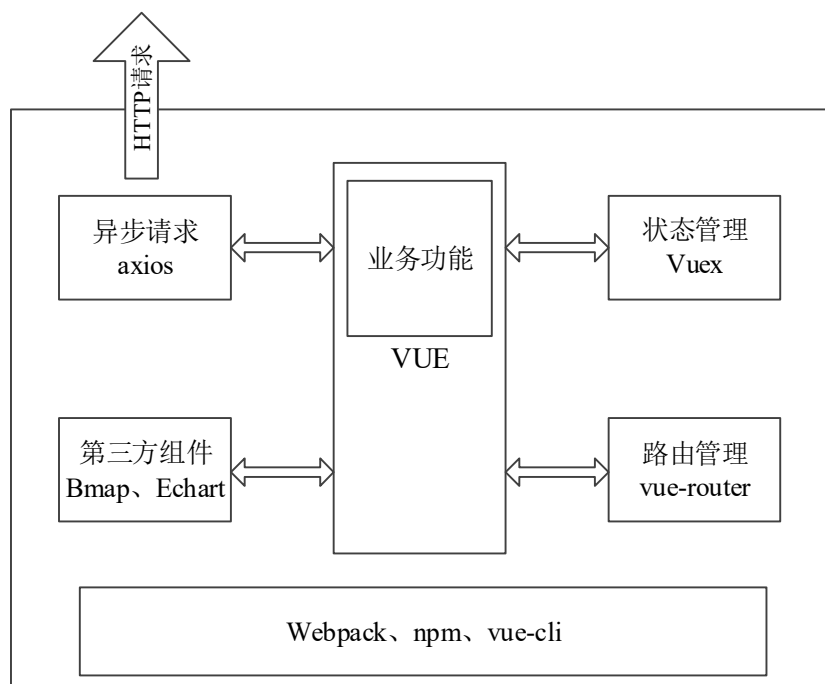


图 2.15 前端技术框架

2.4.2 Web 功能设计

Web 管理平台是工厂用来对所有设备进行统一管理的平台。如图所示，提供数据监测、设备管理、用户管理、报警管理、历史数据管理、网关基本配置共六大模块功能，实现设备远程监控、多用户管理、数据可视化分析等功能。整个 Web 端的功能结构如下图 2.16 所示。

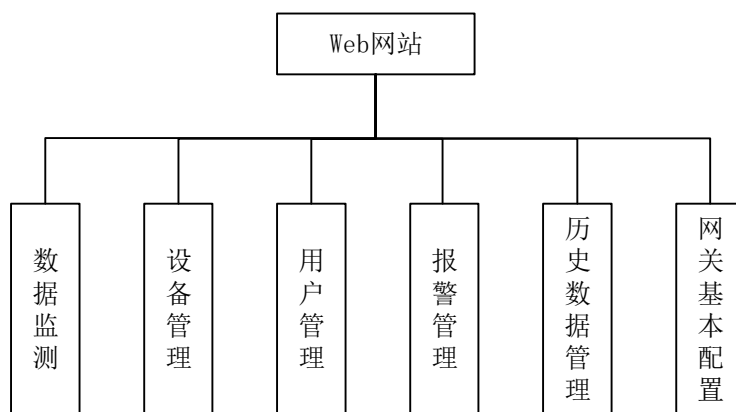


图 2.16 Web 端功能结构

每个模块的详细功能概况如下表 2.2 所示。

表 2.2 web 管理平台功能表

一级分类	二级分类	功能详细描述
登录		只提供登录功能，不提供注册，操作人员账号信息由管理员分配。
数据监测		监控当前网关下添加的所有节点采集上报的数据
设备管理	分组管理	建立网关分组，便于管理。
	添加网关	添加设备绑定的网关，输入基本数据。
	地图展示	地图展示所有网关的位置信息，可在地图上点击查看某台网关的详细信息。
用户管理		创建新用户，分配用户初始账号信息并设置用户权限；同时可删除用户、修改已有用户信息及拥有的权限。
报警管理	当前报警	可查看当天的设备报警信息。
	历史报警	提供查看过去一段时间的报警信息，可任意选择时间段查看。
	报警值设置	添加监控点，设置报警触发器，可查看已设置好的监控点信息。
	报警联系人设置	设置接收报警信息推送的联系人，可选择是否启用。
	网关报警设置	设置是否允许网关进行报警数据推送，并且可更改报警时间间隔。
历史数据		表格或者列表展示设备历史数据，可选择时间段查询；并且提供 EXCEL 导出数据功能。
网关基本设置	网关基本信息	提供查看网关详细信息，可编辑修改。
	网关位置	查看网关具体经纬度信息，可编辑修改。
	节点信息	查看当前网关下所有节点信息，并可添加新节点。

Web 端设计了独特的可视化管理模块，将所有的网关展示在地图中，管理员可对网关的位置、在线状态、ID 等设备信息一目了然，并且可通过地图展示面板对设备进行方便、快捷的信息查看与管理，快速发现设备相关报警信息，及时处理。

Web 端可视化界面如下图 2.17 所示。

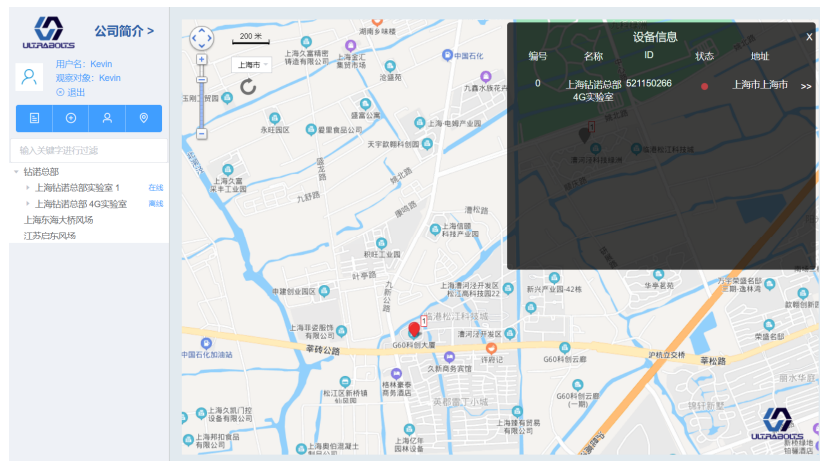


图 2.17 Web 端可视化界面

本文设计的螺栓预紧力实时监测系统通过在 Web 网页端添加与设备关联的触发器来达到预紧力阈值报警推送的目的，及时提醒有关人员检查、处理报警的设备，降低生产事故的发生，提高企业生产效益。

触发器用于监控设备数据的变化，当它的数据达到设置的触发条件时就会进行报警推送，以备相关人员及时进行处理。添加触发器操作需要 Web 端、OneNET、后端业务服务器、智能网关四者共同配合完成整个过程。此操作在 Web 网页端添加了设备的监控点信息，可随时查看与修改。在 OneNET 设备云平台上添加了触发器，添加成功后可对设备的数据进行实时监控，并对监控的信息进行存储，以备后续查询。后端业务服务器在数据库中保存监控点相关信息，同时调用 OneNET 平台提供的 API 为其添加监控点。网关端添加与设备绑定的监控点，上报数据到 OneNET 平台。不同设备有着不同的报警触发条件，用户可以通过设置相应的解析规则来达到区分不同的监控信息的目的。添加触发器的操作可概括为以下几步：

(1) 首先在网页端点击添加触发器操作，填写报警触发条件信息；Web 端根据填写的信息生成 StreamId 后调用添加监控点的 API 接口。

(2) 后端业务服务器收到接口调用后首先往数据库中存储监控点信息，同时在 OneNET 上调用新增数据流的 API。接着判断监控点绑定的网关是否在线。

(3) 如果网关在线的话，异步化向 OneNET 平台发送添加监控点命令，让其转发给网关。网关收到命令后在本地保存 StreamId，并向 OneNET 回复执行成功的命令。

(4) 如果网关离线的话，则将添加监控点命令写入待发送命令表，然后向网页端返回网关不在线的提示信息。此时 Web 网页端显示的监控点是无效的，不会有任何实际数据显示，也无法进行编辑操作。当网关上线后会触发自动添加监控点操作，此时 Web 网页端不进行操作，只需要后端业务服务器、OneNET 平台、设备三者参与。

上述步骤中，后端业务服务器通过 OneNET 将命令发送给网关。由于网关是与 OneNET 进行数据交互，所以服务器想要获得网关的响应信息时需要调用 OneNET 提供的查询命令 API

向网关查询。但是，这个查询响应时间会因为网络的原因而变得不确定。所以，在向网关发送命令时，首先会去判断网关是否在线，如果在线才会进行查询操作。等待 10 秒钟，如果收到发送成功的回复信息，则代表完成操作。否则，则将添加监控点的命令存入待发送命令数据表，系统会间隔一段时间后重新将待发送命令数据表中命令按时间顺序重新发送出去。当刷新 Web 网页端时，新添加的监控点有显示实时数据时，代表在网关端的监控点添加成功了。可以看出，无论网关是否在线，添加监控点的操作都是有延迟的，故需要设置延迟等待时间。

添加触发器时各部分整体协作流程如下图 2.18 所示。

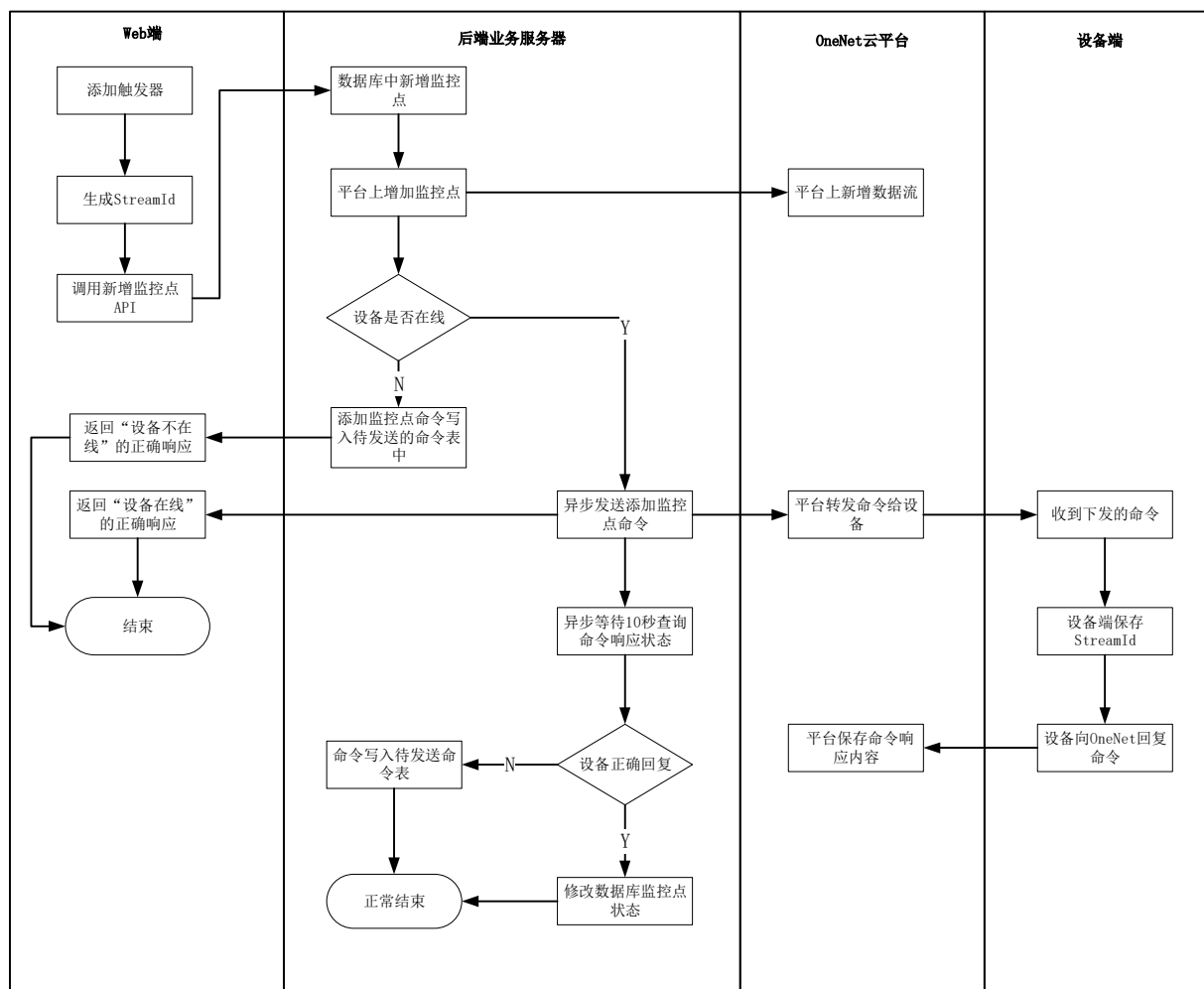


图 2.18 添加触发器流程

在 Web 上将触发器添加成功之后，OneNET 服务器会根据设置好的触发条件去判断是否发生触发事件。当发生报警触发事件时，将报警信息推送到后端业务服务器，接下来由后端业务服务器进行报警数据的转发。首先，后端业务服务器会去查询此次触发事件是否开启允许报警数据推送功能。如果开启了报警推送功能，后端业务服务器将接收到报警信息解析，将数据保存到数据库中之后再以短信的方式将报警信息发送给管理员。如果此时 Web 端处于登录状态的话，无论用户此时访问的是哪个页面，都会收到报警信息的弹窗提醒。如果关闭了报警推送功能，后端业务服务器向 OneNET 发送暂停推送报警信息指令，待开启后再通知

