**分 类 号**  **学号 M201672414**

**学校代码 1 0 4 8 7**  **密级**



**硕士学位论文**

**基于边缘计算的智能家居网关设计与实现**

**学位申请人：王静如**

**学科专业： 控制理论与控制工程**

**指导教师： 何顶新 教授**

**答辩日期： 2019年5月23日**

**Candidate : Wang Jingru**

**Major : Control Theory and Control Engineering**

**Supervisor : Professor He Dingxin**

**Huazhong University of Science & Technology**

**Wuhan 430074, P. R. China**

**May, 2019**

摘 要

近年来，随着物联网技术的蓬勃发展，越来越多的设备被接入家庭。大量物联网设备产生的非结构化数据泛滥，为了更快的响应时间和减少延迟，需要使用边缘计算技术快速处理这些数据。边缘计算处理接近数据源，可以实时处理分析数据，提高网关系统可靠性、性能。

本文从智能家居网关的需求和关键技术出发，对传统智能家居网关的问题进行分析，并提出了基于边缘计算智能家居网关的设计与实现。本文首先对智能家居网关的功能、架构等进行分析与研究，在此基础上进行智能家居系统的软硬件设计，包括网关以及网关节点。本文首先采用Node.js设计出基于三层架构模式的智能家居网关系统，整个网关系统分为感知层，模型层和视图层三层，该系统具有设备感知，远程访问以及云接入等功能。然后基于EdgeX Foundry 边缘网关框架，设计并实现面向智能家居的边缘网关系统，系统采用微服务架构并结合服务发现技术对所用微服务进行管理，整个系统包括设备微服务，核心微服务、支持微服务和应用微服务，以满足智能家居网关系统的需求。随后，搭建智能家居下的实际物理系统——智能看护系统，该系统以移动机器人为中心，配合智能穿戴设备，以及其他数十个分布式传感执行节点，共同完成系统功能，以验证智能家居网关系统的功能和性能。

论文最后采用白盒测试以及黑盒测试验证程序逻辑和功能的正确性，通过多协议协同实验，验证了网关系统兼容多种应用层协议，同时对传统网关和边缘网关进行对比分析，验证了边缘计算的性能特点。测试结果表明，该边缘网关系统能够满足智能家居的使用需求，同时响应速度更高，自治能力更强，安全性更好。

**关键词：****Node.js, 边缘计算，网关，智能家居**

Abstract

In recent years, with the rapid development of Internet of Things technology, more and more devices have been connected to the home. The proliferation of unstructured data generated by a large number of IoT devices requires edge computing technology to process these data quickly for faster response times and reduced latency. The edge computing process is close to the data source, and the analysis data can be processed in real time to improve the reliability and performance of the gateway system.

Based on the requirements and key technologies of smart home gateways, this paper analyzes the problems of traditional smart home gateways and proposes the design and implementation of intelligent home gateway based on edge computing. This paper first analyzes and studies the functions and architecture of smart home gateways. Based on this, the software and hardware design of smart home system, including gateway and gateway nodes. This paper firstly uses Node.js to design a smart home gateway system based on the three-layer architecture mode. The whole gateway system is divided into three layers: the sensing layer, the model layer and the view layer. The system has functions such as device sensing, remote access and cloud access. Then, based on the EdgeX Foundry edge gateway framework, the edge gateway system for smart home is designed and implemented. The system uses microservice architecture and service discovery technology to manage the microservices. The whole system includes device microservices, core microservices, and microservices. And application microservices to meet the needs of smart home gateway systems. Then, build the actual physical system under the smart home - intelligent care system, the system is centered on the mobile robot, with smart wearable devices, and dozens of other distributed sensing execution nodes to complete the system functions to verify the smart home The functionality and performance of the gateway system.

At the end of the paper, the white box test and the black box test verify the correctness of the logic and function of the program. Through multi-protocol collaborative experiments, it is verified that the gateway system is compatible with multiple application layer protocols, and the traditional gateway and edge gateway are compared and analyzed to verify the edge. Calculated performance characteristics. The test results show that the edge gateway system can meet the needs of smart home use, and the response speed is higher, the autonomy is stronger, and the security is better.

**Keywords: Node.js, Edge computing, IoT Gateway, Smart home**

目 录

[摘 要 I](#_Toc510807399)

[Abstract II](#_Toc510807400)

[目 录 IV](#_Toc510807401)

[1 绪论 1](#_Toc510807402)

[1.1 研究课题背景 1](#_Toc510807403)

[1.2 国内外研究及发展现状分析 2](#_Toc510807404)

[1.3 课题研究内容与研究意义 4](#_Toc510807405)

[1.4 论文结构安排 5](#_Toc510807406)

[2 智能家居网关系统研究 6](#_Toc510807407)

[2.1 传统智能家居系统架构 6](#_Toc510807408)

[2.2 传统智能家居网关系统的基本功能与问题分析 9](#_Toc510807409)

[2.3 基于边缘计算的智能家居网关系统分析 11](#_Toc510807410)

[2.4 关键技术分析 11](#_Toc510807410)

[2.5 本章小结 12](#_Toc510807411)

[3 基于三层架构模式的智能家居网关系统的设计与实现 13](#_Toc510807412)

[3.1 智能家居系统需求分析 13](#_Toc510807413)

[3.2 系统软件架构设计 18](#_Toc510807414)

[3.3 智能家居系统节点硬件设计 18](#_Toc510807414)

[3.4 本章小结 20](#_Toc510807415)

[4 基于微服务架构的边缘网关系统设计与实现 21](#_Toc510807416)

[4.1 设备微服务的设计与实现 21](#_Toc510807417)

[4.2 核心微服务 27](#_Toc510807418)

[4.3 支持微服务 27](#_Toc510807418)

[4.2 应用微服务 27](#_Toc510807418)

[4.3 本章小结 47](#_Toc510807422)

[5 实验结果与分析 48](#_Toc510807423)

[5.1 测试环境搭建 48](#_Toc510807424)

[5.2 网关系统功能测试 48](#_Toc510807424)

[5.3 本章小结 55](#_Toc510807425)

[6 全文总结与展望 57](#_Toc510807426)

[6.1 全文总结 57](#_Toc510807427)

[6.2 研究展望 58](#_Toc510807428)

[致谢 59](#_Toc510807429)

[参考文献 60](#_Toc510807430)

[附录 攻读学位期间发表论文目录 64](#_Toc510807431)

# 绪论

## 研究课题背景

本课题来源于国家自然科学基金项目“多维度智能群体网络的感知与协同优化研究”，结合智能家庭环境，设计基于边缘计算的物联网网关系统，并将该系统应用于家庭环境下多个智能终端设备的管理和控制等，目前该项目仍在进行中。

当今世界，随着云计算、无线传感技术以及嵌入式系统技术等技术的发展­­­­，物联网已经广泛应用到家庭、交通、医疗等各个领域。物联网通过将物理对象嵌入信息系统，使其在收集、分析和分发数据上取得了巨大飞跃，人们这些数据转化成信息、知识，最终转化成智慧，从而极大地改进了人们的生活、学习、工作和娱乐方式。物联网被视为继互联网和移动通信网络之后的第三次信息技术浪潮，以其巨大的市场前景受到世界各国政府的密切关注。2017年1月，我国工信部发布了《物联网发展规划（2016-2020年）》，提出了到2020年，我国要形成具有国际竞争力的物联网产业体系。

按照物联网系统的运行和信息交互方式，典型的物联网系统架构分为感知层、网络层和应用层三层。其中，物联网网关位于网络层的边缘。作为物联网系统的一个智能组件，它的基本目标是解决不同终端设备与互联网之间的异构性，加强对终端设备的管理，实现传统互联网与终端设备之间的桥梁。随着市场的发展，物联网网关的角色也在不断地扩展，例如，物联网网关提供云接入能力，使物联网系统可以利用云端的计算能力，提供更加智能化的服务；另一方面，随着嵌入式系统技术的发展，网关硬件平台可以提供更加强大的存储和计算能力。随着人们生活水平和物联网技术的发张，未来会有更多的设备被接入到物联网系统中，根据Gartner预计，到2020年，物联网设备的总规模将从200亿部攀升至500亿部。在这种情况下，大量的物联网设备会造成非结构化的数据泛滥，数据传输也会占用极大的带宽。依靠云计算技术的物联网系统已经不能满足需求：数据传输造成的延迟无法满足设备响应要求；大量的数据会给云端造成极大的存储、计算压力，相应的云端服务会带来高昂的花费等。在这样的背景下，必须依靠物联网网关过滤数泛滥化的数据，避免无用数据浪费带宽，同时利用网关设备的存储和计算能力，在网关中缓存设备数据并对数据进行处理、分析等，以此来缓解云端压力。近年来，人们对数据保护和数据安全的要求越来越高，而设备直接接入云端的方式会让设备数据暴露在网络环境中，通过网关可以在边缘处为设备提供一个智能保护层。所以，研究物联网网关系统，设计出一种改进的综合化的物联网网关系统就显得十分重要和迫切

## 国内外研究及发展现状分析

### 国外物联网网关技术发展现状

国外对物联网的研究起步比较早。1999年麻省理工自动识别中心开始研究网络射频识别（RFID)和新兴传感技术领域并设计物联网架构，同时自动识别中心的联合创始人兼执行董事 Kevin Ashton在Procter&Gamble的一个演示中提出了“物联网“这个概念。虽然物联网的概念是在1999年提出，但早在1982年卡内基梅隆大学改装的可乐自动售货机已经成为第一个接入互联网的设备，当时程序员可以通过互联网连接到冷藏设备，并在检查是否有饮料。2000年初，《卫报》、《波士顿环球报》和《科学美国人》等具有影响力的媒体开始推广”物联网“概念。同年LG宣布它是第一个”互联网冰箱“ 计划。随着到2010年年到来，物联网才开始进入实质发展时期。在物联网发展的同时，业界也开始了对物联网网关的研究。2010年，纳斯达克推出了非蜂窝网关，它具灵活的可编程性和扩展性。2013年12月，巴西博通公司推出了一款集成式的网关平台，用于简化小型蜂窝基站的部署，该平台可以方便地融合当时移动网络运营商的核心平台。2014年4月，英特尔演示了一套全新的物联网智能网关方案，该套方案由多个传感器、Atom/Quark 网关、伽利略主板以及主板内置的控制系统组成。该系统通过后端的控制系统来控制火车模型的起步/停止。2014年6月25日，北京——思科正式宣布推出全新的IR900系列工业路由器，该产品是一款面向智能互联城市无线传感网络应用的物联网网关，如图1-1所示。它是一个模块化，可编程的通用接入平台，其主要作用就是灵活的将不同标准，不同频段的无线传感器网络汇聚到IP网络中。同时该系列支持思科提出的“雾计算”架构，即能在网关进行智能计算和数据存储。

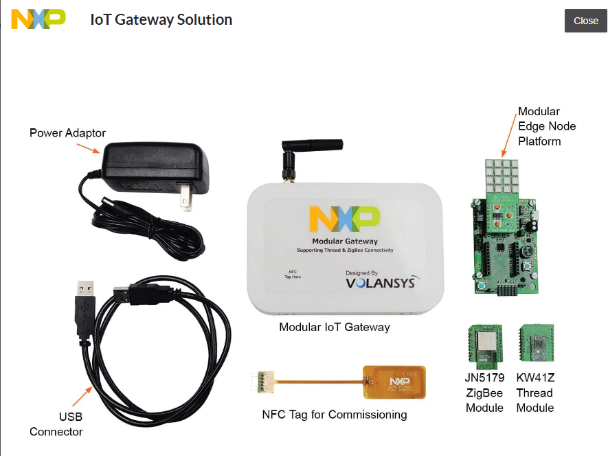
 

图1-1 思科IR900系列工业路由器 图1 -2 恩智浦物联网网关解决方案

2015年10月，亚马逊在其re:Invent开发者大会上公布了AWS IoT物联网应用云平台，该平台支持数十亿设备和数万亿条信息，可以与Amazon Kinesis，Lambda、Amazon Machine Learning、Amazon S3和Amazon DynamoDB相结合，用于物联网应用的研发，物联网基础架构管理和数据分析等。2017年3月，韩国成均馆大学的Byungseok Kang发表的文章中提出了一种基于IoTivity框架的家庭网关，该网关具有良好的可靠性，并且为用户提供配置网关的接口。2017年11月，荷兰恩智浦半导体公司提出了物联网网关解决方案(SLN-IOT-GPI)是第一个完整的开发平台，如图1-2所示，该解决方案以Modular 物联网网关为核心，汇集了安全、生产就绪型物联网系统的构件，包括硬件、软件、连接、安全和云服务。该物联网网关解决方案旨在通过预集成、全面且完全文档化的开箱即用指南和支持，加快满足IoT开发需求。

### 国内物联网网关技术发展现状

在我国，直到 2010 年中期，物联网生态系统的概念才真正发挥作用，部分原因是我国政府将物联网作为其五年计划的战略重点，不断加大支持力度，目前我国已经形成北京与天津、深圳与广州、上海与无锡、重庆与成都为核心的四大物联网产业集聚区。在顶层设计方面，2013年2月，国务院发布的《关于推进物联网有序健康发展的指导意见》，从全局性角度出发对物联网发展进行了全面系统考虑，明确指出发展目标和下一阶段发展思路。在组织机制方面，在全国范围内建立物联网发展部级联席会议制度，加强物联网相关的各部门之间的统筹协调。2010年11月，清华大学的软件与微电子学院提出的一种基于Zigbee和GPRS协议的物联网网关系统，该网关实现了传感器网络控制，协议转换和数据传输功能。2013年9月，华为设计了AR物联网网关，它采用了华为的通信系统架构VRP，具有路由，交换和安全等功能，同时支持IPV6协议。华为的AR502系列网关专门针对严酷环境设计，具有很高的抗干扰性，兼容多种串口设备和通信协议，可以广泛的应用于智慧城市、智能电网、智能楼宇等领域，如图1-4是AR502EGRc-Lc网关实物图。2014年10月，腾讯联合英特尔共同推出了软件、硬件一体化的智能家庭网关解决方案，其中由英特尔提供硬件平台，腾讯负责软件平台设计，双方合作设计的网关产品可以用户通过QQ平台连接、管理家庭中的智能节点设备。2015年1月，小米在极客公园创新大会发布了小米多功能网关，该网关可以将小米智能家居中的其他智能硬件相互串联起来，并连接到移动控制终端，从而实现远程监控。如图1-3是小米智能家庭网关实物图。

图1-3 小米智能家庭网关 图1-4 华为AR502EGRc-Lc网关

2015年5月20日，华为在网络大会（HNC2015）上，正式发布了敏捷物联解决方案，其中敏捷物联网关系列包括工业路由器、工业交换机、ICT融合网关等，作为华为物联网架构的核心，敏捷物联网关系列支持多种接口和协议，具有很强的适应性和智能性。2016年2月，华为在世界移动大会物联网峰会上正式面向全球发布了端到端NB-IoT解决方案。2016年11月30日，由华为技术有限公司、中国信息通信研究院、中国科学院沈阳自动化研究所、美国英特尔公司、英国ARM公司和软通动力信息技术有限公司联合倡议发起的边缘计算产业联盟(Edge Computing Consortium，缩写为ECC)在北京正式成立。该联盟主要致力于边缘计算的研究，搭建边缘计算产业合作平台，推动边缘计算产业健康与可持续发展。2017年10月，全球领先的嵌入式计算供应商研华科技在深圳发布了一款助力工业物联网解决方案的物联网网关产品，即无风扇宽温嵌入式工控机：UTX－3117。2018年8月，由中山大学卡内基梅隆大学国际联合研究院和中山大学电子与信息工程学院联合发表的《通过低功耗蓝牙串口启动的智能物联网关》，提出在网关中增加一个独立的蓝牙模块，允许用户使用蓝牙在近距离内对网关进行简单的管理和配置。2018年10月19日，深圳瑞兴恒方网络有限公司发布了一款面向基于LoRa技术的LPWAN网络部署的新一代物联网网关产品：RHF2S002，该网关兼顾高性能与低功耗，支持大容量节点接入，兼容全球标准LoRaWAN Class A/B/C协议。2018年11月，由上海交通大学机械与动力工程学院和农业部华南都市农业重点实验室的联合发表的《适配多终端的物联网网关设计与实现》中，提出了一种适配多种无线传感网络终端的物联网网关，该网关可以通过WiFi、以太网、GPRS多种通信模式与终端节点进行通信，同时满足多种应用场景。

综上所述，当前正是物联网网关高速发展时期，各种专用的物联网网关设备层出不穷。国内外对物联网网关的研究多在企业内部进行，物联网行业巨头如美国的亚马逊、英国的英特尔、中国的华为、小米等企业都有自己的一整套完善的物联网解决方案。但是这些公司生产的高性能的物联网网关设备价格昂贵，一般应用于工业领域，应用于家庭的网关设备，大部分功能单一，多专注于数据的采集和传输，需要与云终端或第三方应用协同工作，只有固定的接口，可移植性差且底层不能同时兼容多种通信协议，并且应用场景有限，一般企业发布的网关产品只针对自己生产的智能硬件。

## 课题目标及研究意义

智能家居作为物联网系统的一个典型应用场景，其中的网关则直接服务于家庭中的人、智能空调、智能冰箱、智能电视等家用电器，需要满足基本的数据采集和传输之外，具有足够的响应性能以及时向人发布异常信息，避免安全事故。同时还要有基本的数据保护功能，避免家庭中敏感数据泄露。最后，家庭网关需要具有普适性和亲民性，使得大部分家庭都能够支付以作为日常的家电使用。

本课题以智能家居为背景，研究智能家居系统的结构，智能家居场景下对物联网网关的基本功能，分析当前物联网网关存在的问题，针对这些问题需要进行的改进和扩展，设计能够满足当前智能家居系统需求的边缘网关系统。

本课题具有以下现实意义：

（1）提高智能家居网关的响应性能。由于传统的智能家居网关多专注于数据采集与传输，一般需要将数据发送给第三方系统进行处理，因此网关的响应性能有限，通过在网关处处理数据可以系统的响应性。

（2）提高物联网系统的安全性。通过为网关提供存储和计算能力，可以让原始数据只停留在本地局域网，从而避免了广域网环境下的网络安全问题。

（3）降低云计算带来的高昂费用。大量数据传输到云端会占用带宽，同时云端存储和处理的压力也比较大。

## 论文结构安排

本文由六个章节构成：

第一章是本文的绪论，主要阐述了本文的背景及意义，分析了物联网网关发展现状，概述了本文的结构安排和主要研究内容。

第二章是对智能家居网关系统研究，介绍物联网网关的结构和基本功能，分析传统的智能家居网关存在的问题，并且分析边缘计算应用于智能家居网关的优势，最后介绍了本文应用关键技术。

第三章基于单块架构的轻量级的智能家居网关系统的设计与实现，首先对当前智能家居环境下网关进行需求分析，结合Node.js设计了基于单块架构的智能家居网关系统，并给出了智能家居系统的节点硬件设计。

第四章是基于微服务架构的边缘网关系统详细设计。本章结合EdgeX Foundry框架，介绍边缘网关中各个服务的功能和设计方法，其中重点介绍了设备服务层和应用服务层的设计和实现。

第五章是对智能家居网关系统以及智能家居硬件节点的测试分析。结合两种不同的实现方案，使用相同的硬件节点测试三种不同的通信协议，并对系统的各个功能模赶进行测试。

第六章是全文的总结与展望，总结了智能家居网关系统所做的工作以及可以改进的地方。

# 智能家居网关系统研究

## 智能家居系统架构

物联网时代的智能家居系统是以用户住宅为平台，运用物联网，运动控制等技术，协调各个子系统，整合多个子系统信息，其主要目标是为人们提供一个集系统，服务和管理于一体的高效，舒适，安全，便捷，环保的生活环境。智能家居一般提供智能安防、智能照明、室内环境检测等服务。如图2-1是一个典型的智能家居系统的框架图。

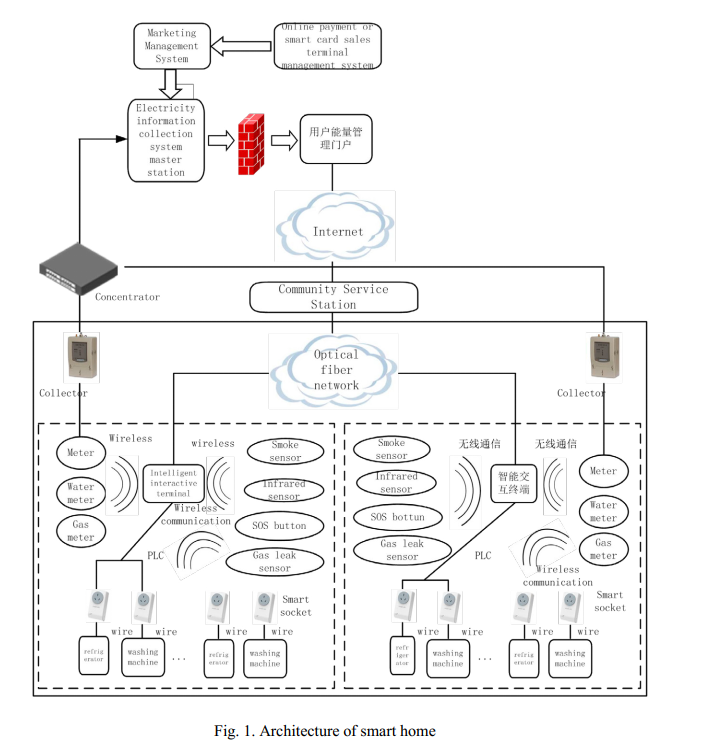


图2-1 智能家居系统框架图

家庭网关是智能家居系统的重要角色，是家庭内部信息的融合点[4]。家庭网关实现了内部局域网和外部广域网之间的协议转换。家庭网关将接收到的控制终端的指令进行解析，然后将其转换成内部网络的数据格式，最后发送给受控终端；同时网关接收受控终端发送的信息，将数据转换成统一格式之后发送给终端云或Web应用等。

### 传统智能家居网关系统存在的问题

目前家庭网关仍处于发展阶段且存在诸多问题：

1. 目前大部分家庭网关的功能单一。技术上，网关重视数据收集，并提供数据转换和传输功能。网关对数据转换之后直接发送出去，缺少本地数据存储功能，用户只能从终端云或Web应用查看历史数据。
2. 网关缺乏数据挖掘与智能分析功能。智能家居环境下，一些智能家电如室温检测器、等会周期性地产生大量数据，对这些数据进行传输和终端云存储成本极高，而且传输的数据中只有部分数据具有实际价值。而目前网关缺乏预处理和过滤数据的功能，以减少传输、处理和存储的成本。
3. 网关延迟高。时间对于一些物联网应用来说至关重要，数据从本地网关发送到终端云分析，然后发送指令到底层节点，最后节点根据指令采取行动，这一过程花费的时间无法满足对时间有严格要求的场景，例如，当传感器检测到婴幼儿靠近家庭中的危险区域时，应该立即触发报警机制提醒父母，同时采取措施规避危险。如果在网关中处理数据，可以减少从发出数据到执行处理的时间。
4. 网关安全性亟待提高。物联网节点直接介入互联网时，很容易被恶意攻击者入侵。当节点通过网关接入互联网时，虽然网关为节点提供了一层屏障，但此时网关会成为恶意攻击者的唯一对象，因此提高网关的安全性、可靠性至关重要。目前虽然行业内投入了大量的资金和力量，安全问题仍然存在。
5. 网关价格普遍高，无法普及。目前物联网网关多用于工业领域，而用于家庭的物联网网关的价格高昂，无法普及到普通家庭用户。
6. 物联网行业内针对家庭网关没有形成一套统一的标准。由于物联网行业性太强，目前市场上有多种物联网网关，但各种物联网网关的公众性和公用行不足。例如，小米的智能家庭网关只能应有到小米公司自己生产的智能终端设备。

## 基于边缘计算的家庭网关系统分析

### 边缘计算

边缘计算产业联盟（ECC）发布的《White Paper of Edge Computing Consortium》中，对边缘计算给出了这样的定义：边缘计算实在靠近物或者数据源的网络边缘，融合网络、计算、存储和应用等核心功能以实现边缘智能化的分布式开放平台。边缘计算能够满足当前工业数字化对快速连接、实时业务、数据优化、智能应用以及安全与隐私保护等多方面的要求。

边缘计算不同于云计算，通过在边缘提供计算能力可以在几毫秒内处理、分析和执行收集到的数据，因此可以极大的提高物联网系统的性能。例如，在智能视频监控系统，现场边缘设备可以试试处理分析画面，只有检测到异常场景时，才会将出现异常的视频片段发送到云端，从而降低通信带宽要求并缩短数据传输时间。此外，通过减少视频数据量，不仅可以降低云端的存储成本还可以减轻云端的计算成本，更易于云端管理数据。

边缘计算为数据保护和数据安全提供了实现途径。当前公共云为了实现私人访问和控制，创建了一系列与分类或敏感数据相关的规则、机制等，但实施的过程繁琐，而且代价大，难以管理。边缘计算可以利用本地数据库缓存数据，让敏感数据只停留在边缘局域网中，从而避免了公共网络环境下的各种威胁。

边缘计算可以降低整个物联网系统的运营成本。边缘计算在通过在边缘处处理数据，可以过滤掉大量无用数据，从而缓解当前物联网数据泛滥的现状，另一方面可以减少云端数据存储；通过消除向云端传输的千兆字节的数据以避免昂贵的带宽；通过自己专用网络分析敏感的数据从而避免购买昂贵的云端服务。

### 基于边缘计算的家庭网关

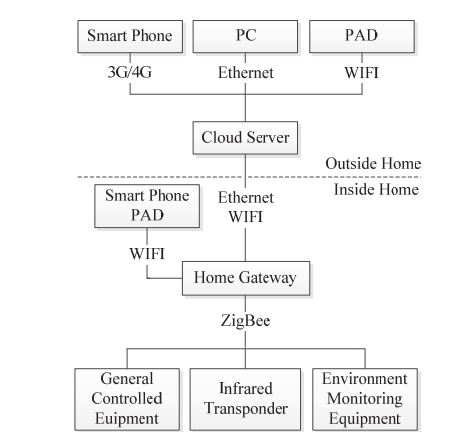
随着嵌入式系统技术的发展，嵌入式设备的存储能力和CPU的处理能力已经完全能够满足边缘计算的要求。

家庭设备管理：利用数据库对设备信息进行存储，包括设备添加，设备更新，设备删除等。用户在通过移动终端、PC、Web浏览器等可以随时查看当前家庭中有哪些设备。

设备数据管理：利用数据库缓存设备数据。例如将老人或小孩的健康监测系统接入到网关中，网关可以对体温、心率等数据进行存储，当老人或小孩生病时，可以将这些数据下载下来，以供医生参考。同时可以在网关中定义关键事件处理机制，当检测到体温、心率异常时，可以采集到的异常数据发送到终端，并触发报警机制。

多设备互联互通：所有的家庭设备在网关处集合，网关可以根据一个设备的数据去向另一个设备发送指令。比如，当检测到智能门锁打开时，同时发送指令到智能灯光设备。

设备数据的深度分析：利用数据库存储的历史数据，可以采用一些优秀的算法对数据分析，获取数据中隐藏信息。比如，利用机器学习算法分析设备的性能，预测设备的寿命等。



## 关键技术分析

### EdgeX Foundry

1. 微服务软件架构

在软件架构体系中，微服务是一种相对较新的术语。目前在业界对微服务还没有一个统一的严格的定义。不过，ThoughtWorks的首席科学家——Martin Foowler认为微服务架构是一种提倡将单一应用划分成一组微小的服务，这些服务之间通过相互协调、配合为用户提供一种价值的架构模式”。【1】

微服务架构中每个服务的开发独立进行，并且在独立的进程中运行。服务之间通过轻量级的通信机制进行通信，一般是REST HTTP协议[5]。微服务架构中的每个服务可以单独部署。此外，对这些服务的管理也是通过一个单独的服务实现，例如EdgeX Foundry框架中使用注册配置服务，实现对其他服务的注册、配置和管理。每一个服务开发时可以根据需要进行技术选型，例如编程语言和数据模型等。

相对于微服务架构，整体架构在其内部也可以实现多个服务，但是它必须作为一个整体进行统一部署，例如使用Java编程语言开发的Web应用最后可以打包成一个War包进行部署。当应用程序较小时，采用整体架构开发起来更方便，随着应用程序的增加，程序的更新和扩展会比较困难，也很难再添加新的开发人员或替换团队成员，所以，系统运维成本会比较高[12]。所以对于大型的物联网系统，使用整体架构不是最优的方案[7]。

1. EdgeX Foundry框架

EdgeX Foundry是Linux基金组织于2017年4月启动的开源项目。其重要目标是为物联网边缘计算提供一个标准化的、互操作的开源框架。EdgeX Foundry的一个显著特点是该框架的设计采用了微服务架构，根据物联网系统应用到资源，如设备和传感器、数据等，整个框架分成多个松耦合、开源的微服务，同时根据物联网中数据流向从“南向”设备到“北向”应用，将多个微服务划分成四个服务层提供支撑服务和两个基础系统服务。如图是EdgeX Foundry架构图：