其打开该功能。

平台报警推送及接收流程如下图 2.19 所示。

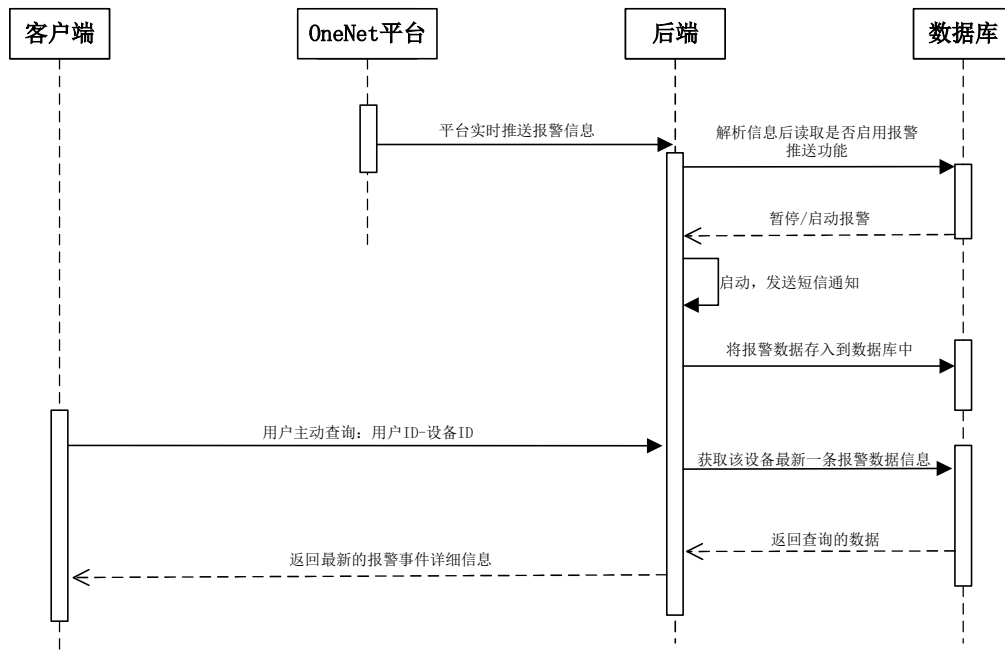


图 2.19 报警接收与推送

2.4.3 Web 功能实现

2.4.3.1 设备管理

设备管理模块涵盖以下功能：设备分组管理、设备安装信息显示、网关在/离线状态、节点在线/离线状态、网关删除/添加。部分功能展示如下：

（1）设备分组管理

设置网关分组，便于管理，可根据分组信息直接查看相关位置安装的所有网关。

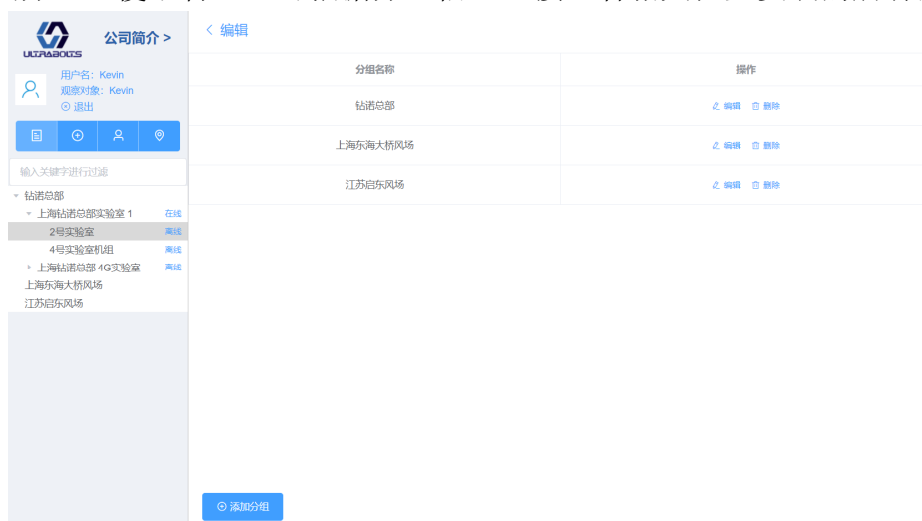


图 2.20 设备分组管理页面

（2）网关添加

输入网关的 SN、密码,经过云平台验证通过以后才可成功添加网关。在网关添加的时候,网关必须处于在线状态,这样可以避免网关离线时由于输入错误的参数信息时依然可成功添加网关。添加网关时刻输入相应的经纬度,同时支持点击地图某点位置添加网关。

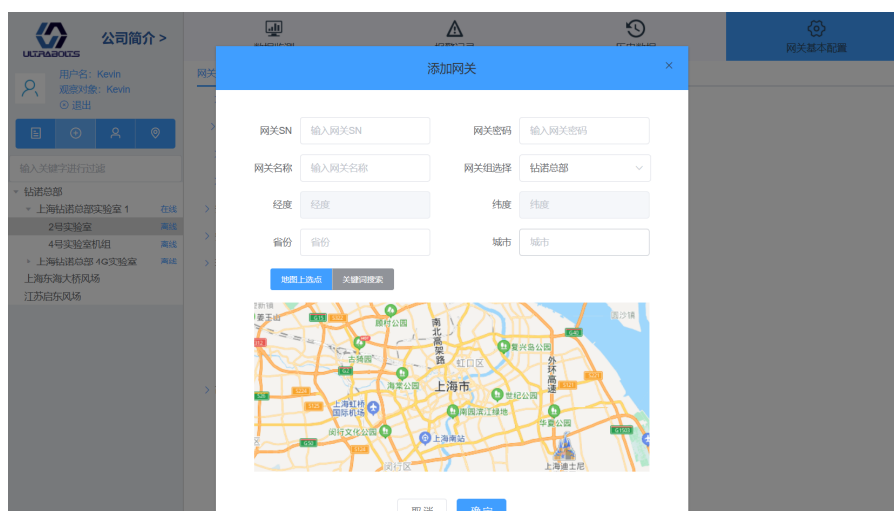


图 2.21 添加网关页面

(3) 设备地区分布

查看某分组下的网关在地图中的位置分布,同时可点击查看某台网关的具体详细信息。

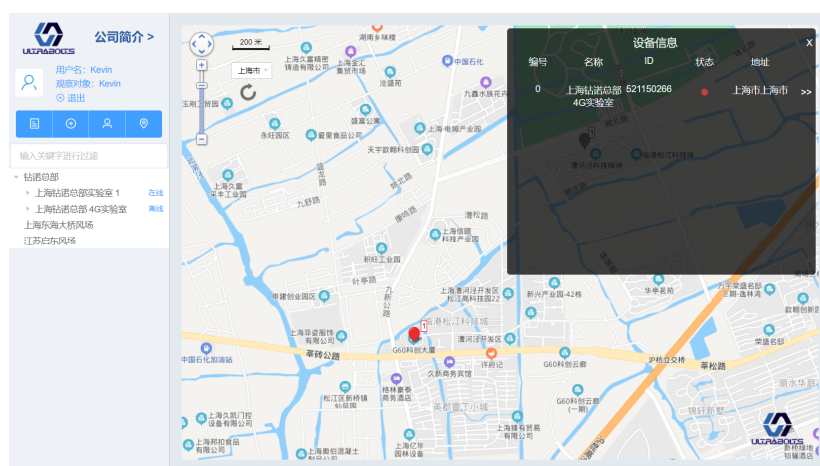


图 2.22 设备分区展示页面

2.4.3.2 用户管理

不同的用户根据其职位分配相应权限,权限分为 4 级,分别为:超级管理员、管理员、查看者、普通用户。

(1) 用户管理

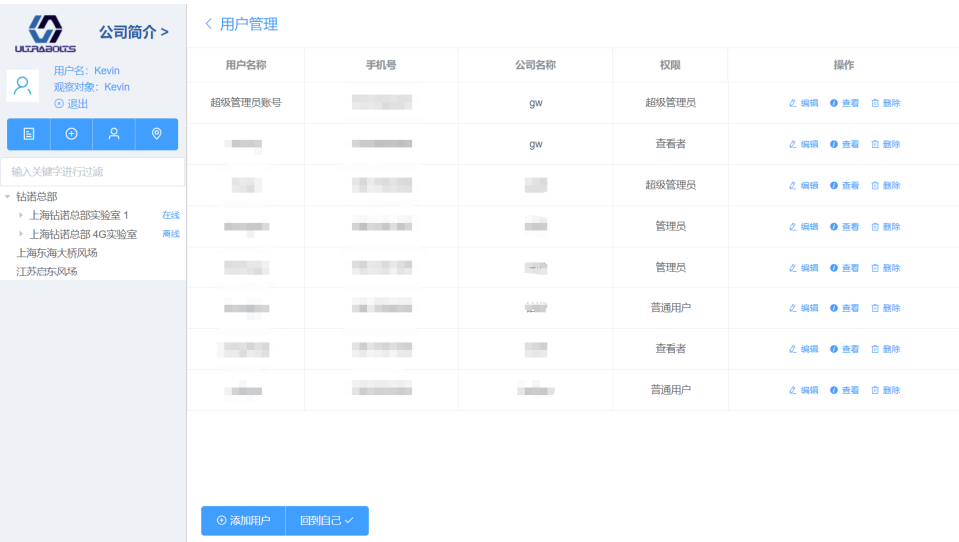


图 2.23 用户管理页面

(2) 添加用户

在此处可添加、编辑用户信息，分配用户初始权限。



图 2.24 添加用户页面

2.4.3.3 数据监测

提供以折线图、柱状图、表格的多种方式展示实时数据、图形全屏显示、节点状态信息、安装位置、安装时间、MAC 地址显示等功能。

(1) 数据实时监控

实时展示设备采集的数据，以表格、折线图、柱状图的方式展示，便于分析。



图 2.25 数据监测页面

(2) 数据全屏展示

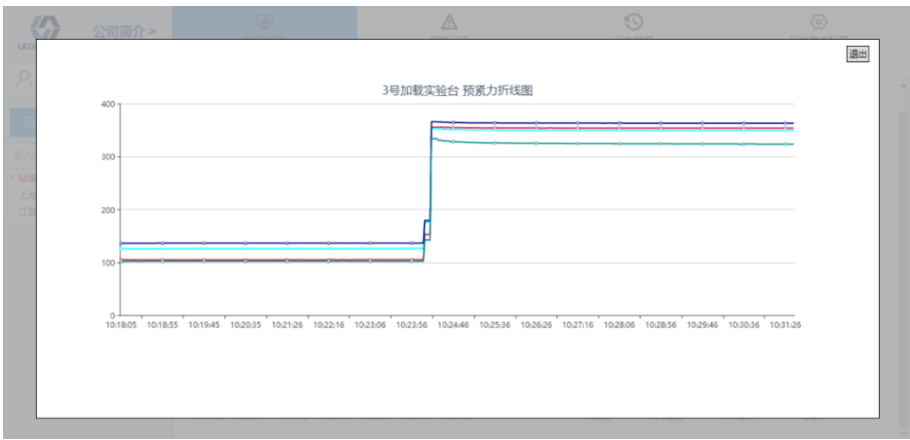


图 2.26 数据全屏展示页面

2.4.3.4 报警管理

提供当前报警、历史报警、报警值设置、报警联系人设置、网关报警配置功能。部分功能展示如下。

(1) 当前报警



图 2.27 当前报警页面

(2) 历史报警



图 2.28 历史报警页面

(3) 添加触发器

通过添加设备报警触发器，达到设备故障报警的目的，可根据设备类型设置不同的报警条件。



图 2.29 报警值设置页面

2.4.3.5 历史数据管理

可设定时间范围查询所有的数据记录，以列表或图形形式展现，并可导出报表数据。部分功能展示如下：

(1) 历史数据查询



图 2.30 列表展示历史数据页面

提供折线图的方式展现历史数据。



图 2.31 折线图展示历史数据页面

2.4.3.6 网关基本配置

提供查看、配置网关基本信息，部分功能展现如下：

(1) 网关基本信息



图 2.32 网关基本配置信息页面

(2) 节点添加

添加监控节点后，传感器便与网关绑定，此后设备采集的数据上传至云平台。提供监控点编辑功能，参数设置出错的话可重新设置。



图 2.33 监控点添加页面

2.5 本章小结

本章主要介绍螺栓载荷实时监控系统的总体设计方案。从实际的需求出发，分析了系统设计的技术特点，梳理出系统的整体框架。针对硬件、云平台、客户端分别进行设计，并且对每个部分的设计原则进行了详细的阐述以及效果展示。

第3章 分层式数据处理机制设计

物联网远程运维项目中，时刻保持着数据的传输。但是，由于其每次传输的数据量小，频次高，所以长时间的数据传输造成的数据量是很大的。为了保证设备通信的便携性，目前大多数设备的数据传输方式采取 4G 的方式，数据量大所带来的带宽费用也很高。

在远程运维系统中，设备需要时刻与云平台保持长连接，并实时上报数据到云平台，实现数据存储及相关业务逻辑处理。目前的方案是将设备与云平台保持长连接，并且通过心跳机制确定设备在线状态，从而实现设备数据实时上报功能。但是，在设备数量多、数据上报量大的情况下，随之会带来以下几个问题^[47]：

（1）带宽成本高。目前，智能设备与云平台之间一般通过 4G 网络进行数据传输。长期的数据传输会带来数据量的提高，由此所消耗的通信费用也就越多。

（2）存储成本高。设备上报的数据要存储在设备云平台及业务云平台中。数据量越大，平台所需要的存储空间也越来越大，随之带来的存储成本也越来越大。并且，数据量过大带来的维护与管理的成本也越来越大。

根据以上的分析，优化数据传输具有重要意义，可降低企业成本，同时也会提高整个系统运行时的性能。

在对远程运维系统进行传输优化时，可从两个方面传输速率与数据传输量进行着手。传输速率受传输协议和网络带宽的限制，从此处优化得到的效果不会有明显提升。从数据传输量处着手，通过过滤掉重复数据、无效数据等一些有效机制可达到传输优化的效果。

当前的数据传输架构采用的是由下到上为设备-平台-客户端的三明治结构。传统的数据传输采用的是透传的方式，即将数据地从一端传输到另一端，传输过程中不会对数据内容做任何修改。

3.1 分层式数据处理机制设计

3.1.1 分层式数据处理机制架构设计

在现有的远程运维系统中，设备不与应用层直接进行数据交互，而是通过设备云平台进行中间数据转发。分层式数据处理机制分为三个层级，它们相互之间进行数据交互，从而形成一个整体。分层式结构为我们的数据传输优化带来了便利，可根据层级之间的传输特点来设计对应的数据传输优化方案，避免了在一体式结构中出现的一处修改而影响整个系统的问题。分层式数据处理机制架构如下图 4.1 所示。

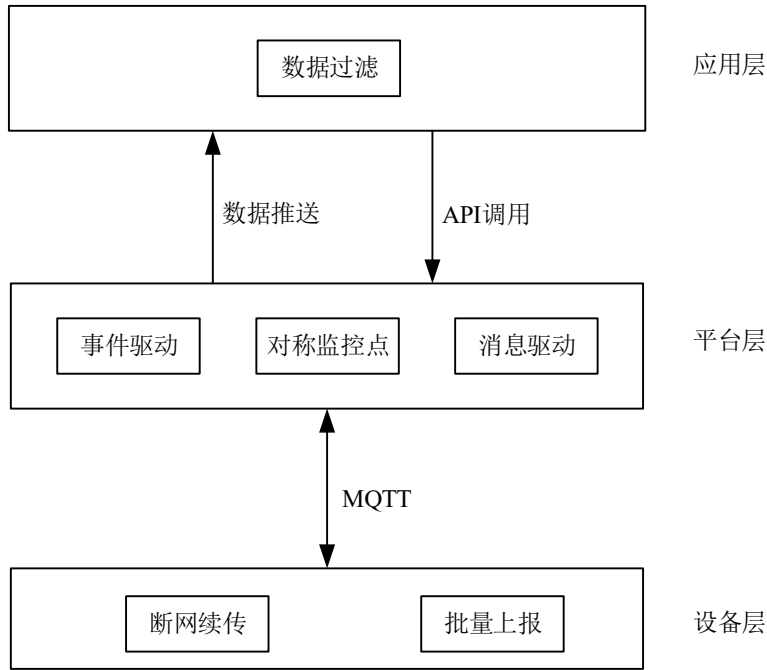


图 4.1 分层式数据处理机制架构

如图 4.1 所示，分层式数据处理机制采用“应用-平台-设备”的三明治架构。此架构是整个数据传输优化机制的核心，由上到下分别为应用层、平台层、设备层。通过分层式数据处理机制，针对不同层之间数据交互的特性来进行优化后降低整个系统数据传输过程中产生的数据量，从而达到优化整个数据传输的目的。

应用层，即客户端。用户使用客户端与设备进行远程交互，提供查看设备信息、设备控制等功能。

平台层，即云平台，包含设备云平台及业务云平台。上接应用层，下接设备层。设备云处理与存储设备上报的数据，业务云处理业务方面的数据。

设备层，采集设备的生产数据，并上报云平台。同时，接收云端发送过来的命令，执行一系列操作。

3.1.2 分层式数据处理机制设备层设计

设备时时刻刻都在进行数据上报，从此处着手进行优化的效果更佳明显。通过设计断网续传、数据批量上报策略进行数据传输优化。

3.1.2.1 断网续传策略

当网络不畅的时候，设备采集的数据就容易产生丢失。当部分重要数据丢失的时候，会影响设备的正常工作。为了保证数据传输过程的稳定，以及避免传输重复及错误的的数据。设计断网续传策略时需要保证设备上传的数据时间与网关云端存储的数据时间同步，这样能够保证数据续传时的可靠性、正确性。设备上报到网关的数据的时间由网关决定，设计时间同步机制保证网关本地时间与云平台同步。同时，设计一个离线缓存队列用于网关离线时进

行数据存储及网关恢复联网时数据有序、正确地上报到云平台。

(1) 时间同步机制

NTP (Network Time Protocol) 是一种用于计算机时间同步的协议，它的目的是让计算机本地时间与服务器的时间同步化^[48]，保证了高精度的时间校正，同时它也可以经过加密确认的方式来防止网络的恶意攻击。网关联网以后需要将网关本地时间与服务器时间进行同步操作，保证后续设备上传数据的时间正确性。在网关本地写入同步脚本文件，网关重新联网之后自动执行，同步校正网关本地时间。其中，NTP 工作模式如下图 4.2 所示。

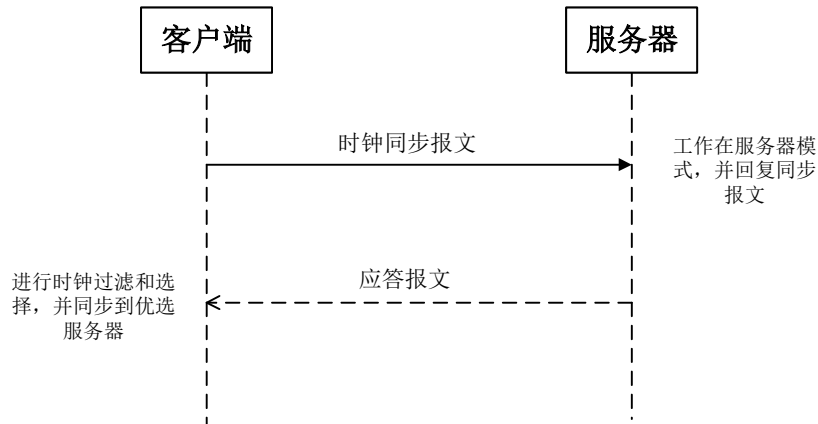


图 4.2 NTP 工作模式图

网关与服务器之间进行时间同步的整体流程如下图 4.3 所示。

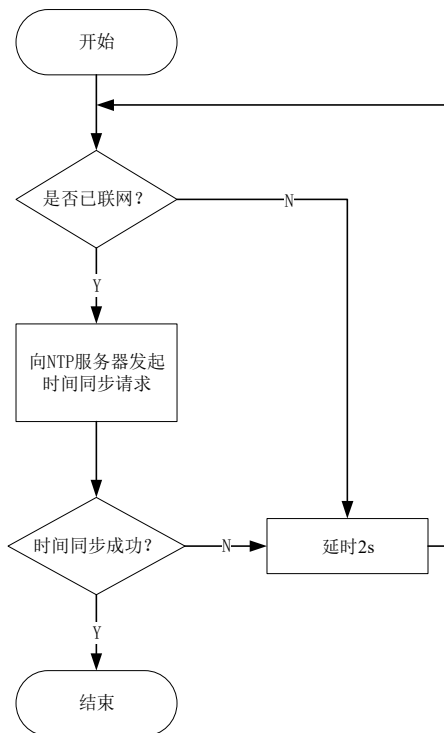


图 4.3 时间同步机制整体流程图

(2) 数据离线缓存队列

时间同步结束之后，开始进行数据采集工作。网关中设计一个离线缓存队列，当它处于离线状态时，设备上传的数据暂时保存在队列中。数据从队列的尾部插入，从头部取出。当队列容量不足时，从头部位置选取部分数据删除。因为头部数据相对于尾部数据的存在时间较长，价值更低。删除队列头部数据保证了网关本地内存不会溢出。离线缓存队列工作流程如下图 4.4 所示。

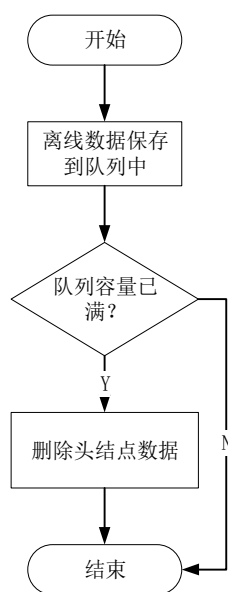


图 4.4 离线队列数据存储流程

时间同步机制保证了设备数据与云端数据的时间连续性，不会产生时间错位的问题。当网关离线时，设备不会停止工作，依然将采集的数据上传到网关。网关将这些数据保存在队列中，等待网关重新联网以后将数据批量上报给云平台。云平台解析完数据信息以后，根据上报数据格式中包含的序号及时间戳信息，进行数据存储及数据处理，保证了数据得到正确处理，同时也保证了数据不会丢失。

3.1.1.2.2 数据批量上报策略

(1) 批量上报机制

在设备传输数据时，对于实时性要求低的数据，可采取数据批量上报策略分批次一同上传到云平台。当网关进行数据上报时，如果每条数据都进行单独上报时，会产生多余的网络开销。这些多余的开销主要由 TCP/IP 协议、MQTT 协议造成。在建立数据通信之前，TCP/IP 协议会进行三次握手建立连接的过程，断开连接时还会进行四次挥手的过程。如果每条数据都采用单独上报的策略，这样就会产生大量的 TCP/IP 握手连接与挥手断开的过程，产生的开销巨大，并且影响系统的性能。在数据通信过程中，每条数据包含了多种协议的首部信息，这在传输校验过程中是难以避免的。通过将多条数据的有效内容汇聚在一起进行上报，减少

了协议首部校验数据的传输。

为了减少数据上报时产生的开销，同时保证数据实时有序的上报到云平台。设计一个缓存队列，用于存储设备产生的数据。当队列容量达到规定的上限时，将队列中的数据批量有序地上传到云平台。同时，为了保证数据上传的实时性，设置定时上报机制。当队列容量没有达到阈值而时间间隔达到时也进行数据批量上报。相较于数据单条上报，批量上报机制减少了数据传输过程中传输信息首部数据包带来的额外开销，同时也避免了频繁地 TCP/IP 三次握手的开销。而且，云平台也可以批量处理网关上报的数据，提高数据处理的效率。数据批量上报的流程如下图 4.5 所示。

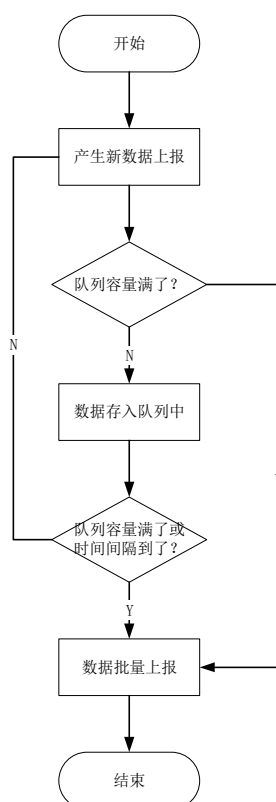


图 4.5 数据批量上报流程

(2) 断点续传机制

为了避免在数据批量上传过程中出现网络故障而造成数据大量丢失，设计断点续传机制。将数据分成多段，每段数据采取不同的线程进行传输。当在数据传输的过程当中出现网络故障，只需要上传未完成的部分，无需从头开始重新上传，减少了数据重复上传的情况。同时，将数据分段并行上传能够提高传输的效率。数据切分成多部分之后，云端的数据段顺序是通过上传过程中指定的序号来确定，按照序号进行数据恢复。将上传的数据进行分段，除了最后一个分段，其余分段大小设置 1024 byte。根据文件内容生成 hash 码并给其按序编号，作为文件的唯一标志。假如文件上传过程中发生断网，则在网络恢复连接后，文件恢复上传。在上传前计算出 hash 值，并把 hash 发送到云平台，一旦云平台找到相同 hash 的文件，则直接

返回文件分段已上传成功的信息，此分片无需重复上传，跳过即可。数据分片结构如下图 4.6 所示。

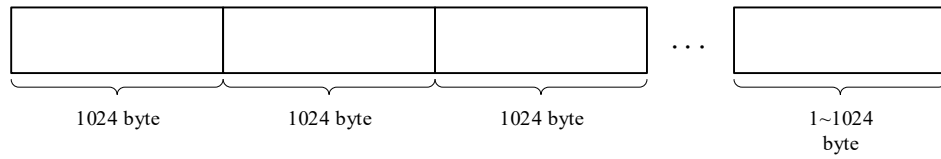


图 4.6 数据分段结构

开始数据批量上报之后，首先将数据从缓存队列中取出，以 1024 byte 大小进行分段。当网关在线时执行多个分段并行上报，直至最后一个分段完成上传后结束整个数据上报过程。在断点续传的过程中，如果出现网关掉线的问题，设定 10 秒延时等待网关上线，如果网关仍然没有恢复上线，则结束整个数据上报过程，等待下次网关上后自动从未完成的部分继续开始上传过程。断点续传的工作流程如下图 4.7 所示。

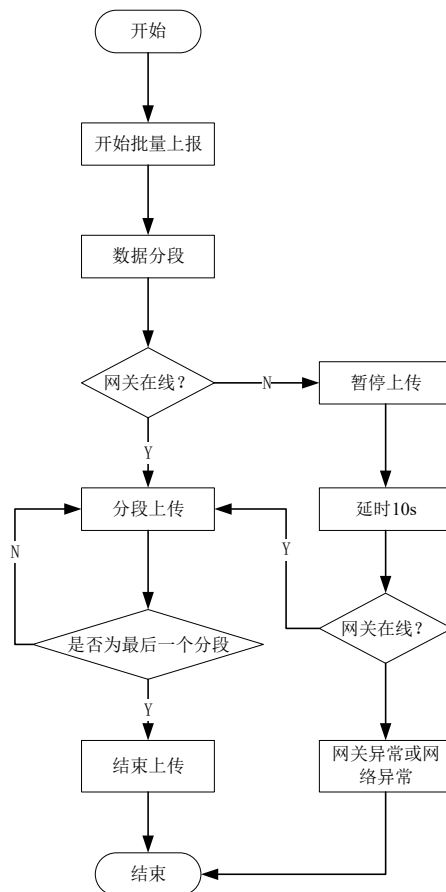


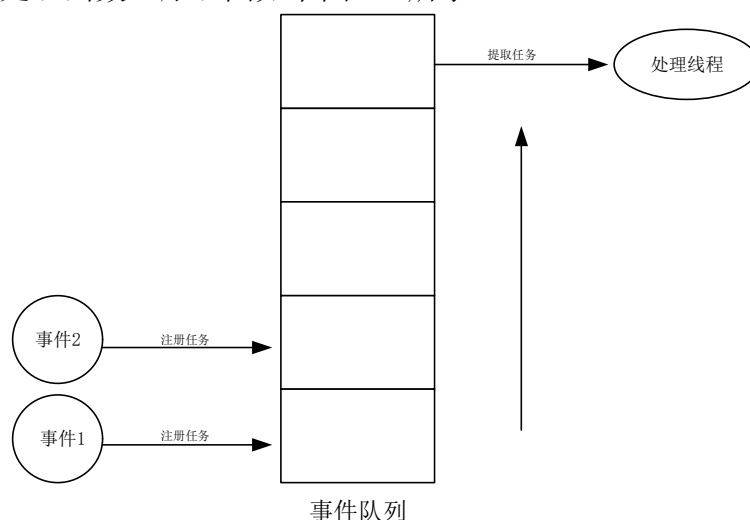
图 4.7 数据断点续传机制工作流程

3.1.3 分层式数据处理机制平台层设计

云平台作为中间层，往上与应用层交互，往下与设备层交互，具有承上启下的作用。通过事件驱动机制与消息驱动机制保证了云平台与设备层、应用层之间的数据交互。

3.1.3.1 事件驱动机制

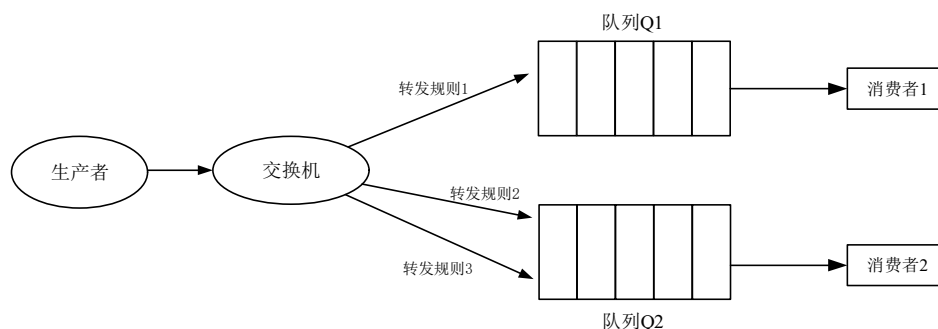
事件驱动机制是一个事件循环调用的过程，当外界某件事件产生时会利用激活的函数进行处理^[49]。它的工作流程可概括为：（1）在这个机制中存在一个事件队列，用于存储即将执行的事件。（2）当事件发生时，放入队列尾部，等待执行。（3）事件中存在一个函数，不断从队列中按序取出事件并调用不同的函数异步地执行事件。（4）事件之间没有干扰，每一个事件有其对应的独立处理函数。原理图如下图 4.8 所示。