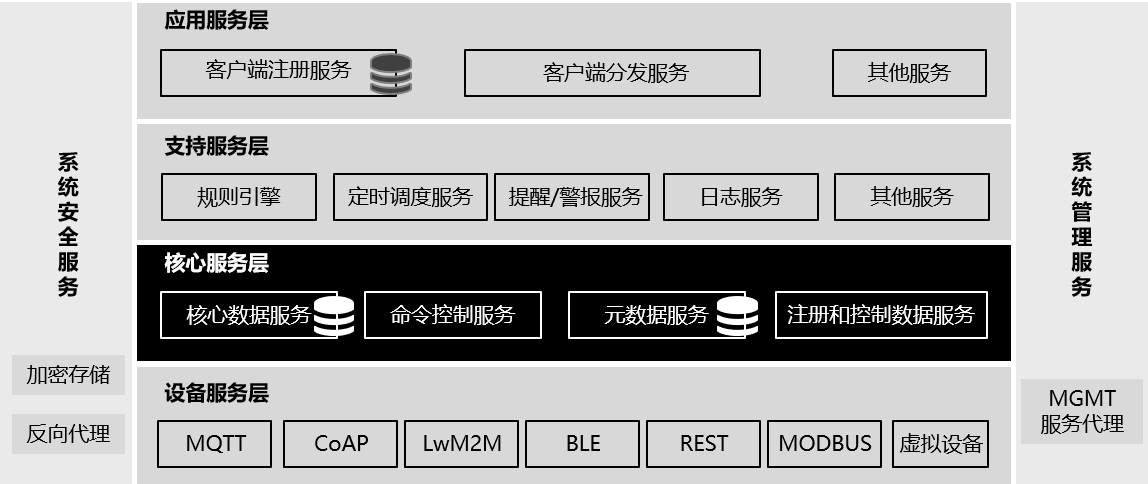


图1-1 EdgeX Foundry微服务框架

EdgeX Foundry框架的设计原则之一是做到了平台无关。通过采用微服务架构，基于EdgeX Foundry设计物联网系统或子系统时，可以根据硬件平台和操作系统来选择编程语言、数据库等；同时，该系统既可以部署在物联网数据产生的边缘，也可以部署到网关、云端和其他的数据中心；开发者可以在EdgeX Foundry的设备服务层设计和实现不同的传输协议接口，使开发的系统具有广泛的接入能力，即做到协议无关。

EdgeX Foundry整个框架非常灵活，框架中每个服务可以随时被更新、替换或扩展，也可以根据底层设备功能进行扩展或缩小。

EdgeX Foundry作为一个应用于边缘的物联网框架，其充分体现了边缘计算的优势。基于EdgeX Foundry设计物联网边缘系统可以有效解决数据传输延迟问题；由于EdgeX Foundry框架设计数据库接口，通过将数据存储到边缘系统，可以不用考虑大量数据传输所需要的带宽；考虑到用户友好性，EdgeX Foundry在框架中设计了开放服务层，开发者可以设计和实现Web应用、PC应用、移动终端应用接入到开放服务层，从而实现对系统的远程操作。

EdgeX Foundry的两个基础系统服务分别安全服务和管理服务。其中安全服务支持一流的工业级安全性，启动安全服务之后，系统通过初始化安全组件为安全服务设置运行环境，然后安全服务就可以管理用户的访问控制以及其它服务创建的资源访问JWT(JSON Web令牌)。其中有两个安全组件：安全存储组件，用于为系统中的敏感数据提供安全的存储环境；另一个是API网关，它可以通过反向代理限制外部的客户端对系统内部资源访问和控制。系统管理服务提供了一系列的管理工具，可以监控系统中所有为服务、固件、操作系统等的安装、升级、启动和停止。

EdgeX Foundry主要采用Vault实现数据安全存储。Vault是一种安全访问机密信息的工具。机密信息指要严格控制访问权限的任何内容，例如API密钥，密码，证书等。Vault为所有机密信息提供统一的界面，同时提供严格的访问控制并记录详细的审计日志。

安全机密存储：任意密钥/值秘密可以存储在Vault中。Vault会在将这些机密写入持久存储之前加密这些机密，因此获取对原始存储的访问权限不足以访问您的机密。Vault可以写入磁盘，Consul等。

动态秘密：Vault可以按需为某些系统生成机密，例如AWS或SQL数据库。例如，当应用程序需要访问S3存储桶时，它会要求Vault提供凭据，Vault将根据需要生成具有有效权限的AWS密钥对。创建这些动态机密后，Vault也会在租约到期后自动撤消它们。

数据加密：Vault可以加密和解密数据而无需存储数据。这允许安全团队定义加密参数，并允许开发人员将加密数据存储在SQL等位置，而无需设计自己的加密方法。

租赁和续订：Vault中的所有机密都有与之相关的租约。在租约结束时，Vault将自动撤销该秘密。客户可以通过内置续订API续订租约。

撤销：Vault内置了对秘密撤销的支持。保险柜不仅可以撤销单个秘密，还可以撤销秘密树，例如特定用户读取的所有秘密，或特定类型的所有秘密。撤销有助于关键滚动以及在入侵情况下锁定系统。

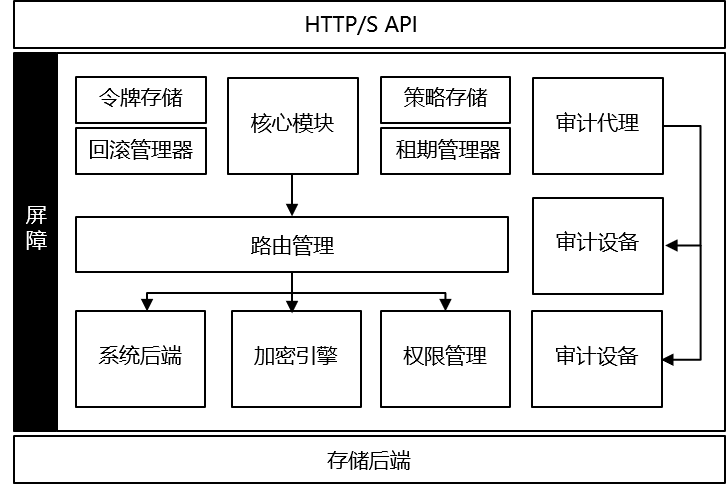


图2 -2 Vault架构图

如图xx-xx,是Vault的结构示意图。在Vault体系中建立了一层安全屏障，安全组件位于屏障内部，非安全组件存储后端和HTTP/S API位于屏障外部。存储后端用于对加密数据进行持久化处理，保证Vault重新启动时可以访问之前保存的数据。HTTP/S API提供客户端与Vault交互的接口。这两个组件在Vault中是不被信任的。Vault启动完成之后，整个服务处于密封状态，在对Vault访问时，需要使用密钥对其开封。

### 物联网应用协议CoAP、LwM2M和MQTT

1. CoAP协议

CoAP协议，全称是Constrained Application Protocol。CoAP协议是一种专门为低功耗的受限设备设计的网络传输协议。在物联网中，端点设备的的资源都是有限的，CPU的计算能力有限，ROM和RAM的大小有限，网络带宽有限等，针对这样的应用场景，CoAP协议借鉴HTTP协议，对其进行简化和压缩，实现了物联网设备之间的一种轻量级的通信机制。

如图xx-xx是CoAP协议的抽象模型。

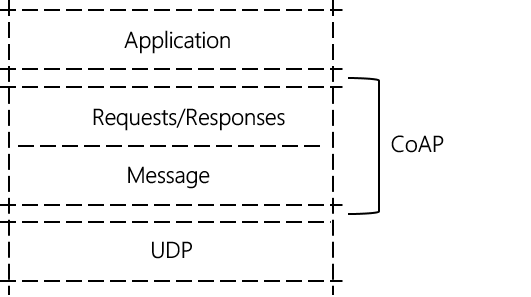


图2 -2 CoAP协议抽象层

默认情况下，CoAP协议通过UDP传输消息，一个CoAP消息占用一个UDP数据报的数据部分。其中CoAP消息被编码成二进制格式。消息以固定的4个字节为开头，接下来是一个可变长度的令牌值，在令牌之后是CoAP选项的序列。

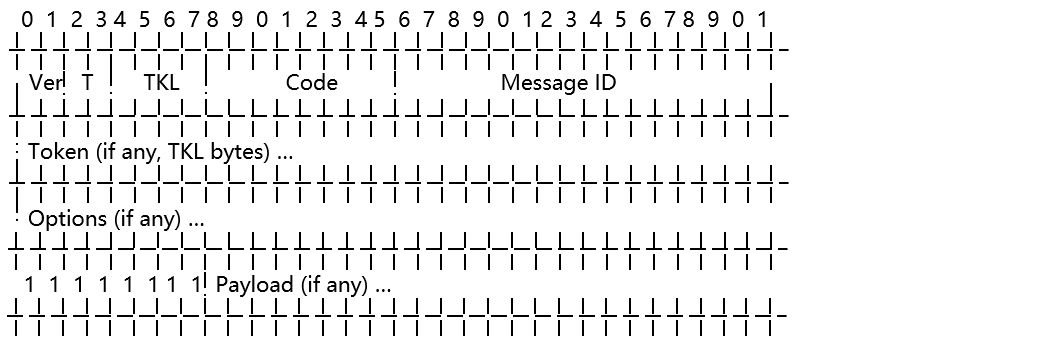


图2 -2 CoAP协议消息模型

CoAP定义了四种消息类型：Confirmable可确认的, Non-confirmable, Acknowledgement, Reset。通过将消息标记为可确认的（CON）可以保证数据的可靠传输。如果消息超时，即在一定的时间内没有得到响应，发送方会重新发送消息，直到接收方返回一个具有相同ID的消息响应（ACK）；当接收者无法处理消息时，即不能提供一个合适的错误响应时，他会回复一个重置消息（RST）。

CoAP消息中携带请求响应语义，包括方法码或响应码。默认情况下请求和响应信息如URI或有效负载作为消息的可选项。令牌用于独立于底层消息匹配请求的响应。

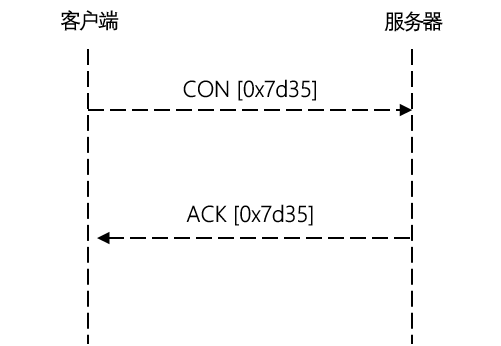
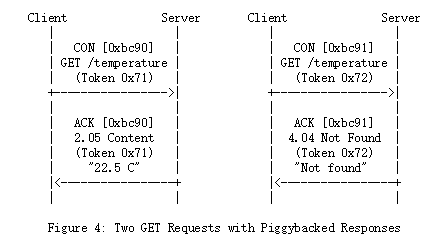


图2 -2 CoAP协议可靠消息传输

当消息不需要可靠传输时，可以将消息标记为NON作为不可确认消息发送。同样的，当消息接收方无法处理消息时，会返回一个重置消息（RST）。



请求在可确认或不可确认消息中携带，如果消息能够被处理，接收方会立即返回确认消息，其中携带对请求的响应。

1. LwM2M协议

LwM2M协议是由OMA SpecWorks于2017年2月15日定义，专门针对受限的物联网设备的独特需求。LwM2M是一种安全、高效且可部署的客户端/服务器协议，用于管理各种资源受限的设备。

LwM2M的协议栈如图xx-xx

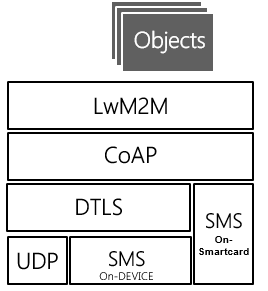


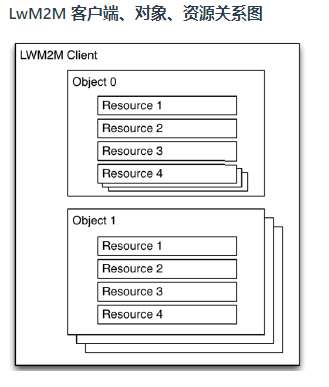
图2 -2 lwM2M协议栈

LwM2M协议承载在CoAP协议之上。作为应用层协议，CoAP协议负责数据的传输，LwM2M主要负责资源定义。

LwM2M协议定义了三个实体：LwM2M Server、 LwM2M Client和Bootstrap Server。其中LwM2M Server一般部署在网络服务供应商处，作为工作服务器，LwM2M Client一般部署在物联网的节点设备上，Bootstrap Server是引导服务器，可以用于引导客户端连接到对应的工作服务器。

在这三个实体之间，LwM2M协议定义了四个接口：引导接口为客户端注册到对应的工作服务器上提供必要的信息；客户端注册接口用于实现客户端和工作服务器之间建立连接，同时将客户端的信息存储在服务器上；设备管理和服务使能接口，用于工作服务器与客户端的信息交互，其中主控方是工作服务器；信息上报接口允许工作服务器订阅客户端的资源信息，订阅成功之后，客户端会按照约定的规则向工作服务器报告资源状况。

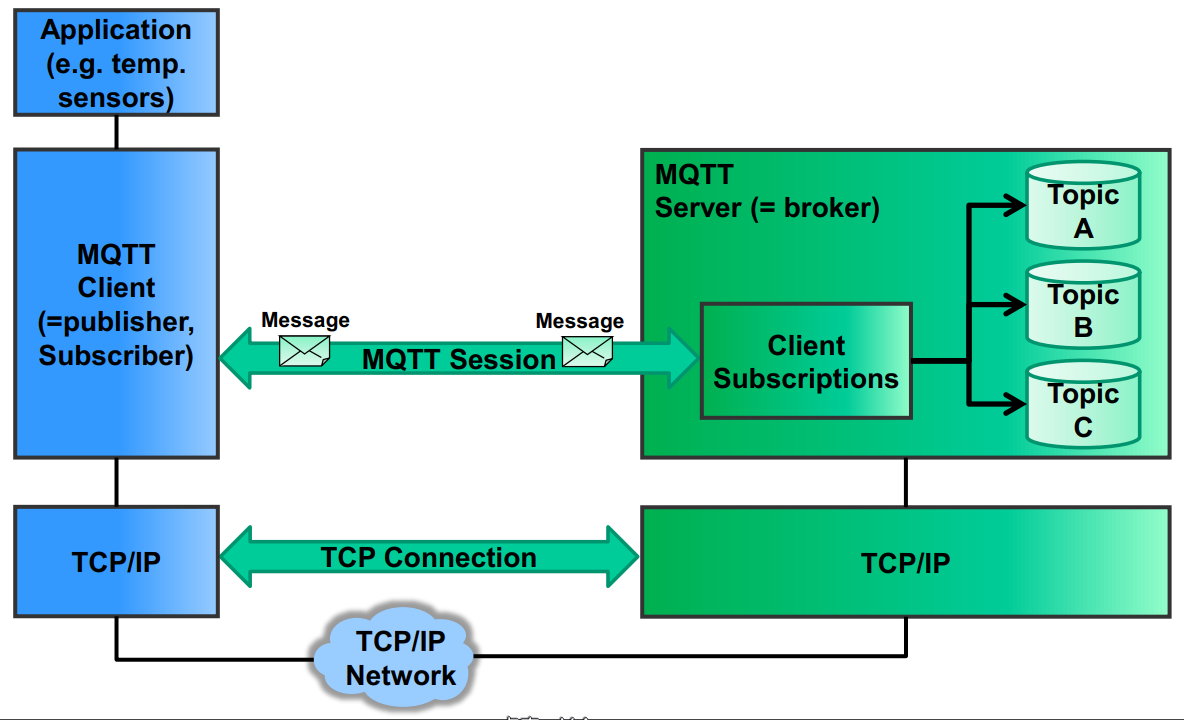
在LwM2M协议中定义了7个标准对象，其中对象是一组资源的集合，如物联网节点设备对象，它包含设备URL，设备中传感器和执行器、设备传感器的值和执行器状态等，同时LwM2M协议为不同的对象及其资源定义了唯一的ID。LwM2M协议也定义了资源模型，其中每个客户端可以有多个对象，对象中又可以有多个资源。