利用事件驱动机制，客户端无需定时查询系统信息是否发生更新，数据变化时会自动触发更新操作。事件的发起者将事件注册到事件队列中，此机制监测到事件后将事件传输给处理线程以异步的方式进行相关处理工作。在客户端与 Web 端进行交互时，客户端希望立即获得结果，无需持续轮询，事件一旦到达就将被推送给对方。由于不再需要轮询，证明其是一个松耦合的结构，系统的工作压力将会得到降低，网络 I/O 也会减少很多。

3.1.3.2 消息驱动机制

消息驱动机制用于云平台与应用层之间的数据交互^[50]。可针对应用场景开发，例如用户接收某台网关下具有使用权限的设备数据时，按需转发数据，而不是向网关下所有用户发送信息。



如图 4.9 所示，消息驱动机制中存在生产者、消费者、交换机、消息队列。生产者对应设备、消费者对应客户端。生产者产生的消息发送到交换机中，交换机根据不同的转发规则将数据转发到对应的消息队列中。值得注意的是，同一个消息可能因为不同的转发规则而同时转发给同一个消费者。消费者通过监听自己绑定的队列，从中获取信息后再进行相应处理工作。生产者与消费者之间存在一对多的关系。生产者将消息发送出去，与其关联的一个或多个消费者会收到此消息，它们之间凭借转发规则区分消息发送方向。通过转发队列的帮助，生产者与消费者之间并没有直接关联，生产者将消息发送到队列之后就结束了对消息的处理。消费者不知道队列中的消息是谁给它发送的，不需要返回确认信息。同时，它无需知道消息的来源，只需根据消息本身执行相应的操作即可。消息驱动机制使得系统各模块之间没有直接关联，新增模块或者修改模块时对其他模块的影响较小。

使用消息驱动机制，将短时间内产生的数据存储到消息队列中，可有效降低系统高并发情况出现的概率。

3.1.3.3 对称监控点策略

不同类型的数据存储在设备中，其代表着用于监控设备的指标，每一个指标都被当作一个监控点。于是，可在云平台上创建相应的监控点将其与设备进行绑定，通过观察云平台上的监控点数据达到设备监控的目的。

通过在云平台上给设备建立一对监控点，用于表示设备当前状态及当前的监控点上实际存储的数据值。设备状态不正常时不进行数据收发，减少数据流量的产生。对称监控点策略工作流程如图 4.10 所示。

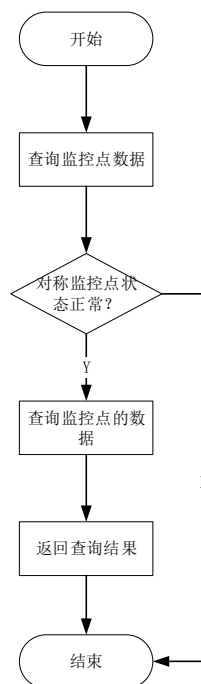


图 4.10 对称策略工作流程

对称监控点策略用于系查询监控点数据时工作，分作两步执行。（1）查询该设备下所有监控点的状态，剔除掉所有处于异常状态的监控点，不对其进行数据查询工作；（2）对所有正常状态下的监控点发送数据查询命令，上报显示异常状态的监控点。在该策略机制下，我们可以提前判断出异常状态的监控点，不对这些异常监控点进行数据查询，减少了数据请求命令的发送，同时也提高了数据查询的精准度，降低了发往云平台的数据流量，从而提高了远程数据收发的效率。

3.1.4 分层式数据处理机制应用层设计

3.1.4.1 重复数据过滤策略

设备在进行远程调试的时候，为了保证数据成功发送到设备中，一般会采取数据重发机制，反复发送相同数据帧，保证命令成功发送到设备上并执行成功。当设备或者网络出现问题时，命令发送次数重复过多的话，反而会造成数据流量的浪费。从客户端处拦截调试软件下发的重复命令数据帧，对发送的数据帧进行帧格式及关联帧检测。对于检测到的重复数据帧只进行一次转发，然后等待设备回复情况。如果设备按时回复，则表示设备成功接收到数据并执行完毕，否则再重复发送一次命令，以此确保设备接收到调试命令。若设备持续一段时间后还是没有回复，则认为设备异常或者网络异常，停止远程调试工作，并通知相关人员检修。流程如下图 4.11 所示。

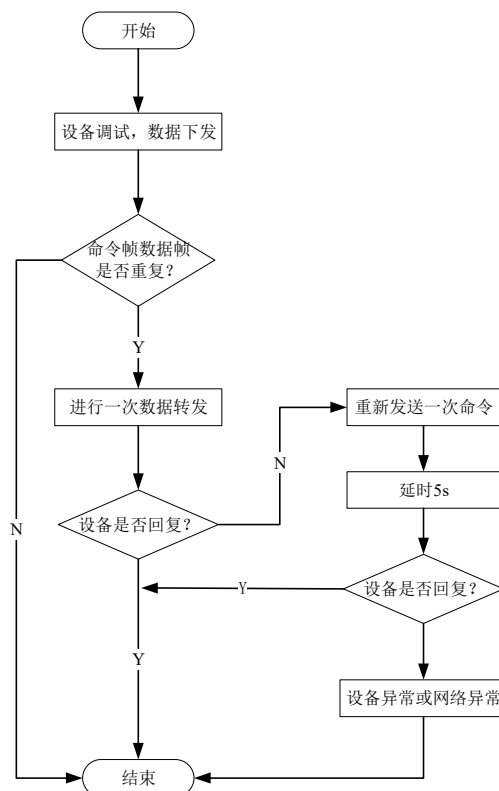


图 4.11 重复数据过滤流程图

本策略设计在客户端处，对发送的数据帧进行过滤操作，减少了实际发送到设备数据量，

从而减少了数据传输过程中的流量资费。

3.1.4.2 固件远程更新策略

设备中运行的程序需要根据实际项目需求进行不定期的更新升级。以往的升级方式是安排人员去往设备安装场地进行有线连接升级，这样对人力、物力资源要求高，企业成本高。为了使系统更加稳定和操作的便利性，设备异常时进行替换，设计互联网远程更新升级方案，远程更新设备中的程序^[51]。

(1) MQTT 方式更新

运维人员将最新版的固件压缩之后上传到服务器，服务器将压缩包转化为二进制文件。通过调用设备云平台的接口，将二进制文件分包下发到网关上，由网关操作更新程序。数据以 MQTT 协议格式按帧下发到网关，网关接收到数据之后进行数据解析，判别帧协议，将有效的数据取出，再按照压缩包的文件形式进行写入。云平台与网关之间采取一应一答的机制，根据帧序号进行数据同步，保证信息写入时不会出错。当压缩文件传输完成时，云平台发送结束帧通知网关本次文件传输已完毕。网关执行提前写入的程序更新脚本，关闭网关其他正在执行的程序，对压缩包文件进行解压工作，提取更新文件，执行更新文件后再重启网关。

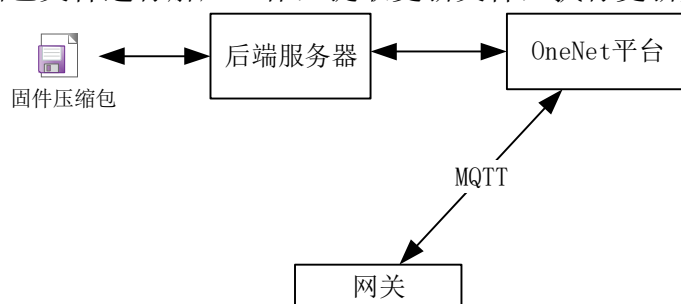


图 4.12 MQTT 方式更新固件流程

(2) HTTP 方式更新

上述 MQTT 的远程更新方案将压缩包转为逐帧文件后发送到网关，服务器与网关之间的通信过程纷繁复杂，数据传输流量较大。引入新的更新机制，减少通信过程中的产生数据传输量。将固件更新文件上传到固件云服务器之后，不再由 OneNET 云平台将程序下发到网关，而是由网关直接向服务器发起请求，直接从固件云服务器下载更新文件。后台业务服务器通过调用 OneNET 提供的 API 接口将更新命令发送到网关，告诉网关需要更新的固件版本号、文件下载网址以及文件校验码。网关端启用下载程序，通过 HTTP 的方式链接到固件云服务器，以文件下载的方式将最新版的固件下载到指定目录后再解压文件。解压之后通过验证云端校验码与网关端校验码是否一致，验证通过后网关向 OneNET 云平台上报消息告知程序下载完毕，执行安装新的固件程序。否则，告知 OneNET 云平台程序下载有误，重新下载并停止执行已下载的程序。

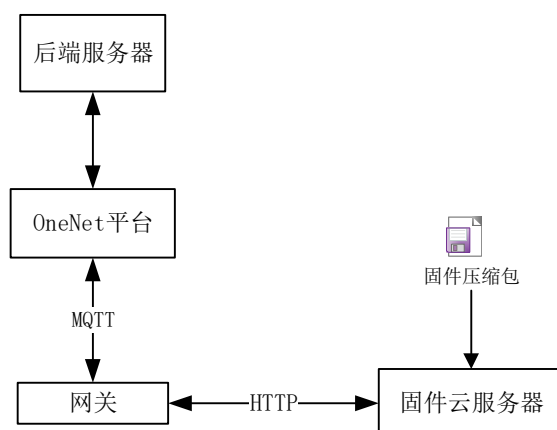


图 4.13 HTTP 方式更新固件流程

3.2 实验测试与结果分析

本次测试采用阿里云上购买的 ECS 云服务器和 MySQL 数据库。业务云服务器的配置为 2 核 8G 的 64 位 Linux 系统，系统版本为 CentOS 7.1。MySQL 数据库的配置为 2 核 4G，数据库版本为 MySQL 5.6.32。网关固件更新文件存储在业务云服务器中。

3.2.1 数据传输流量测试

本次测试针对系统运行时的整体数据传输量大小进行，分别测试透传数据传输模式下与分层式数据处理机制下的数据传输量的大小情况，通过测试数据反映出实际使用过程中的数据流量使用情况，从而说明分层式数据处理机制在降低数据传输量上效果情况。

3.2.1.1 测试方案

本次测试方案针对 4G 模式与 WIFI 模式分别进行，两种模式下保持相同的网络环境与测试环境。测试时间为 2 小时，每 10 分钟统计一次数据传输量，共记录 12 次。两种传输模式下，设定网关间隔 30 秒向云平台发送一次心跳帧，保持云平台的长连接，每次的心跳帧大小为 14 byte。网关向云平台间隔 10 秒发送一次设备采集的数据，大小为 30 byte。4G 模式下监测网关上的网卡流量使用情况来作为实际传输所消耗的数据量，WIFI 模式下利用路由器管理系统监测设备消耗的数据量，记录不同模式下的数据传输量，结果如下表 4.1 与 4.2 所示。

3.2.1.2 测试数据

4G 传输模式下的数据测试结果如下表 4.1 所示。

表 4.1 4G 模式下数据传输量测试结果

测试时间(min)	透传模式数据量大小(byte)	分层式模式下数据量大小(byte)
10	5610	2132
20	11192	4365
30	16356	6510
40	22160	8820
50	29171	11085
60	33903	13221

70	39752	15424
80	46157	17540
90	50183	19672
100	54689	21821
110	61730	24013
120	67148	26188

将测试结果转成折线图后，如下图 4.14 所示。

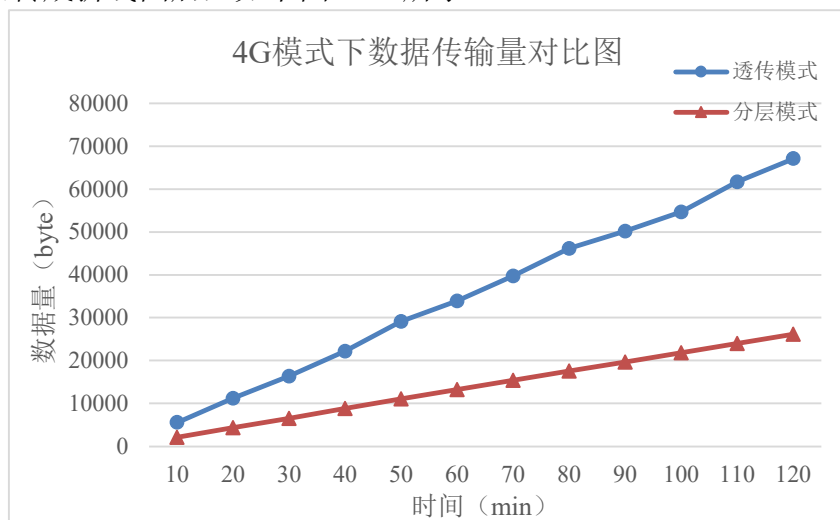


图 4.14 4G 模式下数据传输量对比图

WIFI 传输模式下的数据测试结果如下表 4.2 所示。

表 4.2 WIFI 模式下数据量传输结果

测试时间(min)	透传模式数据量大小(byte)	分层式模式下数据量大小(byte)
10	5451	2126
20	11143	4346
30	16448	6532
40	22140	8856
50	28966	11063
60	34247	13242
70	39517	15412
80	45970	17526
90	50158	19662
100	54726	21836
110	61715	23884
120	67217	26215