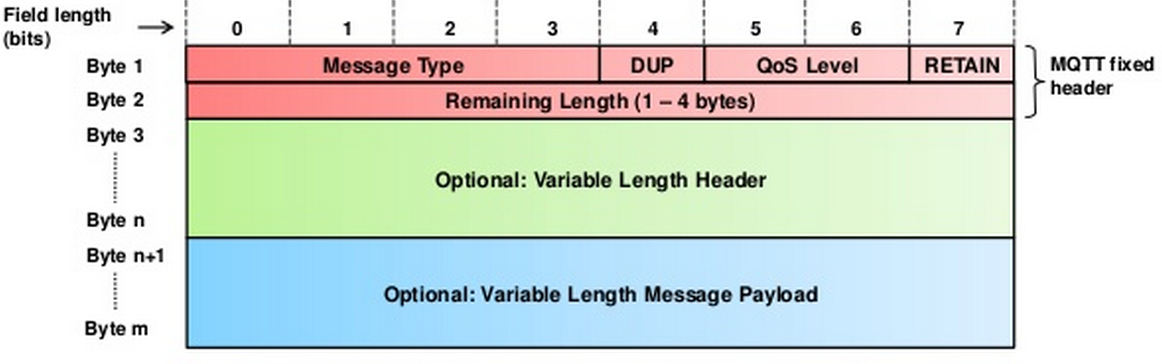
1. MQTT

MQTT协议是由IBM的Andy Stanford-Clark博士与Arcom（现在是Eurotech）的Arlen Nipper于1999年定义的一种通信协议。MQTT是一种基于发布/订阅模式的轻量级的消息队列传输协议，该协议专为受限设备和低带宽、高延迟或者不可靠的网络设计。与CoAP和LwM2M协议相同，MQTT协议也是一种基于客户端/服务器协议不同的是该协议默认构建于TCP/IP协议上。

在MQTT协议中定义了三种身份：消息发布者（Publish）、消息代理者（Broker）（服务器）和消息订阅者（Subscribe）。其中，消息的发布者和订阅者都是MQTT客户端，消息代理者是服务器，消息发布者也可以同时是订阅者。



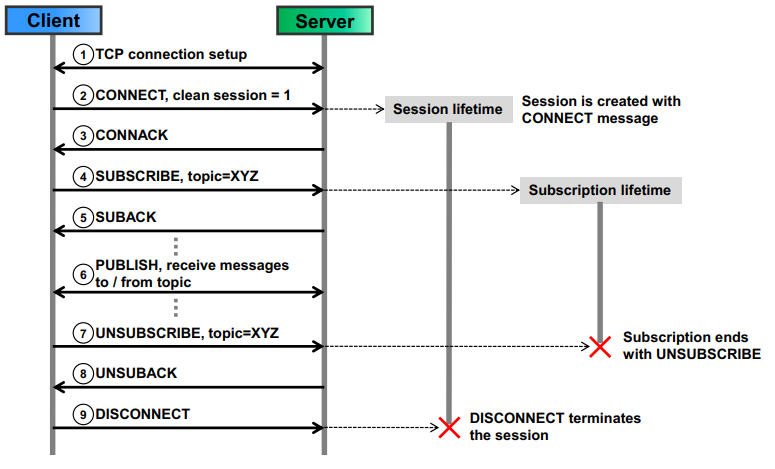
MQTT协议通过交换控制报文来进行通信。控制报文主要用于连接管理，订阅管理，消息传输和身份验证。MQTT的控制报文包含三部分：固定报头、可变报头和有效负载。其中可变报头和有效负载为可选的，主要目的是尽可能地减少传输的数据量，特别是在带宽受限或不稳定网络中。

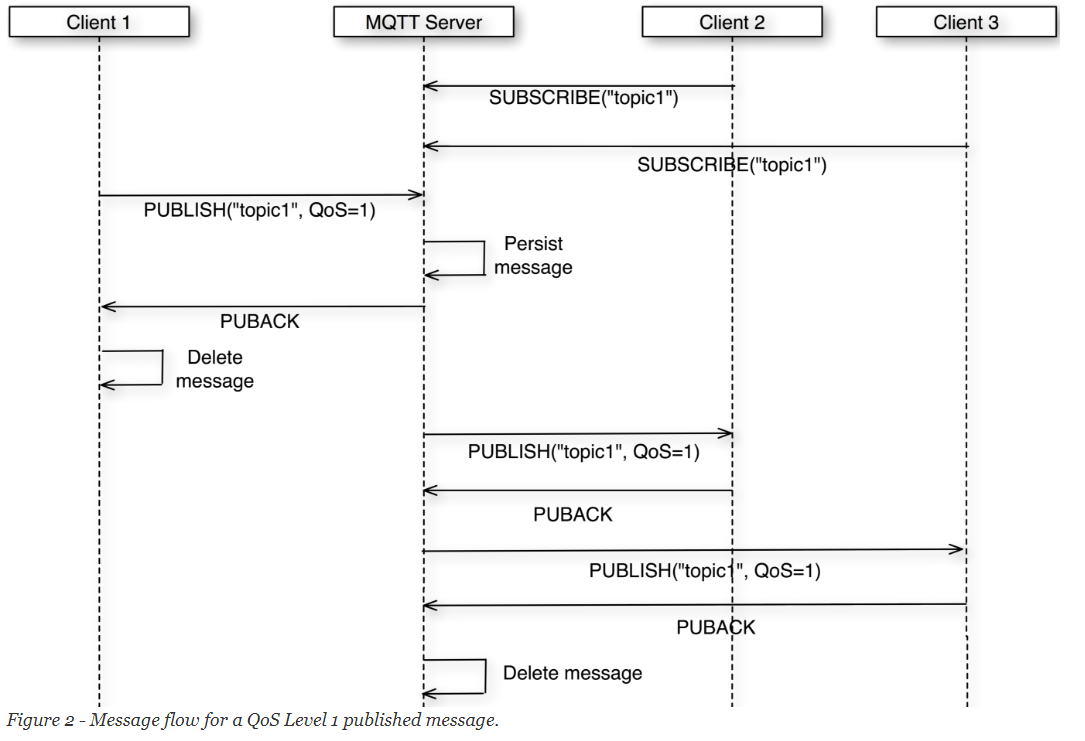


固定报头用于描述控制报文。固定报头分为三部分，前面四个字节用于指定控制报文的类型，在MQTT协议中定义了16中控制报文类型。DUP为信息重传标志，用于指明接收者已经收到信息。QoS用于指定服务质量，用于保证消息发送到接收者。RETAIN标志用于指示服务器或代理保留最新接收到的信息，当有新的订阅者时发送给订阅者。固定报头剩余字节用于指出除固定报头之外控制报文剩余的字节长度。

可变报头对于控制报文来说可选内容。可变报头的内容根据控制报文的类型不同会存在差异。在客户端或服务器发布的消息中，可变包头会包含主题名（Topic）用于识别有效负载要发送的信息通道。

有效负载携带MQTT客户端与服务器之间通信内容。有效负载的数据格式是固定的。





## 本章小结

本章主要记述了对智能家居网关的研究。首先介绍了智能家居系统的结构，分析网关在智能家居系统中的重要作用，提出当前智能家居网关存在的一些问题，同时结合这些问题分析基于边缘计算的智能家庭网关的优点。在本章最后介绍了本文智能家庭网关实现过程中所使用的几项关键技术，包括EdgeX Foudry开源框架，CoAP、LwM2M、MQTT几种物联网中常用的应用层协议。

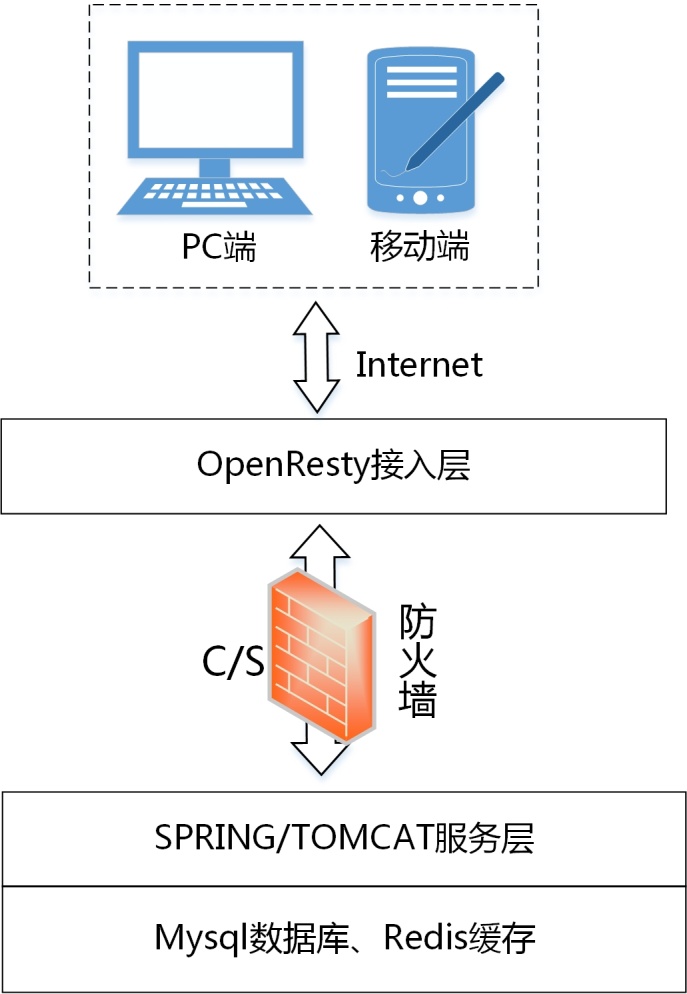
# 基于单块架构模式的智能家居网关系统的设计与实现

结合第二章对智能家居网关的功能分析，本章给出了一种轻量级的智能家居网关系统的设计和实现方案，并给出了硬件节点的设计方案。

## 网关软件总体设计

### 网关架构设计

采用三层架构模式的设计思维将网关从“南向”物理节点与“北向”应用将系统划分为感知层，模型层，视图层三层，如图xx-xx所示，在不同的层次中定义不同职责的部件。三层架构模式降低了层与层之间的调用复杂度，同时简化了层次之间依赖关系[]，每个层更新不会对其他层造成很大的影响。



模型层，是整个网关的核心，网关的所有逻辑功能在模型层实现。其主要功能有：负责对上传数据和下达指令进行解析，并实现两者之间的相互转换，当设备通过感知层发送请求时，模型层将解析请求并转换成统一的JSON格式，从而向上屏蔽底层通讯协议的异构性。当模型层接收到视图层发送的指令时，模型层会根据接收者对指令进行编码；为了加快数据读取速度，模型层采用Redis缓存节点数据，同时使用MySQL实现对部分关键数据的持久化存储；模型层利用MySQL数据库对设备基本信息包括设备名称、设备通讯协议、设备资源等进行存储，实现对节点的统一控制和管理；模型层实现了MQTT接口，通过该接口可以将网关接入到云端，利用云计算为节点提供更多服务；为了实现与视图层的数据交互，在模型层加入HTTP、WebSocket接口。

感知层，是网关系统的接口。感知层具有广泛的接入能力，即能兼容CoAP、LwM2M、MQTT多种通信协议以实现与不同物联网设备的交互。所有的设备必须先通过感知层注册到网关，注册成功之后设备才可以继续发送数据到网关。

视图层，即用户交互层。模型层将处理后的节点数据发送到视图层，通过本地局域网，用户注册、登陆到视图层，查看节点的运行状况和实时数据，同时可以查看设备信息和历史数据，发送控制命令等，实现对节点的管理。模型层

图3 网关工作流程图

流程图

1. 初始化网关，包括感知层各个服务端模块，模型层数据缓冲区。
2. 节点发送注册请求到感知层，感知层将请求转发到模型层进行解析并转换成统一格式，模型层将节点信息缓存到Redis数据库，然后同步到Mysql数据库。模型层同时转换后的数据发送到感知层，一方面通过WebSocket服务端推送消息到感知层，用户使用移动终端、PC等接入与网关相同局域网，通过网页访问指定端口，就可以看到节点已经接入，同时在节点管理界面可以查看节点信息。模型层同时使用MQTT客户端发布消息到云端，当用户处于广域网时，视图层使用MQTT客户端订阅消息并实时显示数据。
3. 节点注册成功以后，发送数据到感知层，感知层将数据转发到模型层进行解析并转换成统一格式。首先访问Redis查看模型层是否有该节点，如果不存在则访问Mysql数据库是否有该节点，如果两个数据库都没有，模型层调用感知层相应的模块发送错误信息到节点，如果节点存在，模型层将数据发送到视图层显示。
4. 用户点击网页，视图层解析点击事件转换成指令，根据用户所处的网络环境，视图层调用WebSocket客户端向服务器端发送指令或使用MQTT客户端发布带有该条指令信息的消息到云端，模型层接收到该指令后对其进行解析，然后调用感知层对应模块将指令下发到节点。
5. 在模型层内设置定时任务，该任务读取Redis数据库缓存的节点数据然后进行处理，如开关，指示灯等只有表示表示状态的数据直接设置过期并删除，体温、心率等连续变化的数据，求平均值之后存储到Mysql数据库，然后Redis缓存的数据设置过期并删除。