将测试结果转成折线图后，如下图 4.15 所示。

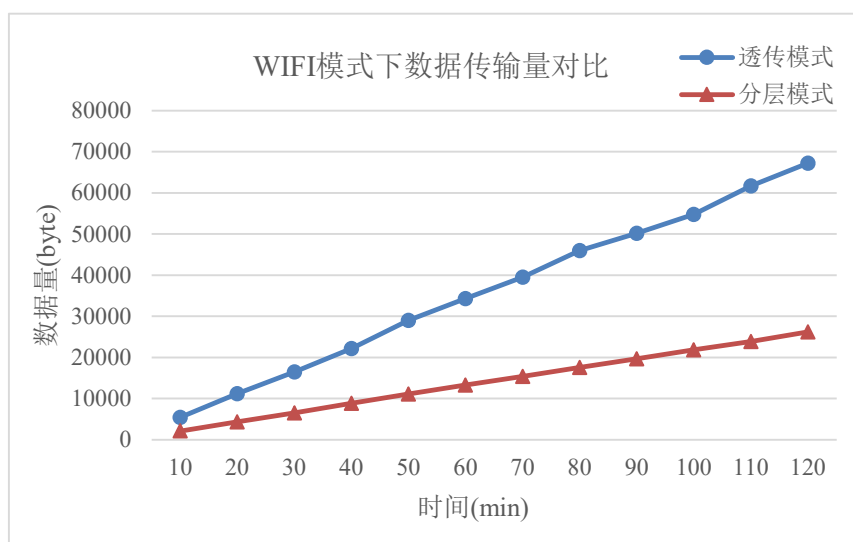


图 4.15 WIFI 模式下数据传输量对比图

3.2.1.3 测试分析

从表 4.1 和表 4.2 的测试结果中,分析结果后可以得出,网络的传输方式不影响分层式数据处理机制的工作。4G 模式与 WIFI 模式下的数据传输量优化效果相差不大,两种方式带来的差异在于数据传输速度的不同,这个由网络带宽决定。

从图 4.14 及图 4.15 中可以看出,随着测试时间的增长,两种模式下的测试结果曲线都呈增长趋势。但是本文提出的分层式数据处理机制下的数据量增长较缓慢,而常规的透传模式下的数据传输量随着时间的变化增加得较快。从测试结果可以看出,分层式数据处理机制对降低数据传输量的效果显著。通过测试不同模式下的数据传输量,可以看出分层式数据处理机制相比透传机制,数据传输量大概降低了 61%。

之所以得到这样的优化效果是因为分层式数据处理机制中多种数据传输优化策略共同作用的结果。其中,断网续传策略减少了设备断网而造成的部分数据重复上传造成的流量;批量上报策略减少了上传过程数据校验带来的流量开销;对称监控点策略减少了向异常监控点发送查询命令的流量;消息驱动机制保证了数据的定向发送,避免了数据发送给不必要的接收对象而产生的流量;重复数据过滤策略减少了设备调试时的重复下发命令发送的次数,减少了命令重发而产生的流量。

由此得出,本文设计的分层式数据处理机制可以有效节省数据通信而产生的流量费用,适合在 4G 模式下的远程运维项目中实际部署运行,可降低企业的生产成本。

3.2.2 固件远程更新测试

本次测试针对网关固件远程更新时的数据流量大小与更新的效率,分别测试 MQTT 更新方式与以 HTTP 文件下载方式下的数据流量大小与更新成功时间。通过测试结果反映何种更新方式更节省数据流量以及效率更好。

3.2.2.1 测试方案

固件更新文件大小为 356 kb，网关当前版本号为 1.0。为了测试方便，固件文件大小保持不变，版本号从 1.1~1.10 记作不同的 10 个版本。保持相同的网络环境与测试环境，一共测试 10 次，记录下每次两种方式下固件更新完成时间、消耗的数据流量以及更新成功率。固件更新时间记为开始更新时间与 OneNET 平台上记录的固件更新时间的差值，单位记作 s。通过记录 4G 网卡数据流量消耗情况来监测不同更新模式下固件更新所消耗的数据传输量。

3.2.2.2 测试结果

固件远程更新测试结果如下表 4.3 所示。

表 4.3 固件远程更新数据量测试结果

测试次数	MQTT 方式			HTTP 方式		
	时间(s)	数据量(kb)	成功率	时间(s)	数据量(kb)	成功率
1	232	1396	100%	46	462	100%
2	227	1378	100%	45	466	100%
3	225	1386	100%	43	459	100%
4	235	1408	100%	44	456	100%
5	223	1392	100%	42	464	100%
6	241	1398	100%	44	467	100%
7	236	1384	100%	43	462	100%
8	237	1388	100%	41	454	100%
9	228	1382	100%	47	453	100%
10	226	1392	100%	44	469	100%

3.2.2.3 测试分析

从表 4.3 的测试结果可以看出，两种固件更新方式的成功率几乎一致。成功率的大小取决于当前设备的网络环境，与更新的方式无关。

从网关固件更新成功的时长中可以看出，MQTT 采用的一应一答的方式消耗的时间较长，几乎为 HTTP 方式的 5 倍。在相同的成功率的情况下，HTTP 方式的效率更高。之所以相差这么大，这是因为 MQTT 方式将固件进行拆包后组成相应的数据帧后发送到 OneNET 平台以后再由 OneNET 下发到网关。网关收到帧数据后校验帧序号，校验成功后再回复服务器，服务器确认之后再发送下一个数据帧。网关将收到的数据帧中的有效数据提取后保存到本地缓存中，直到收到带有终止帧的数据。这段平台与设备之间的一应一答的时间远大于直接从服务器下载文件的时间。并且，应答机制带来的数据传输量相较于 HTTP 也更大，大概为它的 3 倍。

由此可以看出，HTTP 方式的更新效率是 MQTT 方式的 5 倍，消耗的数据流量大概为 MQTT 的三分之一左右，在固件更新的时候可选择使用 HTTP 方式的，可提高固件更新的效率以及减少流量资费。

3.3 本章小结

本章主要分析了物联网远程运维系统的数据远程传输的问题，介绍了目前的远程数据传输的现状，提出了分层式数据处理机制。从设备层、平台层、应用层分别作为出发点，根据每一层数据处理的特点来设计对应的策略去优化数据传输，详细说明了优化的原理。最后，利用实际测试结果证明了分层式数据处理机制对数据传输量的优化效果显著。测试了 http 方式的远程更新方案效率更高，同时也能降低数据传输量。

第4章 基于改进的 BP 神经网络的故障预测方案设计

故障预测主要是根据设备上报的历史状态数据，预测其未来一段时间内的变化趋势，提前预判系统可能出现故障的时间，这种方式可以留给维护人员更多的时间去应对突发事件，降低故障引起的损失。本章结合时间序列和改进后的 BP 神经网络算法，提出了结合动态时间序列的 PSO-BP 神经网络螺栓松动预警方法，通过螺栓预紧力传感器采集的数据实现设备在重大故障发生前对其进行有效预警。

4.1 故障预测

故障预测指的是提前预测到系统未来一段时间内的运行状态将发生故障，从而提高维护的效率，减少故障带来的损失。故障预测基本思路是基于系统内部结构、历史数据、系统参数等多个参考因素，然后通过某个合适的预测算法来判断系统未来的运行状态是否正常。最后，通过预测的结果做出相应的措施。

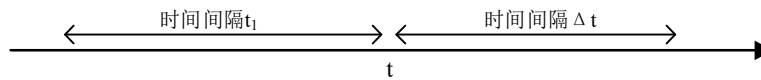


图 3.1 故障预测示意图

如图 3.1 所示， t_1 表示过去的某段时间，作为样本数据取样时间间隔。 t 代表当前时间点， Δt 表示未来要预测的一段时间，用于判断故障在这段时间内是否会发生。故障预测的工作流程可简要概括为：（1）以时间点 t 为出发点，获取并分析 t 时刻前的 t_1 时间间隔内的系统样本数据。（2）利用相应的技术手段对获取到的样本数据进行分析处理。（3）根据处理后的数据对 t 时刻之后的 Δt 时间段内的数据进行预测，再与故障限制条件比对后判断是否会有故障发生。在上述的故障预测过程中，通过修改时间间隔 t_1 的大小可以掌控系统获取的样本数据量，从而影响到故障预测的准确性。为了保证故障预测后及时响应， Δt 不能太小，否则没有足够的时间采取措施。

4.2 BP 神经网络基本原理

BP 神经网络（Back Propagation Neural Network，简称 BPNN）于 1986 年由 Rumelhart 提出的一种采取误差反向传播算法的神经网络模型^[33]，主要由输入层、隐含层、输出层三个部分组成。BP 神经网络是目前使用最广泛的前向神经网络，将其应用到螺栓松动预警系统中，它的网络结构可表示成下图 3.2 所示。

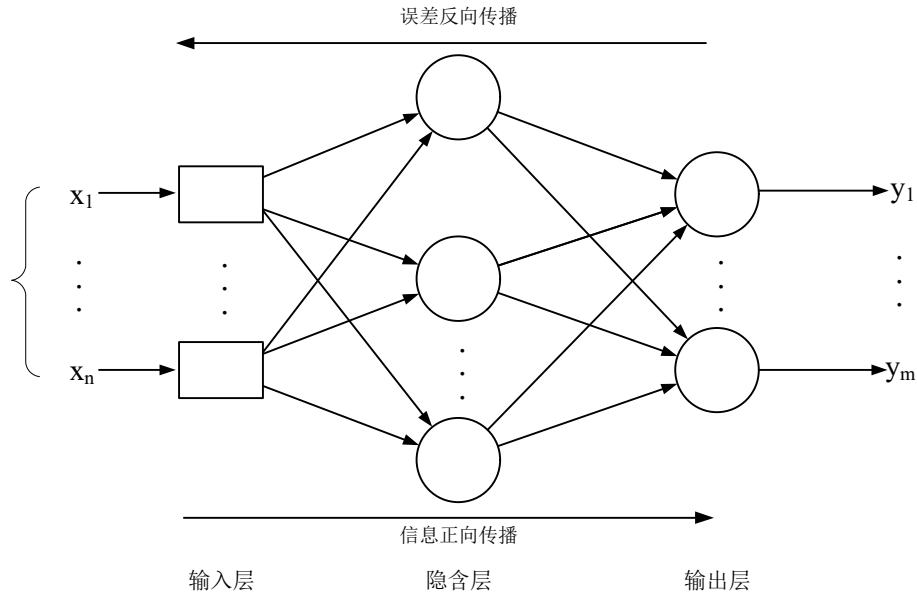


图 3.2 BP 神经网络原理图

由上图 3.2 可知，将螺栓预紧力传感器采集到的历史数据作为输入，螺栓预紧力预测值作为输出。隐含层作为衔接输入层与输出层的节点，它的个数影响了神经网络的性能。

输入层是样本数据传输到神经网络的入口。样本数据在进入网络之后经过隐含层处理后再传向输出层，最后经过输出层处理之后产生预测值。这个过程称作信号的前向传播，此时网络的各层连接权值没有发生改变。如果此时的预测值与设定的目标值间的误差过大的话，网络则会进入误差反向传播过程，将输出层的结果反向传播到隐含层。在这个反向传播的过程中，网络会对各层间的连接权值与阈值进行调整来提高结果的精确度，然后又重复执行前向传播。如此经过多次的往返迭代计算，直至达到最大迭代次数或满足误差精度时才终止训练。

总结来说，BP 神经网络通过给定的数据作为训练数据进行多次的前向传播和误差反向传播，并沿着梯度方向对网络的权值与阈值进行迭代更新，直至达到最大迭代次数或误差达到要求才终止训练。由于 BP 神经网络在调整权值与阈值时采用梯度下降算法，致使其在实际应用中呈现收敛速度慢且易陷入局部最优解的问题^[34-36]。

4.3 PSO 算法及其改进策略

4.3.1 PSO 算法基本原理

粒子群优化算法（Particle Swarm Optimization, PSO）是由 Russell Eberhart 博士和 James Kennedy 博士共同提出的一种进化算法^[37]，它通过鸟群觅食过程来学习和共享群体智能进而求得问题的最优解。PSO 算法由于参数少、易于实现而受到普遍关注，常在模糊控制、神经网络训练、函数优化等领域得到应用^[38]。

粒子群算法在优化计算的过程中，对于计算过程中所遇到的问题，每个粒子存在有对应

于此问题的局部解和全局解。在迭代更新粒子时，每个粒子的适应度值会得到调整，向期望目标的方向发展。在进行个体与社会学习之后，群体中的每个粒子都会改变其飞行的速度与所处的位置，不断靠近搜索空间的最优点。搜索过程中粒子的速度与位置的调整可由公式(3.1)和(3.2)得到。

$$v_{id} = \omega v_{id}(t) + c_1 r_1 [p_{id} - x_{id}(t)] + c_2 r_2 [p_{gd} - x_{id}(t)] \quad (3.1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3.2)$$

上式中， ω 为惯性权重，它的值越大，粒子的惯性越大，速度的变化就渐渐变慢。 p_{gd} 表示粒子在搜索空间中的最优位置，这是对所有粒子而言的，具有全局属性。 p_{id} 表示当前粒子在搜索空间中所处的最佳位置，只针对某一粒子个体而言。 c_1 为粒子的自我认知学习因子， c_2 为社会认知学习因子。 r_1, r_2 取0到1的随机数，粒子速度调整时加入随机数扰动，增加了种群的多样性。