### 网关编程语言JavaScript

Node.js[2]是一个基于 Chrome V8 引擎的JavaScript 运行环境，被称为是运行在服务端的JavaScript。目前，JavaScript是世界上最流行的脚本语言之一，主要用于构建Web应用[3]。 随着JavaScript的普及，开发人员不再满足于仅仅将JavaScript应用于前端开发，并在其它领域寻找JavaScript的发展之地，如三星使用JavaScript开发了物联网应用平台IoT.js和轻量级引擎JerryScript，上海南潮信息科技有限公司为硬件开发设计的JavaScript运行时Ruff等。本章设计的网关是使用Node.js编程语言实现的一种轻量级网关。使用Node.js的原因如下：

1. JavaScript是一种事件驱动的编程语言。

作为一种事件驱动的编程语言，JavaScript非常适用于事件驱动的应用程序。事件驱动的应用程序可以执行大量的任务，添加一个新任务不必等待其他任务完成。这有助于实时响应时间，并行处理多个任务，并允许多个物联网节点响应同一个事件。而物联网中数据的显著特点是海量、周期性强[9]，使用Node.js设计网关可以满足节点大量持续的并发访问请求，同时有助于网关设备节省电池电量。

1. JavaScript有完善的内存管理机制。

JavaScript有一个内存垃圾收集器，它不需要显式地释放内存。网关硬件一般是资源受限的嵌入式设备，内存管理对设备性能有很大影响。JavaScript可以自动释放未使用的内存，避免内存泄漏。因此，开发人员可以专注于开发的其他方面。

1. JavaScript拥有一整套完善的工具、库和插件

随着JavaScript在各种开发领域中的应用增加，JavaScript有许多开发资源可以使用。针对物联网开发，JavaScript有多种开发框架和轻量级引擎，如CycloneJS， IoT.js，JerryScript，Duktape等。

1. 提供了兼容JavaScript的硬件解决方案

针对多种物联网中的硬件，JavaScript提供了不同的解决方案。本文所使用的树莓派 3B支持Node.js，因此可以充分利用JavaScript优点的同时，使用NPM包管理工具简化开发过程。

1. 安全的执行环境

Node.js的VM模块利用V8引擎的Virtual Machine contexts动态地编译和执行代码，同时执行代码的上下文与当前进程隔离，从而提供一种安全的沙箱环境。

## 模型层

### 数据库设计

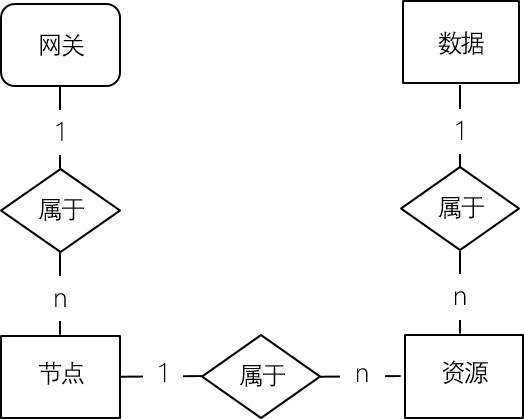
对网关中所有节点及其资源的特征进行抽象，设计对应的数据模型是模型层开发的重要一环。一个好的数据模型，不仅可以提高系统开发的效率，还有利于系统后续的扩展，并影响整个系统的性能和可靠性。

数据库设计一般分为六个阶段：首先对系统进行需求分析，结合系统需求分析要处理的对象，收集支持系统稳定运行的数据并确定数据之间的约束关系；然后将需要的内容抽象为实体信息，即概念模型设计；其次，进行逻辑设计，将概念模型转换成相应的数据模型；物理设计则对各个属性的类型、范围、约束条件等进行具体定义；将前面几步设计的数据模型转换成具体的表，然后进行测试，测试通过之后数据库就可以投入使用。

本家庭网关的模型层采用Mysql对节点及其数据进行持久化存储，数据库表的也是基于Mysql建立的。Mysql是一个性能和可靠性都相当高的数据库管理系统。[17]

本文采用E-R图对网关进行概念模型的建模[14]。E-R图又称实体-联系图(Entity Relationship Diagram)，是针对数据库表概念设计的建模方式，可以很好的表示体统中实体类型、属性和关系。如图xx-xx是围绕网关节点管理所产生的概念模型。网关节点管理主要包括：

1. 节点管理，即对智能家居环境中智能家电、可穿戴设备等所有接入网关的节点的管理
2. 节点资源管理，节点拥有资源包括温度传感器、开关等。
3. 数据管理，即对节点资源产生的数据进行管理。



节点管理概念模型图

由E-R图可以看出网关中实体之间的关系均为一对多，这种情况下可以直接用两张数据库表表示。数据库表根据E-R图进行设计[20]

网关管理主要围绕节点信息表展开，节点信息表用于记录各个节点的基本信，查询节点所属网关、节点的资源、节点数据等都依赖于节点信息表，其存储结构如表xx-xx所示。

表4-1 节点基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **类型** | **NULL** | **默认值** | **描述** |
| device\_id | int(11) | N | 自增 | 节点ID，主键 |
| device\_name | varchar(50) | N | 空字符串 | 节点名称 |
| create\_time | datetime | N | 当前系统时间 | 节点注册时间 |
| ip\_addr | varchar(50) | N | 空字符串 | 节点IP地址 |
| port | int(11) | N | 0 | 节点端口号 |
| gateway\_id | int(11) | N | 0 | 网关ID |

表4-2 节点资源基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **类型** | **NULL** | **默认值** | **描述** |
| resource\_id | int(11) | N | 自增 | 资源ID，主键 |
| device\_id | int(11) | N | 0 | 节点名称 |
| resource\_name | varchar(50) | N | 空字符串 | 资源名称 |
| create\_time | datetime | N | 当前系统时间 | 资源添加时间 |

表4-2 数据基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **类型** | **NULL** | **默认值** | **描述** |
| data\_id | int(11) | N | 自增 | 数据ID，主键 |
| resource\_id | int(11) | N | 0 | 所属资源ID |
| value | int(50) | N | 0 | 数据值 |
| create\_time | datetime | N | 当前系统时间 | 资源添加时间 |

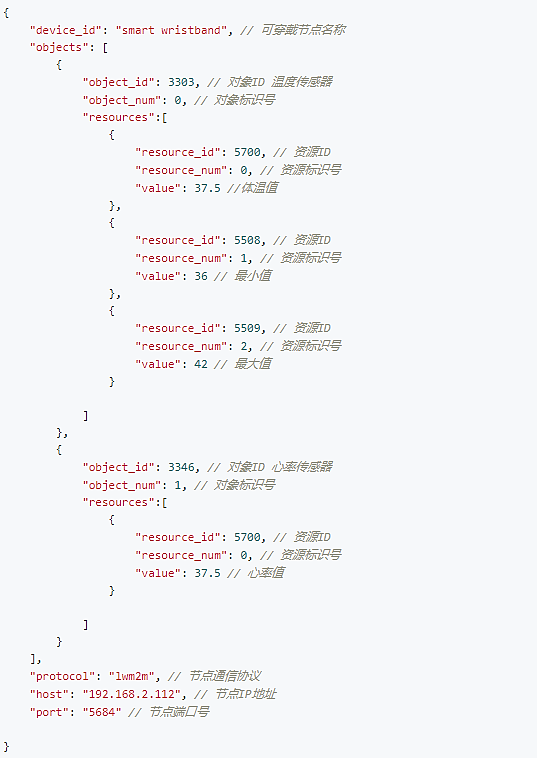
完成上述数据库表的设计之后，编写对应的SQL脚本，然后运行脚本以创建和初始化数据库，测试和调试之后将数据库应用到该网关中。

### 数据转换与消息推送

1）数据转换

本网关的无线节点分别使用CoAP协议、LwM2M协议和MQTT协议三种应用层协议传输数据，模型层在接收到节点的数据之后，需要进行数据转换将数据转换成统一的格式。

模型层参考LwM2M协议中对对象和资源的定义标准，设计了节点数据结构体。如图xx-xx所示，该结构体代表一个可穿戴节点，该结构体包含节点名称、节点协议、节点IP地址和端口号，这些数据结合在一起可以指向唯一的节点。节点数据结构体中还包含节点的对象实例，即节点上的传感器、执行器等，一个对象实例包含一个或多个资源。为了避免同一个节点上包含多个相同的对象，使用作为对象唯一标识。同理，使用作为同一个对象中资源的唯一标识。



为了将所有节点发送的数据转换成上述格式，在CoAP节点中，统一使用如图xx-xx所示的格式作为请求路径。

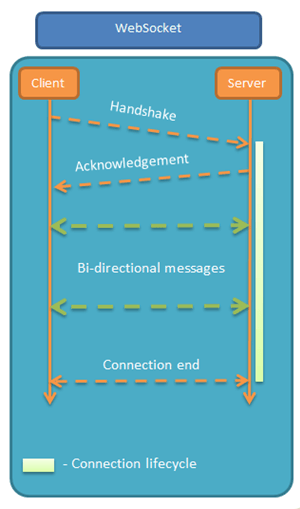


同理，MQTT节点在发布或订阅数据时，统一按照上图设置消息主题Topic。

节点数据被转换之后，一方面数据被缓存到Redis数据库，通过设置定时任务把部分数据同步到Mysql数据库；另一方面数据被实时推送到视图层展示给用户。

2）消息推送

消息推送是目前Web应用和安卓应用中常用的一种技术，指服务端以主动的方式将信息发送到客户端。目前常用的推送技术有长连接、客户端轮询、苹果推送通知服务APNs、Android 推送服务C2DM[21]。目前实现消息推送时有两个关键点：保证链接的可靠性，在使用TCP协议保持的链接有可能因为网络因素断开，而服务器端和客户端无法感知链接是否断开；保证消息推送的可靠性，即保证服务端推送的消息能够达到客户端。本系统模型层采用WebSocket协议向Web应用推送节点数据，Websocket是HTML5为推送业务专门实现的一种轻量级的传输协议，该协议建立在TCP协议之上。首先，WebSocket借助HTTP协议完成握手，在服务器和浏览器之间建立一条全双工的通信信道。然后服务端就可以向客户端推送消息了。如图xx-xx所示：



## 感知层和视图层设计

### 感知层

感知层分别根据CoAP、LwM2M和MQTT协议标准设计了三个模块，分别负责与三种节点进行通信。如图xx-xx所示，一次指令的下发的工作流程：用户在网页触发指令发送，视图层通过特定的协议将数据发送到模型层，模型层根据定义的节点数据结构解析指令。感知层根据解析后的数据判断节点类型，调用对应模块将指令发送到对应的节点。

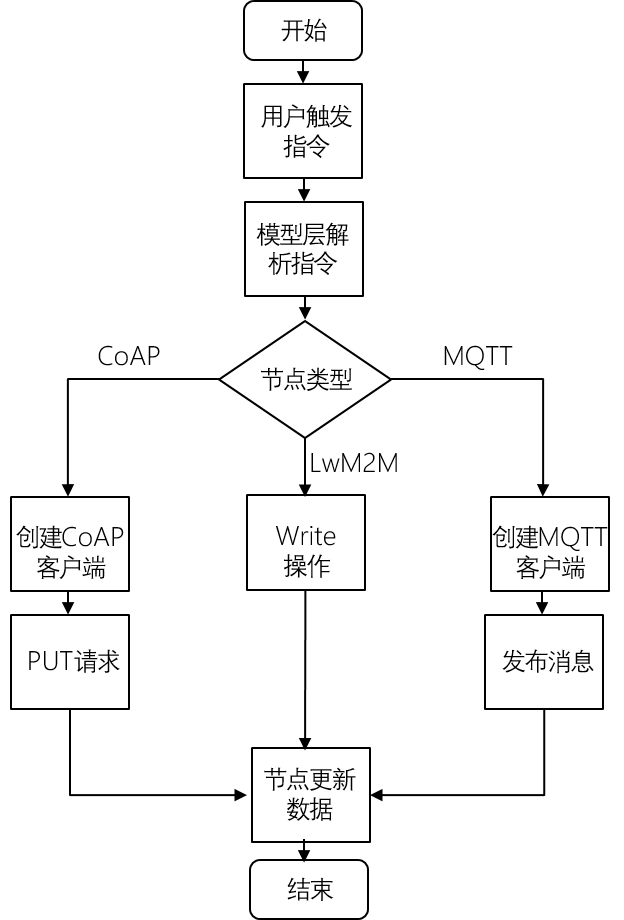


图3-1 感知层下发指令流程图

### 视图层设计

视图层主要包括网关节点管理和网关节点数据实时显示两部分。

1. 网关节点管理

网关节点管理主要用于管理网管中所有的节点，可以实现单个节点的查找、删除。其中，用户没有权限添加节点和修改节点信息，只有当节点通过感知层注册到网关之后，模型层会将节点信息添加到数据库中。当用户在网页中删除一个节点后，节点的资源、节点资源的数据也一起被删除，同时Redis中缓存的相关数据被设置过期并删除，如果相应底层节点向网关发送数据，网关检测到数据库没有对应节点，会向用户返回错误信息。如图xx-xx所示，【Action】表示用户权限，点击其中一个节点，可以跳转节点信息页面，该页面不仅展示节点的基本信息，同时展示节点资源列表，即节点包含的传感器、执行器等。同理，用户可以点击查看资源的信息、历史数据等。

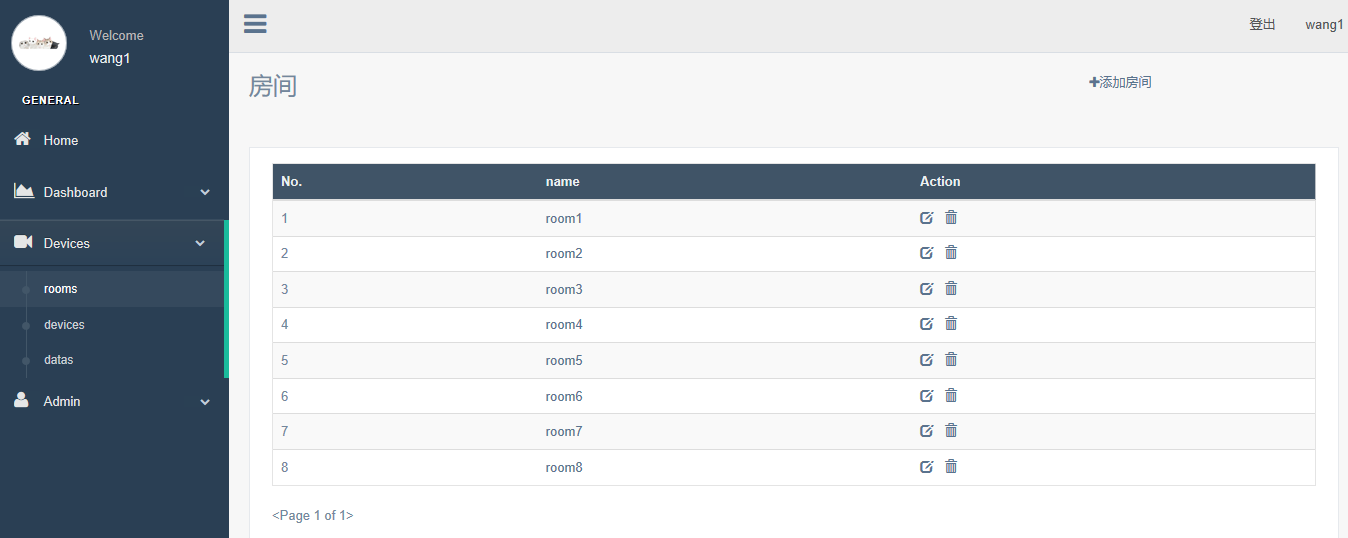


图3-1 Freeboard

1. 实时数据展示

实时数据是模型层Websocket服务器推送的数据，前端基于Freeboard框架设计。

Freeboard一个轻量级的单页的Web应用框架，该框架提供了一种插件架构，其中数据源插件用于获取数据，窗口部件用于表示数据。如图xx-xx所示，图中A是数据源列表，通过点击【ADD】添加数据源，点击删除图标可以删除数据源，与数据源相关的窗口部件会一起删除。点击B处可以添加窗口部件，C是数据显示区域，其中使用了不同的窗口部件显示不同的数据。通过编写JavaScript脚本实现Freeboard的插件，每一个插件都封装在闭包[]中。

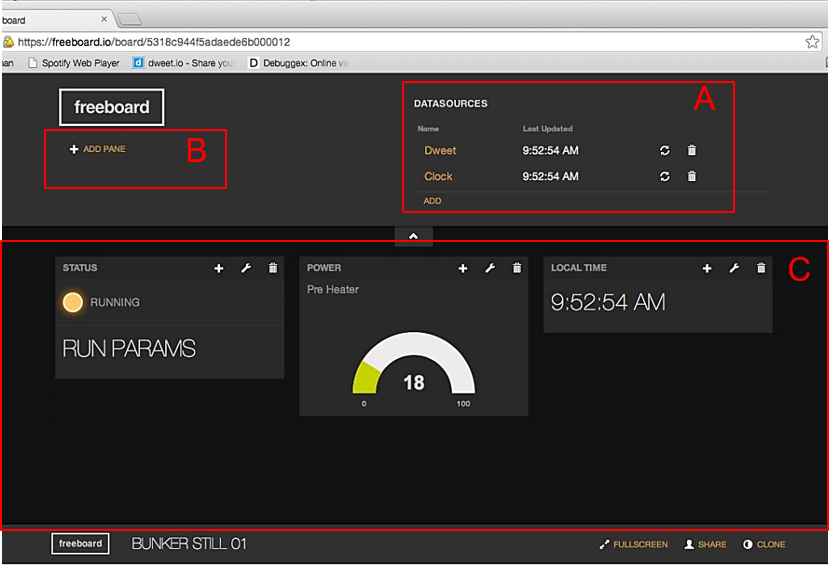
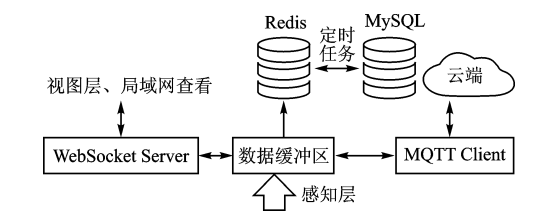


图3-1 Freeboard

本文根据 Freeboard定义的插件规范，设计了Websocket数据源插件和MQTT插件。Websocket数据源插件本质上是一个Websocket的客户端，接收到WebSocket服务器推送的数据后，该插件可以解析数据，并根据数据类型调用不同的窗口部件展示数据。同理，MQTT数据源插件是一个MQTT客户端，该客户端通过订阅消息，以获取云端或其他第三方应用发布的数据。用户可以在网页中修改数据源信息，如WebSocket服务的IP地址、端口号等。

本文所设计的节点的数据包括两类：反应状态布尔量和连续的整形数据。因此，在视图层设计了三类窗口部件：可以交互的开关、显示状态的指示灯，显示连续变化数据的图表。其中图表插件是基于Highcharts图表库设计。

视图层会根据所处的网络环境选择数据源，当用户与网关处于同一局域网环境时，视图层直接通过WebSocket插件获取数据并实时显示；当处于广域网环境下时，视图层通过MQTT协议从云端获取数据，此时静态网页部署在Github上。



## 智能家居系统节点硬件设计

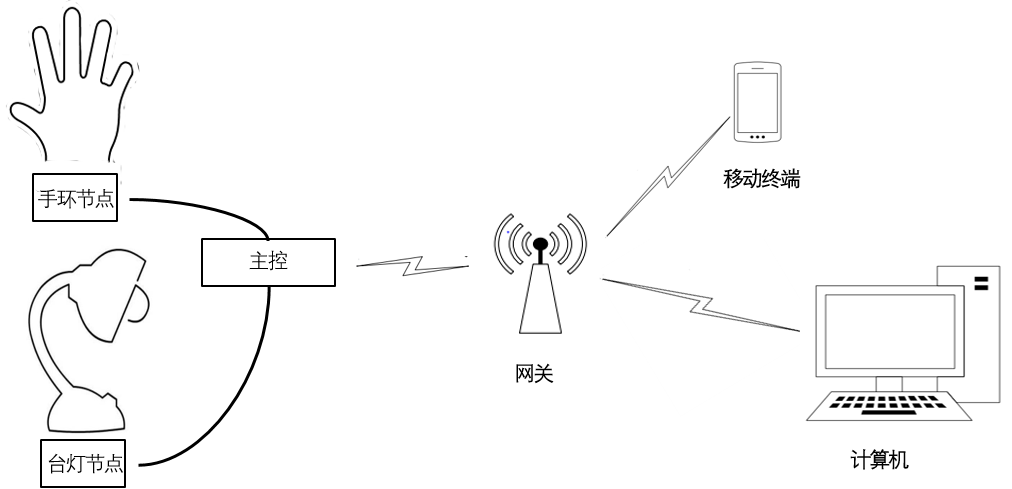


图1-1 智能家居系统框架图

如图上图所示：系统硬件节点是一个可穿戴节点和一个台灯，可穿戴节点是一个用于监测老人或婴幼儿睡眠状况的智能手环，能够实时监测体温、心率和睡眠状况，台灯节点是智能家庭环境下智能家电的代表。可穿戴节点和台灯分别通过不同的通讯协议发送数据到网关，用户通过访问Web应用查看设备数据并控制设备状态。

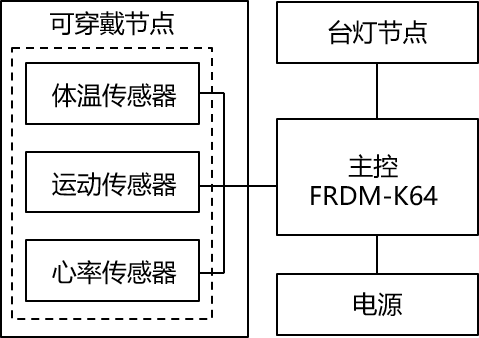


图1-1 智能家居节点硬件框架图

### 可穿戴节点芯片选型

可穿戴节点可以作为一个智能手环或脚环，为了不影响正常的运动，可穿戴节点需要尽可能缩小体积和重量，同时硬件成本要尽可能低。本文设计的可穿戴节点集成了MLX90614体温传感器、MAX30102心率传感器、MPU6050三轴加速度计和电源管理模块。

本文采用MLX90614作为体温测量的原件，MLX90614是一款高性能、高精度的非接触式红外温度传感器，且测量速度快、效率高，目前主要应用于医疗、工业等领域。MLX90614的测量范围是，可以用于测量人体温度，且测量精度可达[13]，能够满足国家食品药品监督管理局提出的《医用电子体温计(GB/T 21416-2008)》对医用电子体温计的性能要求。

引脚功能

表 xx—xx MPU6050引脚图

|  |  |
| --- | --- |
| **引脚** | **功能** |
| SCL / Vz | 用于2线通信协议的串行时钟输入。内置5.7V稳压二极管，用于外界双极型二极管，使MLX90614Axx系列使用于8~16V的供电 |
| SDA / PWM | 数字输入输出。正常情况下，该引脚可用于读取PWM制式的温度值。在SMBus兼容模式下，此引脚自动配置为开楼NMOS |
| VDD | 电源线 |
| VSS | 地线 |

从MLX90614读取到数据（DataH: DataH）之后，体温值T(单位)需要经过进一步换算得到，换算公式如xx-xx:

心率测量则采用Maxim公司设计的一款高灵敏度的血氧和心率生物传感器模块MAX30102。作为MAX30100的升级版，它具有更高的性能。两个模块部分参数对比如表xx-xx所示：

表 xx—xx Maxim MAX3010x系列产品参数对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **MAX30102** | **MAX30100** | |
| 分辨率（Bit） | 18 | | | 16 |
| FIFO深度 | 32 | | | 16 |
| 最大采样（SPS） | 3200 | | | 1000 |
| 耗电功率（mW） | 440 | | | 464 |
| 工作温度（） | -40 ~ 85 | | | -40 ~ 85 |

由表xx-xx可以看出MAX30102不仅性能更优，而且能耗更小。在可穿戴设备开发时，耗电问题是一个必须考虑的重要因素，因此本文选用MAX30102作为心率数据采集端。

MAX30102通过光电反射式原理来测量脉搏和血氧饱和度。MAX30102集光电收发、数模转换和数据滤波为一体[]，将采集到的数据发送到主控，主控采用相应的算法就可以得到对应的心率值和血氧值，文献[14]以MAX30102为核心设计了一套人体血氧测量系统，系统误差小于%3。

目前市场上智能手环监测睡眠的原理基本相同：在手环内部集成运动传感器，传感器测量佩戴者手部的细微运动，利用采集到的数据并结合睡眠算法判断佩戴者是处于活跃状态还是睡眠状态。MPU6050是一款常用的低功耗、低成本、高性能的空间运动传感器模块，目前主要应用于智能手机、平板和可穿戴设备。MPU6050集成了三轴陀螺仪传感器、三轴加速度传感器以及一个可扩展的数字运动处理器。如图是MPU6050的引脚功能介绍：

表 xx—xx MPU6050引脚图

|  |  |
| --- | --- |
| **引脚** | **功能** |
| VCC | 电源线，3.3V或5V |
| RX | 串行数据输入，TTL电平 |
| TX | 串行数据输出，TTL电平 |
| GND | 地线 |
| SDA | I2C数据线 |
| SCL | I2时钟线 |

可穿戴节点集成的传感器没有CPU，即没有计算能力，所以需要连接到主控芯片。这里采用双芯片方案，以恩智浦的FRDM-K64开发板作为主控，开发板通过SPI接口外接MRF24J40 IEEE802.15.4射频芯片，为网络提供通信的物理通道。

FRDM-K64是一款面向Kinetis K64、K63和K24 MCU的超低成本的开发平台。平台集成了MK64FN1M0VLL12 MCU，该MCU主频120M，闪存1M，256KB RAM，所以平台具有低功耗特性。平台的软件支持和应用开发由Kinetis软件开发套件(SDK)提供平台支持Zephyr操作系统和Arm Mbed，所以板级资源丰富，基于该平台可以快速实现节点的原型设计。

IEEE802.15.4无线收发无线收发器MRF24J40芯片内部包含有SPI接口、控制寄存器、MAC模块、PHY驱动器四个主要的功能模块，支持IEEE802.15.4，MiWiTM，ZigBee，Thread等协议，工作在2.405～2.48 GHz ISM频段，接收灵敏度为-91 dBm，最大输入电平为+5 dBm，输出功率为+0 dBm，功率控制范围为38.75 dB，集成有20 MHz和32.768 kHz主控振荡器，MAC基带部分采用硬件CSMA-CA结构，自动ACK6和FCS检测，CTR、CCM和CBC-MAC模式采用硬件加密(AES-128)，电源电压范围为2.4～3.6 V，接收模式电流消耗为18 mA，发射模式电流消耗为22 mA，睡眠模式电流消耗为2μA，具有低功耗模式。目前MRF24J40芯片主要应用于无线传感网络、家庭自动化、楼宇自动化等，所以该芯片可以为系统提供物理信道的理想方案。

## 本章小结

本章首先结合了第二章对智能家居网关功能需求分析，自主设计一个轻量级的智能家居网关，介绍各个功能部件的详细设计与实现过程。最后，对智能家居系统节点进行硬件设计。

# 基于微服务架构的边缘网关系统设计与实现

本章分析和确定基于边缘计算的智能家居网关系统的功能需求，并结合Edge X Foundry微服务框架给出基于微服务架构的边缘网关系统的总体设计。本章基于微服务架构的边缘网关系统会沿用第三章对智能家居系统节点硬件设计，因此本章不再赘述。

## 设备服务

### 设备服务需求分析

设备服务层作为边缘网关与设备之间的链接器，其结合了第三章中三层架构模式的网关的感知层和模型层，同时作为边缘网关的一部分，它又体现了边缘计算的优势，将设备所有数据包括设备信息，设备通信数据和设备命令等转发到数据库缓存，为边缘计算提供数据；另一方面，设备服务通过设备监视器，预定义规则事件等使得边缘网关更加智能化。

总结设备服务的功能如下：

1. 收集和转发设备数据

默认情况下，设备服务将从设备和传感器接收到的数据立即发送到核心服务层，可以通过定义特定的规则来触发或延迟推送数据到核心服务层。当成功完成设备数据的收集和转发之后，设备服务会使用新的数据更新设备状态。

1. 接收并响应命令

微服务（主要是Core Command）通过发送GET请求到设备服务以获取对应设备的最新数据。默认情况下，设备服务会将GET请求编码成一定的格式发送给设备；设备服务可以将设备上传的最新数据存储在设备数据缓存区，当接收到GET请求时，设备服务不必立即转发命令到设备或传感器以获取数据，而是通过缓存获取设备或传感器的数据，当设备服务的数据是通过被推送获取时，这种方式是必要的，当设备和传感器的通过异步传输数据流到设备服务时，设备服务有必要缓存接收到的数据。当设备服务成功地从设备和传感器获取到数据后，设备服务会使用新的数据更新设备状态。

微服务（主要是Core Command）通过发送PUT请求到设备服务以更新设备状态，与GET请求不同的是，GET请求必须响应一些数据，而PUT请求不必相应任何数据，除了在错误或异常情况时。