粒子群算法的工作流程可概括为以下几步：

第一步：初始化每个粒子，形成随机粒子群。确定种群规模 N ，设定每个粒子的位置 x_{id} 和速度 v_{id} ，并规定其变化的范围。

第二步：对粒子进行评估，求出每个粒子在粒子群里的适应度值 $fit[i]$ 。

第三步：更新个体极值。将第二步中获得的适应度值 $fit[i]$ 与粒子群的个体最优解 p_{id} 进行比较，若 $fit[i]$ 小于 p_{id} ，则将 $fit[i]$ 的值赋给 p_{id} 。

第四步：更新全体极值。将第二步中获得的适应度值 $fit[i]$ 和最新的全体最优解 p_{gd} 作比较。若 $fit[i]$ 小于 p_{gd} ，则将 $fit[i]$ 的值赋给 p_{gd} 。

第五步：确定好粒子极值 p_{id} 与全体极值 p_{gd} 后再通过公式(3.1)和(3.2)将粒子的速度 v_{id} 和位置 x_{id} 重新计算后再赋予其新值。

第六步：判断适应度是否满足要求或者算法达到设定的最大的迭代次数，满足即结束搜索。否则返回第二步开始下一次的搜索过程，直至满足结束条件。

粒子群算法流程图如下图 3.3 所示。

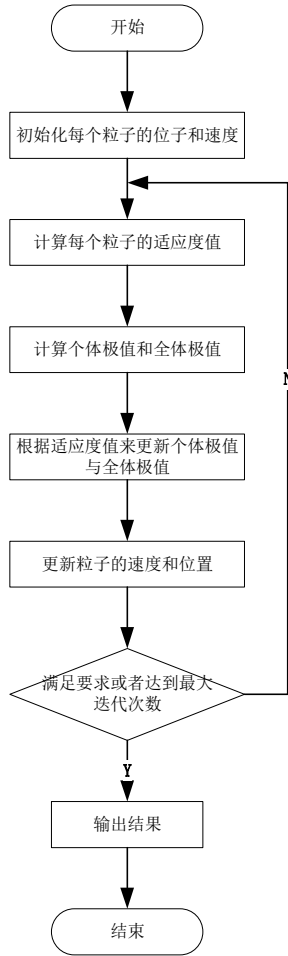


图 3.3 粒子群算法工作流程图

4.3.2 PSO 算法改进策略

传统的粒子群算法在搜索过程中易于出现局部极小值。为了将其更好地应用到实际项目中，需要对粒子群算法进行优化。本文主要是对粒子群算法的惯性权重 ω 和学习因子 c 两参数进行调整，从而达到算法优化的目的。

4.3.2.1 改进惯性权重

惯性权重 ω 是粒子群算法中的一个关键参数。通过调整 ω 的大小可以达到对粒子群算法的全局搜索能力与收敛快慢的调整。在粒子群算法运行的不同时期对粒子的搜索能力有着不同的要求，需要考虑到局部与全局两个方向上的搜索能力的发展。在算法计算的初始阶段，较大的 ω 能够保证粒子具有强悍地全局搜索能力，提高种群的多样性，并且使其能够应用在全局搜索的场景中。在算法计算的后期，较小的 ω 可以帮助其在局部进行精细探索及加快算法收敛。

当下常用的方法是使用线性递减权重的方式调节惯性权重 ω ^[39]。即在算法运行的过程中将 ω 从前期的较大值慢慢地线性递减成一个较小值^[40]。 ω 利用线性递减策略更新的表达式如下式（3.3）所示。

$$\omega = \omega_{max} - \frac{t \times (\omega_{max} - \omega_{min})}{t_{max}} \quad (3.3)$$

上式中, t 为算法当前已运行的迭代次数。 t_{max} 为算法可迭代的最大次数, 作为算法终止运行的限制条件。 ω_{max} 代表惯性权重初始设置值, 此时其取最大值。 ω_{min} 代表算法运行至最大迭代次数时的最小惯性权重。

在算法进行目标搜索的过程中, 利用式 (3.3) 调整 ω 的做法难以有效地处理粒子群算法的非线性行为, 最后导致算法搜索性能下降^[41], 表现为算法运行初期运行开销大、收敛速度较慢; 运行后期易陷入局部极小值。针对上述的算法在运行过程中出现的不足之处, 本文提出一种非线性动态调整 ω 的策略, 将惯性权重赋上随机属性。于是, 在算法计算前期, 当粒子位于搜索空间的最优点附近时, 利用随机因子调整 ω 以求得到较小的权重值, 提高算法的收敛速度。如果在迭代过程中陷入局部最优解时, 则通过随机权重在后续迭代时不断地调整粒子的飞行速度, 尝试跳出局部。 ω 随机赋值的公式如下式 (3.4) 所示。

$$\begin{cases} \omega = \mu + \sigma \times N(0,1) \\ \mu = \mu_{min} + (\mu_{max} - \mu_{min}) \times \text{rand}(0,1) \end{cases} \quad (3.4)$$

上式中, μ_{max} 、 μ 、 μ_{min} 分别为随机权重的最大值、均值、最小值。 σ 代表随机权重的方差。 $N(0,1)$ 代表期望值 0、标准差为 1 的正态分布随机数。 $\text{rand}(0,1)$ 为 0~1 的随机数, 增加了权重的随机性。

4.3.2.2 改进学习因子

学习因子 c_1 、 c_2 代表了粒子自我认知与社会认知的程度, 用于控制粒子的自我学习与群体间相互学习。

当 c_1 、 c_2 均为 0 时, 粒子无学习能力会朝着一个方向一直飞行, 飞行速度保持不变, 直至到达边界。此时粒子探索的空间有限, 找到最优解的机会不大。当 c_1 为 0 时, 此时粒子只有社会认知能力, 只能进行群体间信息共享, 粒子收敛速度较快, 易陷入局部最优。当 c_2 为 0 时, 粒子失去了社会认知能力, 只能在粒子的个体间相互分享信息。信息源减少后算法不易找到最优解。

如果学习因子选取不好的话, 便会影响粒子的寻优能力。本文设计了一种异步动态调整学习因子的策略, 当粒子的群体间共享信息不足时, 利用 c_1 指引粒子进行目标搜索。随着迭代次数的增加, 粒子的自身学习能力越来越强, c_2 指引粒子在个体间学习的同时逐渐注重全局探索。在迭代过程中, c_1 、 c_2 表现出非线性变化的特点, 表达式如下式 (3.5) 所示。

$$\begin{aligned} c_2 &= 2 \tan\left(\frac{\pi}{4} \times \frac{t}{T_{max}}\right) + 0.5 \\ c_1 &= 3 - c_2 \end{aligned} \quad (3.5)$$

上式中, t 表示目前为止的迭代次数, T_{max} 表示最大迭代次数。

由式 (3.5) 可知, 在算法计算前期, 粒子群中没有太多的信息可共享, 粒子需要发展自身。此时, c_1 取接近于 2.5 的较大值可使粒子的个体认知能力最强, 向着个体最优的方向发

展。在算法计算的中后期，每个粒子的个体经验充足。此时 c_2 取接近于 2.5 的较大值，粒子的社会认知能力强，粒子朝着社会最优的方向发展，算法易于收敛于全局最优解。其他时间 c_1 、 c_2 的大小根据公式（3.5）进行动态调整。

4.3.3 改进后算法性能测试和分析

为了验证改进后的 PSO 算法的性能，本文采用 Sphere、Rastrigin、Griewank、Schaffer 四个测试函数来对改进的算法进行评价。测试函数及相关属性如下表所示。

表 3.1 测试函数及相关属性

函数	函数表达式	自变量范围	全局最小值
Sphere	$f(x) = \sum_{i=1}^n x_i^2$	$x \in [-100,100]$	0
Rastrigin	$f(x_i) = \sum_{i=1}^D [x_i^2 - 10\cos(2\pi x_i) + 10]$	$x \in [-5.12,5.12]$	0
Griewank	$f(x_i) = \sum_{i=1}^N \frac{x_i^2}{4000} - \prod_{i=1}^N \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$x \in [-600,600]$	0
Schaffer	$f(x_1, x_2) = 0.5 + \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{[1 + 0.01(x_1^2 + x_2^2)]^2}$	$x_1, x_2 \in [-10,10]$	0

利用上述四个函数来测试改进后前后的算法性能。初始测试参数设置如下表所示。

表 3.2 改进后 PSO 算法与 PSO 算法的相关测试参数

参数	PSO	改进后的 PSO
维度	30	30
种群规模	50	50
最大迭代次数	500	500
最大速度范围	[-1,1]	[-1,1]
学习因子	$\omega_{max} = 0.9, \omega_{min} = 0.4$	$\begin{cases} \omega = \mu + \sigma \times N(0,1) \\ \mu = \mu_{min} + (\mu_{max} - \mu_{min}) \times \text{rand}(0,1) \end{cases}$
惯性权重	$c_1 = c_2 = 2$	$\begin{aligned} c_1 &= 2 \tan\left(\frac{\pi}{4} \times \frac{t}{T_{max}}\right) + 0.5 \\ c_2 &= 2.5 - c_1 \end{aligned}$

注：由于 Schaffer 函数是个二维函数，故此函数的测试维度选为 2，其余 3 个函数的测试维度选为 30。

将两种算法运行在同等测试条件下 50 次后，记录下每次的适应度值，并且计算出最大误差、最小误差以及平均误差。测试结果如下表 3.3 所示。

表 3.3 四种函数测试结果对比

测试函数	维度	算法	最小误差	最大误差	平均误差
Sphere	30	PSO	0.173440328	182.3467343	0.475686571
		改进的 PSO	0.079242368	175.4369796	0.186530893
Rastrigin	30	PSO	2.043692518	439.4528955	124.54919382
		改进的 PSO	1.656097019	412.9065467	45.90312467
Griewank	30	PSO	0.416282342	4.412573594	1.918654126
		改进的 PSO	0.003642856	4.396438213	0.429022296
Schaffer	2	PSO	0.0023653283	0.023219424	0.002643726
		改进的 PSO	0.0002453916	0.014395086	2.43E-04

从上表 3.3 的测试结果可以看出, 改进后的 PSO 算法平均误差均小于未改进时平均误差大小。以上 4 个测试函数在寻找最优解时有一定的几率会陷入局部极值的问题, 但表 3.3 的测试结果从侧面表明了改进后的 PSO 算法的适应度得到有效降低, 证明其收敛性能更好, 能减少局部极值的发生。

4.4 基于 PSO-BP 神经网络的故障预测方案设计

4.4.1 改进的 PSO 算法优化 BP 神经网络流程

利用 PSO 的全局搜索能力对 BP 神经网络的参数进行优化, 弥补了 BP 神经网络的不足。粒子群算法中的粒子适应度函数选取 BP 神经网络的输出与目标观测值之间的均方误差^[42], 误差越小代表粒子的搜索性能越好, 表达式如下式 (3.6) 所示。

$$MSE = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M (y_{ij} - t_{ij})^2 \quad (3.6)$$

上式中, N 为样本个数, M 为输出层节点个数。 y_{ij} 表示神经网络预测值, t_{ij} 表示目标观测值。

改进的粒子群算法优化 BP 神经网络的过程可分为以下几步:

第一步, 根据实际需求确定 BP 神经网络的拓扑结构及相关初始参数。确认粒子群算法的搜索空间维度。

第二步, 初始化粒子算法的相关参数及每个粒子的速度与位置。

第三步, 创建粒子与网络权值、阈值的关系, 再根据式 (3.6) 计算出每个粒子的个体及全体最优值。

第四步, 根据适应度值迭代更新粒子个体、全体最优值, 更新每个粒子的速度和位置。

第五步, 当满足要求或达到最大迭代次数时停止 PSO 算法的迭代寻优过程, 将最优的权值与阈值输入到 BP 神经网络中继续训练调优, 当满足训练目标时输出结果。

改进后的 PSO 算法优化 BP 神经网络的流程如下图 3.4 所示。

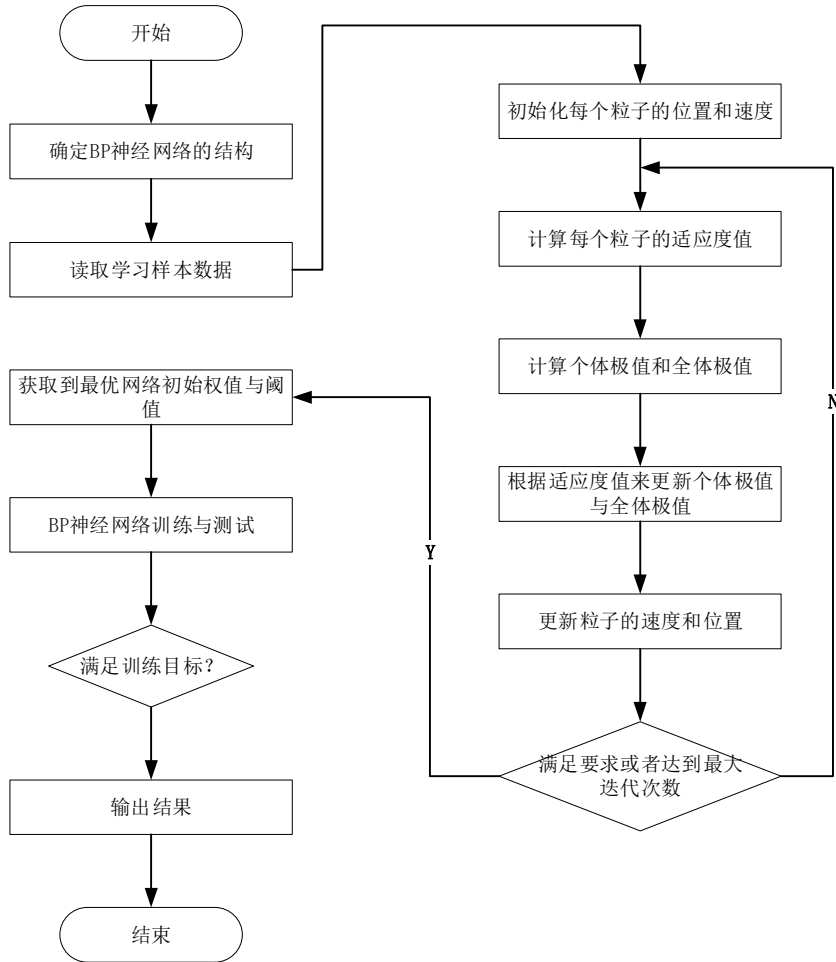


图 3.4 PSO 算法优化 BP 神经网络算法流程

4.4.2 结合动态时间序列的预测方案设计

4.4.2.1 时间序列

时间序列指的是将数据按时间先后顺序排列而形成的序列，具有非线性特性^[43]。结合本文中的实际项目，时间序列就是螺栓预紧力采集模块在不同时间上采集上报的数据所形成的序列，记为 $y_1, y_2, \dots, y_N$ 。时间序列预测就是将这段数据作为参考点，通过找到数据的内在规律，进而预测之后的 m 个值。

结合时间序列的 BP 神经网络预测模型的预测值与观测值之间的关系可表示为下式：

$$\hat{y}_{N+m} = f(y_1, y_2, \dots, y_N) \quad (3.7)$$

4.4.2.2 基于动态时间序列预测设计

时间序列预测可概括为：单步预测、多步预测、动态时间序列预测^[44-45]。

单步预测。顾名思义，一次只会得到一个预测值。结合式 (3.7)，它的基本原理是依据历史观测值组成时间序列 $y_1, y_2, \dots, y_N$ ，预测下一个值 y_{N+1} ，即式中的 m 取值为 1，预测结果为 $\hat{y}_{N+1}$ 。如果要预测 y_{N+2} 的值，则需要重新输入新的序列 $y_1, y_2, \dots, y_N, y_{N+1}$ ，继而获得预测值 $\hat{y}_{N+2}$ 。

由于其预测跨度极小的特点，使得其预测精度较高，但是对于未来一段时间内的数据，单步预测模型没法做到。

多步预测的基本思路与单步预测相同，但是它可以对时间序列之后的 m 个值进行预测，此时的 m 是大于 1 的，预测结果可表示为 $\hat{y}_{N+1}, \dots, \hat{y}_{N+m}$ 。依据时间序列中的 N 个值对之后的 m 个预测，并没有利用到之前已获得的预测值的特征。随着预测值数目的增多，观测值与预测值之间的误差越来越大，不利于实际应用。

动态时间序列预测的基本原理是首先利用时间序列 $y_1, y_2, \dots, y_N$ 进行单步预测后得到结果 $\hat{y}_{N+1}$ 。然后，将 $\hat{y}_{N+1}$ 作为样本数据输入时间序列中，并且删除掉时间序列中存在最久的一组数据 y_1 ，得到新的时间序列 $y_2, \dots, y_N, y_{N+1}$ 。最后，利用这个新生成的时间序列去预测 $\hat{y}_{N+2}$ 。不断地重复这个过程，实现预测多步的目的。动态时间序列预测删除掉了时间序列中存在时间最久的一组数据，因为其距当前时刻最为久远，数据的参考价值最低。同时，将预测结果放入时间序列中作为样本数据存在，减少了多步预测中的误差累计，预测精度更高。动态时间序列预测过程可表示如下式（3.8）所示。

$$\begin{cases} \hat{y}_{N+1} = f(y_1, y_2, \dots, y_N) \\ \hat{y}_{N+2} = f(y_2, y_3, \dots, \hat{y}_{N+1}) \\ \dots \\ \hat{y}_{N+m} = f(y_m, y_{m+1}, \dots, \hat{y}_{N+m-1}) \end{cases} \quad (3.8)$$

结合动态时间序列的故障预测原理如下图 3.5 所示。

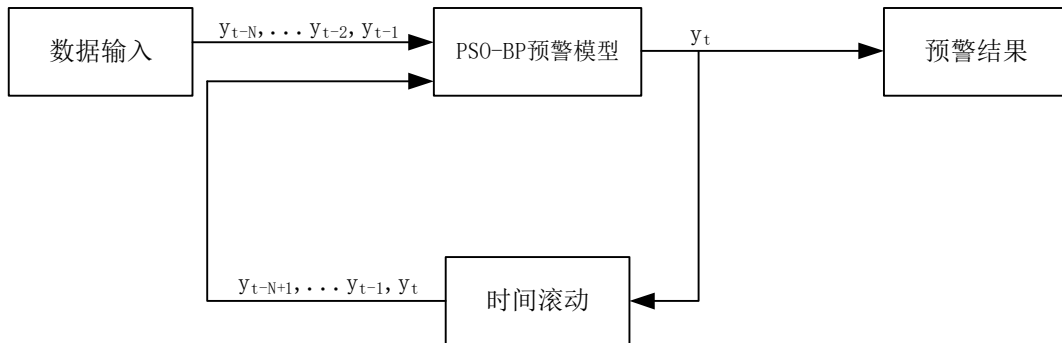


图 3.5 结合时间序列的故障预警模型

主要流程可概括为以下几步：

第一步，从数据库中获取螺栓预紧力采集模块上报的历史数据，取距离当前时间点一段时间的数据生成时间序列 $y_1, y_2, \dots, y_m$ ，将其输入到 PSO-BP 神经网络预警模型中。

第二步，确定 BP 神经网络与粒子群算法的相关初始参数。

第三步，将时间序列 $y_1, y_2, \dots, y_m$ 输入 PSO-BP 预警模型中经过训练后得到下一时刻的预测结果 $\hat{y}_{m+1}$ 。基于上一步的预测结果，再通过时间滚动得到新的时间序列 $y_2, y_3, \dots, y_m, y_{m+1}$ ，将其作为输入得到预测值 $\hat{y}_{m+2}$ 。通过不断地迭代输出，最终得到任意步的预测结果 $y_{m+1}, y_{m+2}, \dots, \hat{y}_n$ 。

第四步，将每一次的预测值与设定报警条件进行比较，判断螺栓在未来一段时间是否会发

生松动。若有问题的话，系统提醒相关人员进行维护。

4.5 测试分析

为了测试改进后的 PSO-BP 神经网络算法预测的精度。本次测试包括样本数据测试和实际数据测试。

4.5.1 测试初始参数设置

BP 神经网络的结构选择三层拓扑结构为 4-5-3，最大训练次数 2000，收敛精度 10^{-7} ，学习速率为 0.001。

根据 BP 神经的结构，利用公式 (3.9) 计算出粒子群的维度 M 为 43。

$$M = (k + 1) * m + (l + 1) * m \quad (3.9)$$

上式中， k 表示为输入层的节点数； m 表示为隐含层的节点数； l 表示输出层的节点数。则改进前后的 PSO 的测试参数设置如表 3.4 所示

表 3.4 改进的 PSO-BP 算法参数值

参数	改进的 PSO-BP 算法
维度	43
种群规模	50
最大迭代次数	500
最大速度范围	[-1,1]
学习因子	$\begin{cases} \omega = \mu + \sigma \times N(0,1) \\ \mu = \mu_{min} + (\mu_{max} - \mu_{min}) \times \text{rand}(0,1) \end{cases}$
惯性权重	$c_1 = 2 \tan\left(\frac{\pi}{4} \times \frac{t}{T_{max}}\right) + 0.5$ $c_2 = 2.5 - c_1$

4.5.2 样本数据测试

4.5.2.1 测试方案

利用系统抽样法从测试样本库中筛选出测试集与验证集，其中取 640 组设置好的测试样本数据对改进前后的 BP 神经网络进行训练，再利用 33 组样本数据对训练好的神经网络进行测试实验，验证神经网络算法的预测精度。

4.5.2.2 测试结果

将测试结果数据转成折线图，横坐标表示样本数据的编号，纵坐标表示样本数据与算法预测值的绝对误差值，两种算法的测试结果如下所示。

(1) PSO-BP 神经网络算法测试结果

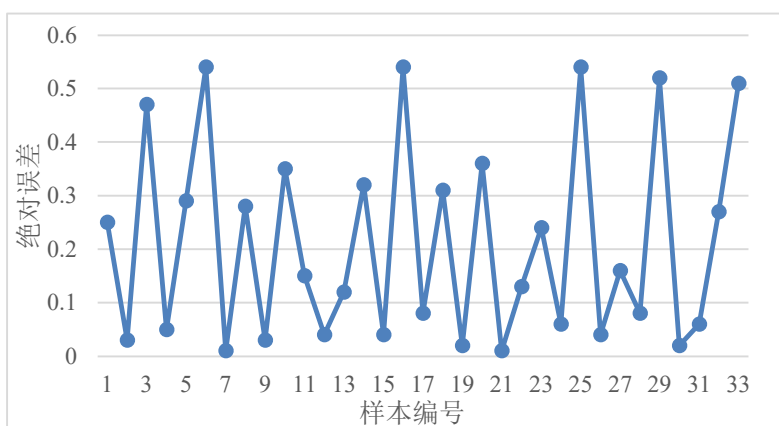


图 3.6 PSO-BP 神经网络算法预测值误差曲线

(2) 改进的 PSO-BP 神经网络算法测试结果

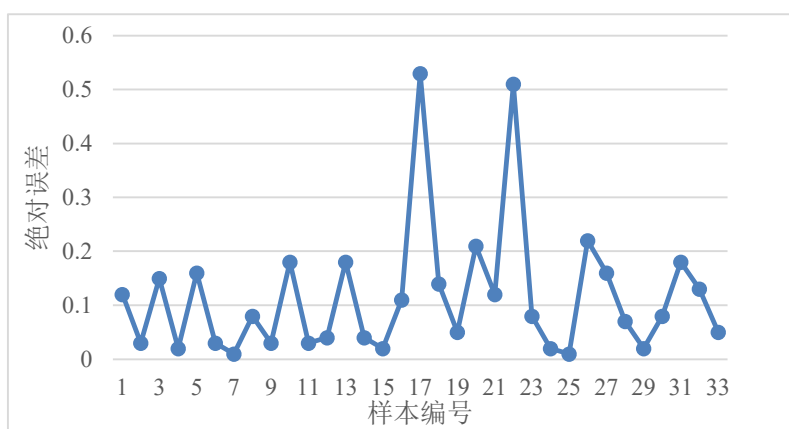


图 3.7 改进的 PSO-BP 神经网络算法预测值误差曲线

4.5.2.3 测试分析

在本文中，针对样本数据，规定当误差超过 0.5 时，认为故障预测值偏离过大，放弃此预测结果。

通过分析上面两个测试结果折线图，可以得出以下结论：

(1) 改进后的预测模型的测试结果的平均误差为 0.1155，相比于改进前的 0.2096 降低了 44.94%。

(2) 改进后的预测模型的绝对误差值大多数为 0.2 以下，而改进前的误差值相对分散，说明改进后的模型预测值更加稳定，预测误差更小。

(3) 对于 33 组样本数据，改进前的算法预测正确率达到了 84.8%，而改进后的算法的预测值正确率达到了 93.9%，比改进前提高了 9.1%。

综上所述，改进后的 PSO-BP 神经网络算法的预测正确率更高，预测误差波动较小，稳定性更高。

4.5.3 实际数据测试

4.5.3.1 测试方案

选取前 30 天的螺栓预紧力传感器采集的数据，以传感器每一小时上报的数据为一组数据，一天 24 组数据，30 天共 720 组数据。利用此 720 组数据训练神经网络，训练好之后再预测后 10 天的螺栓预紧力值。

4.5.3.2 测试结果

由于螺栓预紧力每天变化的幅度较小，所以每天只记录一次。测试数据如下表所示。

表 3.5 PSO-BP 算法改进前后螺栓预紧力预测结果对比

时间(天)	实测值	PSO-BP 算法预测值	改进后的 PSO-BP 算法预测值
31	291.543454	293.459023	292.673456
32	290.753478	288.873452	289.083436
33	289.379804	287.542312	288.069524
34	288.742371	291.214332	288.294579
35	288.153456	286.438709	287.675412
36	287.384586	285.216594	286.530123
37	286.435127	283.957521	285.970903
38	285.812351	283.326543	284.021137
39	284.438095	281.956763	283.051235
40	283.596796	280.893452	282.462109

将上表的数据转成折线图，横坐标表示第几天中午 12 点的螺栓预紧力值，纵坐标表示螺栓预紧力值，测试结果如下图 3.8 所示。

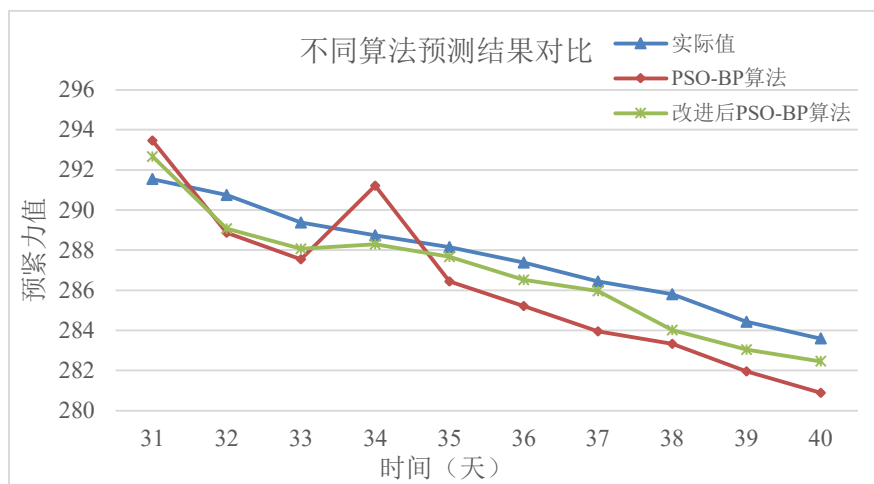


图 3.8 不同算法预测结果对比曲线

本文以 RMSE 均方根误差为标准来衡量算法的预测精度，公式如下：

$$\text{RMSE} = \sqrt{\frac{1}{M} \sum_{i=1}^M (y_i - \hat{y}_i)^2} \quad (3.10)$$

上式中 y_i 表示传感器实际采集上报的数据, $\hat{y}_i$ 表示算法预测值, M 表示预测的步数。RMSE 的值越趋近于零, 代表误差值越小^[46], 即算法预测精度越高。

分别计算改进前后的算法预测值的均方根误差, 结果如下表 3.6 所示。

表 3.6 PSO-BP 算法改进前后的 RMSE 对比

	PSO-BP 算法	改进后的 PSO-BP 算法
RMSE	2.238	1.165

4.5.3.3 测试分析

图 3.8 直观地反映了改进前后的 PSO-BP 神经网络算法的预测值与实际测量值的数据吻合情况。相比于 PSO-BP 神经网络算法的预测值, 改进后的算法预测值曲线与实际值曲线更加贴合, 并且曲线波动的幅度更小, 表明了它更加稳定。进一步, 由表 3.6 的数据可知, 改进后的 PSO-BP 算法的均方根误差相比较改进之前降低了 47.9%, 表明改进后的算法误差更小、稳定性更高。