1. 数据转换

设备服务必须实现数据转换逻辑，设备和传感器通过不同的通信方式将数据发送到设备服务，设备服务将数据发送到Core Data之前需要将这些数据转换成特定的统一格式。

1. 断开设备连接：

设备和传感器可以请求断开与设备服务之间的连接，也可以请求从设备服务中删除设备。这种情况下，Core Metadata会删除设备信息并删除缓存中所有的可选数据。断开设备连接不会影响此后的设备重连。

1. 报告设备状态

设备服务可以周期性的“ping“设备来确定设备是否可以访问；设备可以定期推送状态到设备服务。当设备服务在收集设备数据或对命令做出响应时发生错误或异常，设备服务会更新设备状态为错误或异常，甚至中断设备连接。当确定设备状态之后，设备服务会将次更新推送到Core Metadata。

1. 设置设备管理状态

设备服务会处理设备请求更新设备的管理状态。当设备管理状为未锁定时，设备服务将停止推送相关的数据到Core Data，同时忽略发送给设备的GET和PUT命令。

1. 报告设备服务自身运行状态

设备服务必须定期向Metadata和Config/Registry服务报告其自身运行状态，设备服务运行状态可以是启用或禁止。

1. 响应Metadata命令

设备服务必须能够接收Metadata的通知或回调。例如，EdgeX Foundry的用户界面等应用程序通过Metadata发送请求以删除设备，则该请求将通过设备服务的Metadata回调函数被中继和处理

1. 响应配置更新

设备服务通过回调从config/registry服务获取更新的配置数据。设备服务会永久性地支持响应配置更新。设备服务可以根据配置更新动态调整自身以使用最新配置，也可以存储配置更新并在下一次启动服务时使用它。

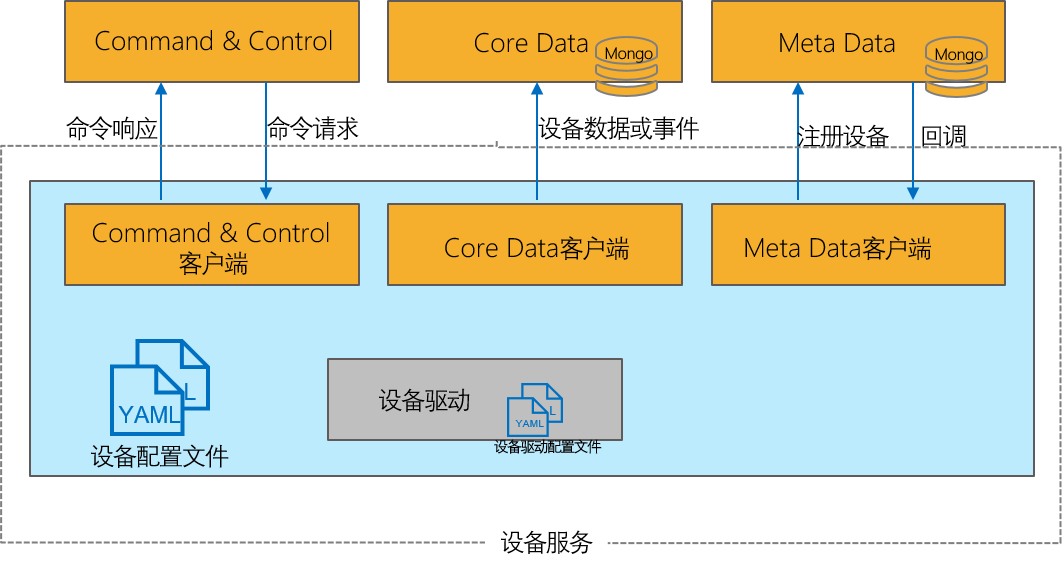
1. 其他功能

设备和传感器白名单/黑名单：设备和传感器的白名单/黑名单决定设备服务是否能够管理对应的设备和传感器。

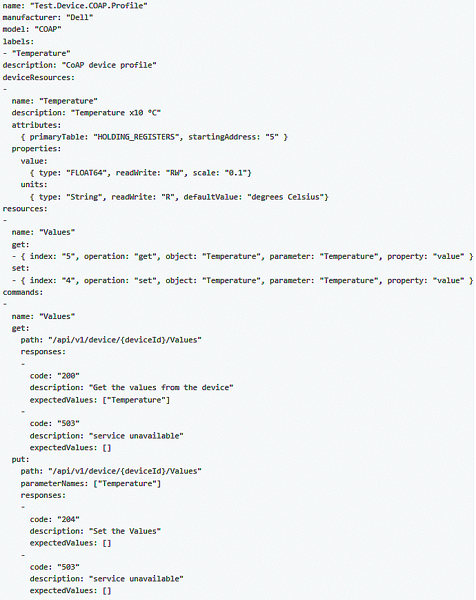
设备发现：在一定的条加下设备服务具有设备发现能力：设备服务实现了与设备和传感器通信协议接口；设备和传感器接入边缘网关所在的局域网之后，开始通过广播、组播等方式来“广播”控制请求。设备服务通过搜索或其他方式“发现”新设备时，触发设备服务与设备和传感器连接，连接成功之后，设备和传感器会被注册到Meta Data。在设备服务中通过建立发现规则以定义设备发现的条件和扫描时间。

### 设备服务总体设计

如图XX-xx是一个设备服务的框架



在设备服务层，一个设备服务对应一类设备，默认是具有相同通信协议的设备，这些设备具有相同的一般特征，因此每个设备服务中用一个设备配置文件定义设备。设备配置文件详细描述设备类型的一般特征，如设备名称，设备提供的数据，设备命令方式等。设备配置文件格式是YAML格式，这里采用YAML格式是因为它不仅易读性强且可以用于跨不同语言和框架的配置文件。如图xx-xx是一个用于测试的CoAP设备服务的配置文件。



对设备发送命令的格式

设备资源以及资源属性

设备基本

信息

资源访问方式

设备服务中根据设备通信协议定义不同的驱动以实现与设备之间的数据交互。为了实现设备服务层广泛的接入能力，即兼容多种通信协议与不同的设备通信，可以在设备服务层实现多个不同的设备服务，其中每个设备服务具有不同的驱动。设备服务在启动时会加载一个文件，其中定义了驱动使用的协议以及协议的IP地址和端口号。

1. CoAP设备服务驱动

CoAP是运行在UDP协议上的物联网应用层协议，而UDP协议是面向非连接的协议，即客户端在传输数据之前和服务器端不建立连接。服务器端通过绑定端口来监听数据，客户端只需要将数据封装到一个报文中，然后发送给服务器端。因此需要在设备服务驱动中部署一个CoAP服务器，用于监听底层加点发送的数据。当设备服务接收到命令时，首先需要将命令转换成CoAP命令格式，设备服务驱动会创建一个CoAP客户端将命令发送个底层节点并等待响应。如图xx-xx是CoAP设备服务的驱动结构图。

1. LwM2M设备服务

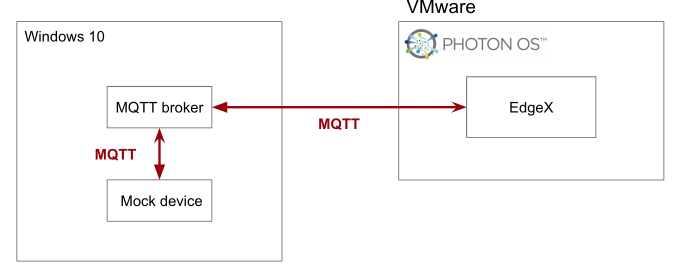
LwM2M协议使用CoAP作为底层的传输协议，它提供了设备管理功能。在LwM2M协议中将底层节点抽象为Object，节点的提供的功能属性抽象为Resources，并用使用一个对象注册表定义物联网开发中各种节点常见的功能和属性。因此，可以直接将LwM2M协议的对象注册表映射为设备配置文件。如图xx-xx是LwM2M设备服务驱动的结构图，

LwM2M设备服务驱动

其中在设备驱动中部署一个LwM2M服务器，底层节点作为该服务器的客户端。底层节点首先需要发送注册请求到设备服务，LwM2M服务器接收到请求后，将设备注册为自己的可管理资源，同时将设备信息添加到Core Metadata服务中。当底层节点注册成功之后，LwM2M服务器会启动Observer命令监听底层节点，当设备服务接收到命令时，设备服务的驱动将命令转换成LwM2M协议的Read或Write命令以获取或更新底层节点。

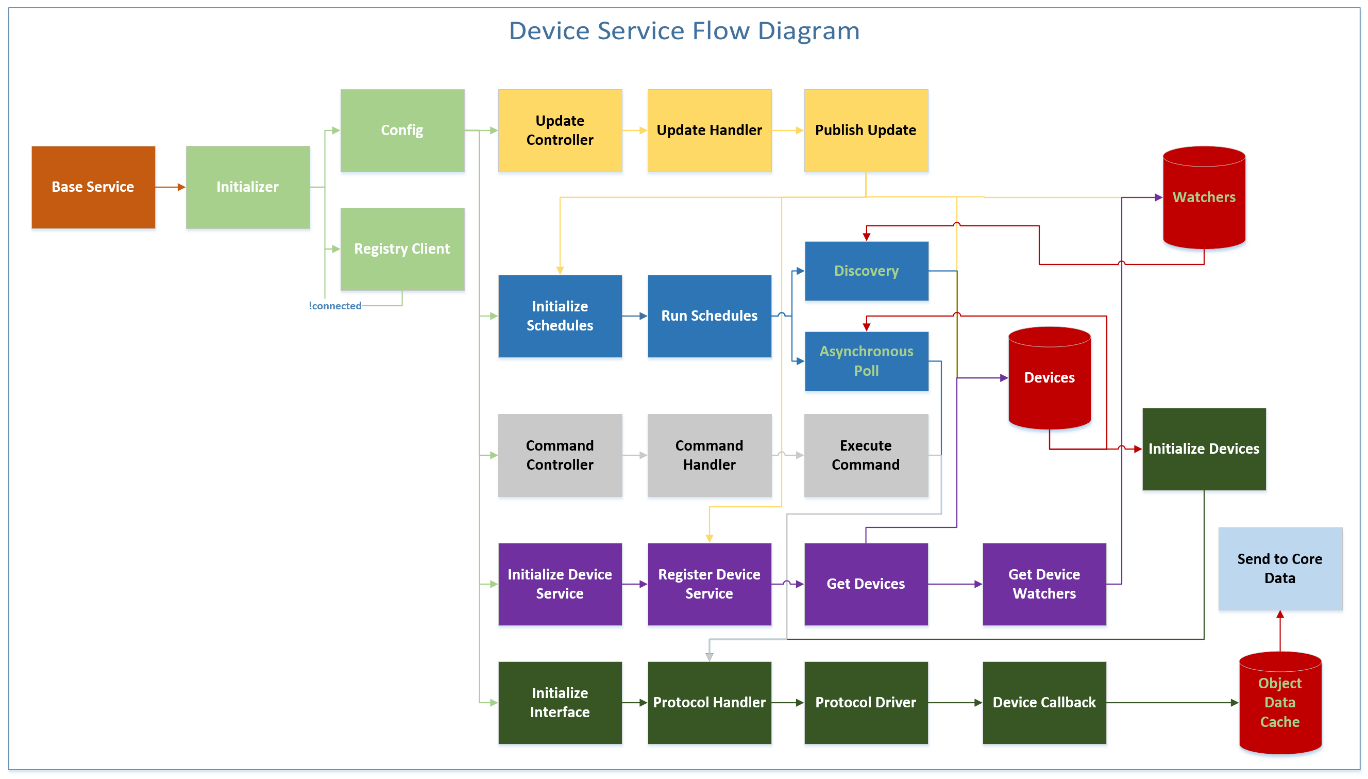
1. MQTT设备服务驱动

如图xx-xx是MQTT设备服务驱动的结构图



在MQTT设备服务驱动中部署了一个MQTT代理服务器，用于接受来自客户的网络连接并处理客户端发布的消息，同时向消息订阅者转发消息。MQTT设备服务驱动定义了三个Topic：DataTopic、ResponseTopic和CommandTopic。底层节点发布数据时使用DataTopic主题，MQTT代理服务器接收到消息后，设备服务会将消息转换成JSON格式发送到核心服务。当设备服务接收到命令时会创建一个MQTT客户端，该客户端会使用CommandTopic，将命令作为Payload发布出去，底层节点通过订阅CommandTopic主题来获取命令，底层节点使用ResponseTopic主题发送响应。

设备服务工作流程如图xx-xx



设备服务启动之前需要初始化设备服务依赖的客户端，其中Metadata Client和Core Data Client分别与Metadata和Core Data建立连接。在初始化Metadata Client和Core Data Client时需要检测Metadata和Core Data两个服务是否正常运行。设备服务的初始化过程会处于挂起状态，直到设备服务能够成功的使用Metadata和Core Data两个服务。Logging Client是设备服务的另一个依赖客户端，Logging Client与支持服务Logging 建立连接，为设备服务提供日志服务。

如图中紫色部分是设备服务的工作流程。首次启动时，设备服务请求Config/Registry服务注册自己。如果Config/Registry服务正在运行，设备服务在正式运行之前将从Config/Registry服务获取自身的配置参数，否则从本地配置文件中获取。设备服务再次启动时需要检测自上次启动以来配置更新，并将更新同步到Core Metadata等所有与设备关联的微服务中。当设备服务完成注册后，首先加载设备参数信息，包括设备名称，设备参数和命令等。在一个设备服务中所有设备的参数信息相同。设备服务配置文件中可以预定义设备，设备服务启动时会将设备添加到Metadata中，同时也可以为设备预定义规则事件，设备服务通过ScheduleClient与支持服务的Schedule建立连接，触发相应的规则事件。获取设备监视器。当设备服务首次启动时，会添加一组预定义的设备，设备监视器会监听新设备的介入请求，然后将设备的元数据发送到Metadata。

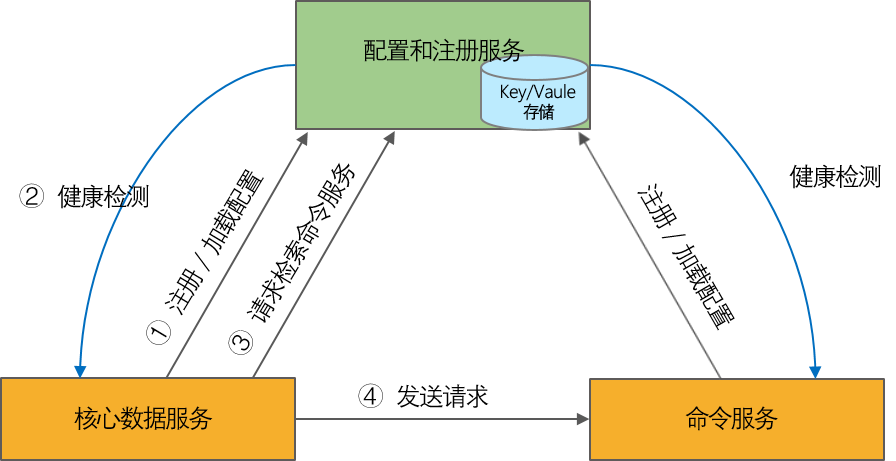
由图可以看出，设备服务需要多个服务协同工作。

## 核心服务层设计

核心服务层，顾名思义是整个系统的核心。在EdgeX Foundry框架中核心服务层是物联网“南向”物理节点与“北向”应用之间的桥梁。在核心服务层部署了四个微服务：配置和注册服务、核心数据服务、元数据服务和命令服务。