4.6 本章小结

本章首先研究了 BP 神经网络算法原理, 指出了它的局限性, 提出利用粒子群算法优化。然后, 介绍粒子群算法原理及工作步骤, 并分析了它的优缺点。针对标准粒子群的缺点, 提出随机惯性权重法与动态调整学习因子两种策略共同优化此算法。通过四个常用的测试函数来测试改进算法的性能, 表明改进后的粒子群算法的收敛性能更好, 能有效避免局部极值的问题。最后, 经过测试表明改进的 PSO-BP 神经网络算法能有效提高故障预测的精度, 相比于改进前的均方根误差降低了 47.9%, 提高了预测精度及稳定性。

第5章 总结与展望

5.1 本文工作总结

本文针对实际项目---螺栓预紧力实时监测项目进行系统设计。基于物联网技术设计了一套螺栓预紧力实时监测系统。通过分析项目需求与技术需求，设计了系统的整体框架。利用网关实时获取设备采集的数据。智能网关以 4G/WIFI 的方式与设备云平台保持长连接，将数据上传至云平台进行处理与存储。开发了 Web 端，设计多个功能模块，便于设备管理与控制；加入故障报警通知功能，方便企业运维。同时，着重对系统性能进行了两方面的优化：为了提高预测螺栓松动时间的正确率，设计了一种神经网络预测算法，利用改进后的 PSO 算法去优化神经网络算法进而实现提高预测精准度的目的；针对远程运维时出现无效数据传输的问题，设计一种分层式数据处理机制，设计了多种策略优化数据传输，降低了传输过程的数据流量。本文做了以下工作：

(1) 分析了目前螺栓预紧力监测系统技术上的一些不足之处，并设计实现了改进后的螺栓预紧力监测系统。系统包括硬件、业务云、客户端三大部分。通过将螺栓预紧力传感器设备入网，达到设备生产数据智能采集上报的目的。通过智能网关将数据上传至云平台，进行数据存储与处理，实现螺栓预紧力的远程监测。设计了 Web 客户端，实现了对设备的远程可视化管理与监控。

(2) 针对实际项目中螺栓松动故障预警问题，基于 BP 神经网络算法创建故障预警模型。针对 BP 神经网络算法易陷入局部最优解的问题，提出以粒子群算法对其权值与阈值进行迭代优化。针对标准的粒子群算法存在的不足，利用随机的惯性权重与动态调整学习因子两种策略来优化。从测试数据中得出改进后的 PSO-BP 神经网络算法在故障预警精准度上得到了明显的提高，达到了系统的设计要求。

(3) 针对远程运维所出现无效数据传输的问题，设计了分层式数据处理机制。在每一层设计相应的策略进行无效数据检测与处理。在设备层，设计了断网续传策略，避免了因设备断网后部分数据丢失需再重传而带来的流量损耗；批量上报策略，降低了数据上报时的开销。在平台层，利用事件驱动机制避免了轮询式的数据更新方式，提高了系统的性能，减少了请求流量；利用消息驱动机制对数据进行定向发送，避免了数据的无效发送；设计对称监控点策略减少了对处于异常状态监控点的数据查询命令，从而减少了命令下发时的流量。在应用层，设计了重复数据过滤策略减少的重复数据的发送；固件远程更新策略减少了数据校验产生的数据流量。最后对系统进行数据传输流量大小测试，使用分层式数据处理机制后的数据流量相较之前整体上节省了 61%。

5.2 展望

本文设计了一套螺栓预紧力实时监测系统，解决了设备远程运维的难题，并基于历史数据去预测设备未来是否会发生故障。对故障预测精度与数据传输量进行优化，实现了螺栓预紧力监测系统的智能化升级。但由于个人经验不足，螺栓松动的原因错综复杂，本系统仍然存在不足之处，需要进一步改进和完善，具体如下：

（1）当前的在线监测系统只是为了螺栓松动监测而设计的，系统复用性、扩展性不高，无法快捷的将此系统运用到其他在线监测项目中。接下来，可以重新设计系统架构，统一硬件接口、网络协议以及后端 API 接口，结合 docker 等技术将此在线监测系统搭建成一个通用的监测平台。

（2）个人对 BP 神经网络算法与粒子群算法的理解程度与使用经验不足，只验证了改进前后 PSO-BP 神经网络算法的性能差异，缺乏与其他故障预测算法的对比，未来可以加入此部分。同时，提供基于多种算法预测功能，这样就可以根据系统所处环境的特点选择更合适的故障预测算法。

（3）随着技术的发展，监测系统的规模会越来越大，设备接入数目会越来越大，云平台要处理的数据量越来越大，压力也越来越大。此时可以加入边缘计算的功能，提高设备端数据处理的效率，同时减少云平台数据处理的压力，提高系统性能。

致 谢

此时，论文已接近尾声，我才蓦然发觉我的学生时代已接近尾声。回首研究生期间那些学习和工作的忙碌而快乐的日子，百感交集，这三年不仅增长了我的学识，也丰富了我的人生。感谢给予我帮助的那些人，也感谢在杭电的学习生活带给我美好的一切。

感谢我的导师陈科明教授。研究生期间，陈老师不仅向我传授“为学之道”，更教我“为人之本”。陈老师丰富的工程实践经验、严谨的治学之道以及勤勉的工作精神深深地影响了我。在此，谨向陈老师表示由衷的谢意。

感谢我的同窗好友俞锋、王彪、安建伟、吴新虎、司端杨、郑杭明、杜坤和王奕怡同学。他们在学习、工作和思想上都给予了我莫大的帮助和鼓励，难忘与你们一起度过的日子。

感谢父母在这三年研究生生涯中给与的理解与支持。你们在背后辛苦的付出才使得我能够一直坚持走下去。谢谢你们，我的父亲和母亲。

感谢我的女朋友汪文野，谢谢你一直在我身边鼓励我、支持我，在我写论文期间给予我坚持下去的力量。谢谢你十年的陪伴。

最后，感谢在百忙之中参加论文评阅和答辩的各位老师。感谢你们对本文的批评改正，辛苦了。

参考文献

- [1] 王群.钢结构高强度螺栓连接应用范围的探讨[J].钢结构,2007(09):76-78.
- [2] 张耀春,周绪红.钢结构设计原理[M].北京:高等教育出版社:,2010,1-2.
- [3] 刘元秀,张光蓉.俄罗斯萨彦-舒申斯克水电站事故及启示[J].四川水利,2013,34(06):86-90.
- [4] 吴键,傅少武,金伟.低功耗多点螺栓松动监测系统研究[J].电子设计工程,2015,23(09):77-80.
- [5] 王涛,李桥,邵俊华,刘绍鹏,李友荣.基于时间反演法的螺栓健康监测研究[J].科学技术与工程,2014,14(19):20-24.
- [6] 闫航瑞,曾国英,赵登峰.基于小波分析的螺栓联接结构的动态特性分析[J].机械设计与制造,2012(09):148-150.
- [7] 石耀坤. 压电阻抗法在结构健康监测中应用研究与系统实现[D].长江大学,2019.
- [8] 邵俊华. 基于压电阻抗频率变化的螺栓联接状态监测机理与方法研究[D].武汉科技大学,2018.
- [9] 陈凯. 风电塔法兰螺栓松动对结构振动影响数值分析[D].中国地震局工程力学研究所,2019.
- [10] 赵俊锋. 基于卷积神经网络的螺栓松紧状态智能监测方法研究[D].长安大学,2019.
- [11] 汪正傲. 基于螺母外侧压电材料阻抗峰值频率变化的螺栓松动监测[D].武汉科技大学,2018.
- [12] 罗志伟. 基于振动声调制技术的螺栓结构紧固状态检测方法研究[D].哈尔滨工业大学,2016.
- [13] 张义政. 基于模态分析和 RBF 神经网络的 CFRP 搭接板螺栓松动识别方法研究[D].济南大学,2020.
- [14] Vincent Caccese, Richard Mewer, Senthil S. Vel. Detection of bolt load loss in hybrid composite/metal bolted connections[J]. Engineering Structures, 2004, 26(7).
- [15] F Amerini, M Meo. Structural health monitoring of bolted joints using linear and nonlinear acoustic/ultrasound methods[J]. Structural Health Monitoring, 2011, 10(6).
- [16] 马硕,焦现炜,田柯文,吕世乐,赵阳,郑善军.故障预测技术发展分类[J].四川兵工学报,2013,34(02):92-95+116.
- [17] 唐伟峰,宋岩,张颖,茅乐.大数据下机械智能故障诊断的机遇与挑战[J].南方农机,2020,51(07):124.
- [18] 张树波. 基于阶次分析的风电轴承故障诊断[D].燕山大学,2017.

- [19]赵鹰. 矿用通风机故障预警系统的研究[D].西安科技大学,2018.
- [20]杨静,彭荣华,赵世民,钟阳万.基于 OneNET 物联网平台的车位锁控制系统[J].科技与创新,2019(08):94-95.
- [21]吴昌政. 基于前后端分离技术的 web 开发框架设计[D].南京邮电大学,2020.
- [22]Peter Mersky. Grumman F-14 Tomcat Owners' Workshop Manual, All Models 1970-2006[J].Naval Aviation News,2019,101(01):24-25.
- [23]Kasikrit Damkliang,Pichaya Tandayya. Middleware for running and debugging Taverna workflows utilising RESTful web services[J]. International Journal of Simulation and Process Modelling,2021,15(6).
- [24]丁亚争,王琬茹.基于 JDBC 的数据库连接池技术研究[J].科技信息,2009(04):500.
- [25]Ivan Merelli,Federico Fornari,Fabio Tordini,Daniele D'Agostino,Marco Aldinucci,Daniele Cesini. Exploiting Docker containers over Grid computing for a comprehensive study of chromatin conformation in different cell types[J]. Journal of Parallel and Distributed Computing,2019,134.
- [26]熊耀. RESTful Web 服务在云平台下的设计与实现[D].成都:电子科技大学,2018:31.
- [27]周虎.一种基于 JWT 认证 token 刷新机制研究[J].软件工程,2019,22(12):18-20.
- [28]刘梓良. 面向 Java Web 的 3A 安全框架研究与设计[D].西安电子科技大学,2017.
- [29]张子浩. 融合 MD5 的 RC4 混合加密算法研究[D].辽宁工程技术大学,2020.
- [30]曹震寰,蔡小孩,顾梦鹤,顾小卓,李晓伟.基于访问控制列表机制的 Android 权限管控方案[J].计算机应用,2019,39(11):3316-3322.
- [31]达庆佺. 基于访问控制的医疗数据共享隐私保护研究[D].南京邮电大学,2020.
- [32]田海晴. 基于 SpringBoot 和 Vue 框架的共享运营管理平台的设计与实现[D].山东大学,2020.
- [33]McClelland J L, Rumelhart D E, PDP Research Group. Parallel distributed processing[J].Explorations in the Microstructure of Cognition, 1986, 2: 216-271.
- [34]Timothy T. Rogers,James L. McClelland. Parallel Distributed Processing at 25: Further Explorations in the Microstructure of Cognition[J]. Cognitive Science,2014,38(6).
- [35]Zhendong Ji,Wei Liu. Open-circuit fault detection for three-phase inverter based on backpropagation neural network[J]. Neural Computing and Applications,2019,31(9).
- [36]Yanxi Zhang,Xiangdong Gao,Seiji Katayama. Weld appearance prediction with BP neural network improved by genetic algorithm during disk laser welding[J]. Journal of Manufacturing Systems,2015,34.
- [37]Kennedy J, Eberhart R. Particle swarm optimization[A]. Proceeding of IEEE international conference on neural networks[C]. Australia, 1995, 4: 1942-1948.

- [38] 吴静,罗杨.动态调整惯性权重的粒子群算法优化[J].计算机系统应用,2019,28(12):184-188.
- [39] 秦媛. 粒子群算法改进及其应用研究[D].南京邮电大学,2018.
- [40] 董红斌,李冬锦,张小平.一种动态调整惯性权重的粒子群优化算法[J].计算机科
学,2018,45(02):98-102+139.
- [41] 赵志刚,黄树运,王伟倩.基于随机惯性权重的简化粒子群优化算法[J].计算机应用研
究,2014,31(02):361-363+391.
- [42] 温荣岩. 基于改进粒子群小波神经网络的煤自燃预测系统[D].中国矿业大学,2019.
- [43] 孙友强. 时间序列数据挖掘中的维数约简与预测方法研究[D].中国科学技术大学,2014.
- [44] 王成. 基于记忆网络的时间序列多步预测算法研究[D].北京交通大学,2020.
- [45] 秦赫. 基于混沌时间序列和 RBF 神经网络的微电网电压安全评估和预测方法[D].南京邮
电大学,2020.
- [46] Barnston AG. Correspondence among the Correlation, RMSE, and Heidke Forecast Verification
Measures; Refinement of the Heidke Score[J]. Weather and Forecasting, 1992, 7(4):699-709.
- [47] 李瑾,冯献,马晨,郭美荣,贾娜. “互联网+” 农业:产业融合机理、困境与出路[J].山西农业大
学学报(社会科学版),2018,17(12):53-59.
- [48] 沈婷. 工业以太网时钟同步方法研究[D].华北电力大学,2019.
- [49] 汪小飞. 基于本体的事件驱动架构模式识别与重构方法研究[D].东南大学,2019.
- [50] 贵芳. 基于消息驱动的软件架构研究与实现[D].合肥工业大学,2017.
- [51] 魏民,王艺.物联网云平台终端远程更新技术研究与应用[J].电信科学,2018,34(10):137-142.

附录

作者在读期间发表的学术论文及参加的科研项目

参加的科研项目：

- [1] **高端海洋装备远程运维系统，横向课题
- [2] 暖通物联网平台开发，横向课题
- [3] 智能电动自行车充电平台开发，横向课题

专利：

- [1] 一种号段模式下生成分布式 ID 的方法[P]. 浙江省：CN112181306A, 2021-01-05.