### 配置和注册服务

本文设计的边缘网关系统中具有多个微服务，服务调用者需要知道被调用服务的访问信息，才能进行访问。另一方面，由于服务的访问信息可以动态改变，人为地添加系统中所有服务的访问信息不仅效率低，而且可靠性和稳定性无法保障。因此需要一套完善的服务发现机制来实现服务注册、服务发现自动化，并且可以动态地实现服务的注册、查找和删除。本文以开源的Consul服务发现框架为基础，设计并实现了配置和注册服务。



配置和注册服务是整个系统中所有服务的管理器，负责为每个服务提供配置信息，且该配置信息可以覆盖服务内部配置。当服务的配置信息如端口号已经被占用时，配置和注册服务会动态为其分配一个新的端口号，这种情况下，配置和注册服务会向该服务发送通知提醒端口号被改变。

配置和注册服务是整个系统中的注册表，配置和注册服务掌握系统中所有服务的访问地址和当前运行状态。每个服务在启动时，首先必须向配置和注册服务注册自己，注册信息包括服务名、IP地址、端口号等，收到注册请求时，配置和注册服务会以Key/Value的方式将信息添加到注册表中，然后，配置注册服务会定期的“ping”该服务，以检测该服务的健康状况，如图xx-xx所示。当系统中的一个服务需要访问另一个服务时，首先通过配置和注册服务检测该服务是否可以访问，如果可以则获取对应的IP地址和端口号。

系统可以在没有配置和注册服务的情况下运行，这种情况下，系统中的服务使用默认的配置在本地运行，并且服务之间通信时无法保证对方是否可以访问。

边缘网关中，每个服务都有一个配置文件

### 边缘数据存储

为了在边缘网关中实现对节点及相关数据的持久化存储，在核心服务层加入了元数据服务和核心数据服务，并基于MongoDB数据库实现两个服务。

MongoDB是一种非关系型的分布式文件存储库。文档是MongoDB的核心概念，作为基本数据单元，文档可以包含任何结构的数据，如节点读写事件、时间序列数据、文本等其他任何结构。集合是MongoDB的另一个核心概念，集合存在于MongoDB的数据库实力中，是一组文档的集，一般将同一类型的文档放置在一个集合中，避免数据混乱，同时数据更加集中，以提高查询效率。

MongoDB数据库的特点：

1. 物联网节点数据的一大特征是非结构化，MongoDB数据读写效率高，使用文档模型将一个实体对象的相关数据集中在一个文件中，同时对数据结构没有要求，可以满足非结构化物联网数据的存储需求，同时使用集合将同类型的实体集中在一起，提高数据查询效率；
2. 可扩展能力强-关系型数据库很难做分布式的原因就是多节点海量数据关联有巨大的性能问题。如果不考虑关联，数据分区分库，水平扩展就比较简单；

3） 动态模式-文档模型支持可变的数据模式，不要求每个文档都具有完全相同的结构。对很多异构数据场景支持非常好；

4） 模型自然-文档模型最接近于我们熟悉的对象模型。从内存到存储，无需经过ORM的双向转换，性能上和理解上都很自然易懂

、为了实现边缘网关系统对节点信息以及节点所传输数据的持久化存储，在网关框架中加入了元数据服务和核心数据服务。

根据节点数据的时间有序、实时性要求高、数据量等特点，系统选择使用MongoDB数据库。

建立两个数据库，metadata数据库，其中包含三个集合，分别是s

MongoDB可以直接使用一个文件涵盖一个节点所有信息。

数据库设计：

采用EAV表对节点数据进行建模。EVA，即实体-属性-值。使用EVA模型建模时，需要建立两张表：其中一个表格在不同的行中存储不经常发生变化的数据，如节点名称、IP地址和端口号、通信协议等；另一个表用于存储只存在某些行为或多变的数据，如温度、心率、速度等。

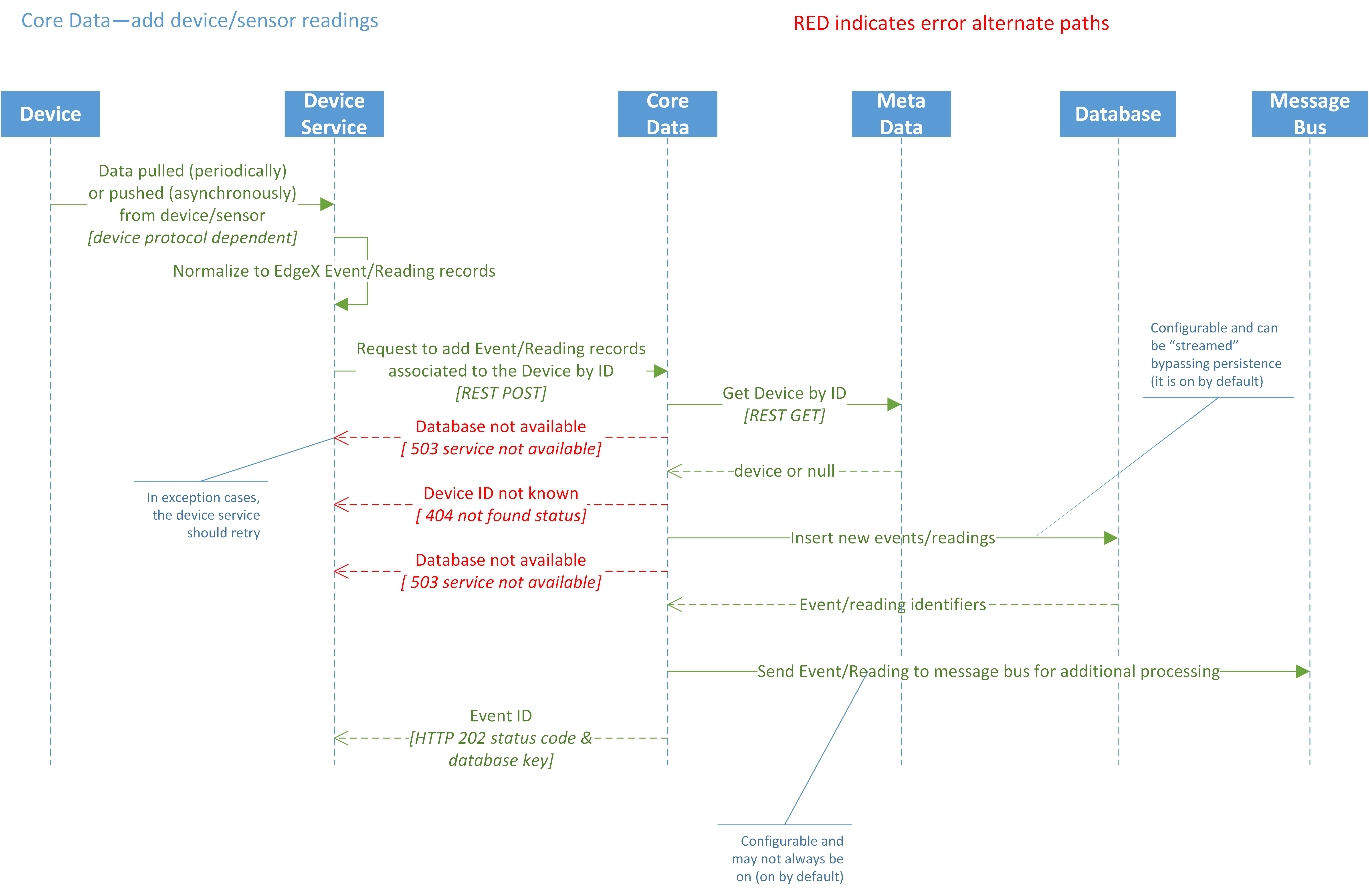
MongoDB支持自动分片,可以摆脱手动分片的管理.集群自动切分数据,做负载均衡.

在网关中每一个节点对应一个实体。通过一个对象表获取每个节点的共同数据

节点通过设备服务层接入到边缘网关系统中，其对应的设备服务配置文件和设备配置文件的信息会被存储到元数据服务中。其他服务需要访问设备时，首先通过元数据服务检索被访问设备的通信协议、设备地址以及设备可访问资源等信息。

节点发到设备服务层，设备服务层将数据处理之后发送到核心数据服务进行存储。默认情况下，核心数据服务会永久存储这些数据，也可以通过修改配置将数据转发到系统中其他服务。

如图xx-xx所示，节点将数据发送到设备服务层，设备服务将数据处理之后发送到核心数据服务，然后核心数据服务调用元数据服务以查询该节点是否存在，当节点存在时，元数据服务将数据存储到MongoDB数据库，否则返回异常响应。

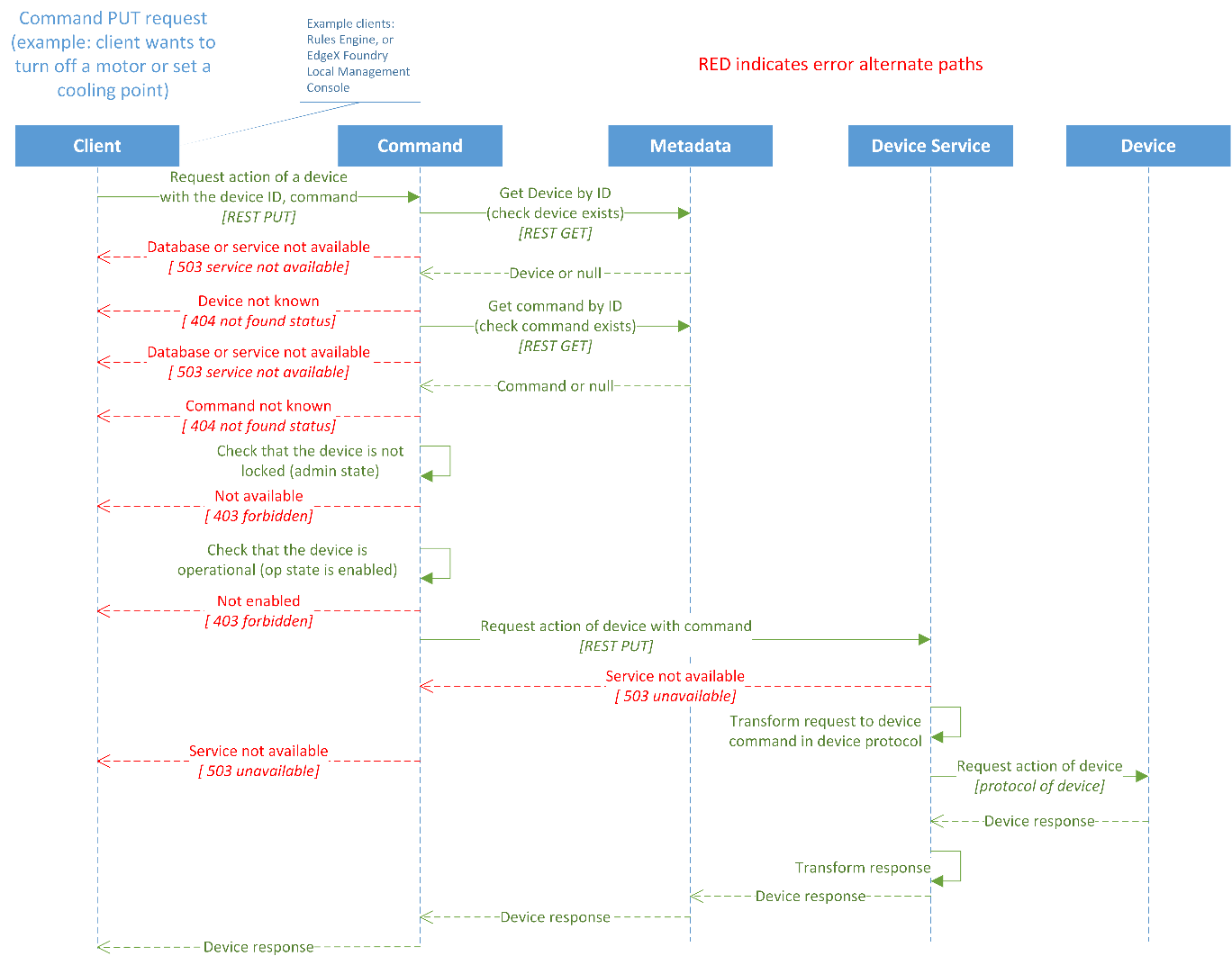


### 命令控制服务

本文边缘网关系统通过命令控制服务为其他所有服务提供访问节点设备的接口命令控制服务为每个节点定义了两组访问命令：

1. GET命令，发送GET命令到设备服务，可以获得对应节点的特定资源的当前值，如温度传感器的当前温度值，开关的当前状态。
2. PUT命令，发送PUT命令到设备服务，可以更新对应节点的特定资源的值。

如图当用户通过Web应用发送PUT指令时，首先指令被发送到命令控制服务，如果在元数据服务中可以查询到对应的节点并且指令合法，命令控制服务会将指令发送到设备服务，最后设备服务将编码之后的指令发送给对应的设备。

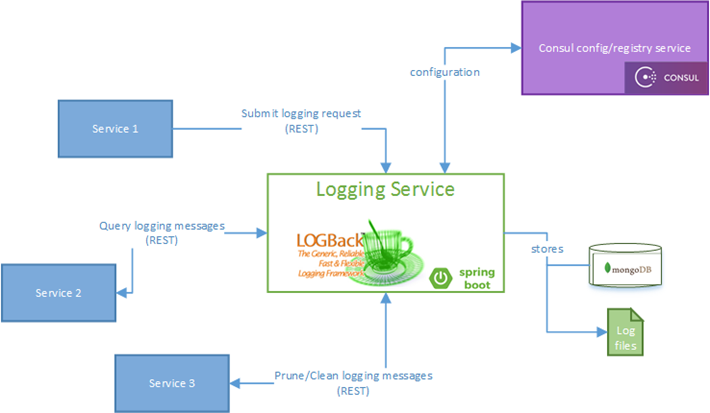


## 支持服务

支持服务层主要目标是通过多个服务为边缘网关提供边缘分析和智能化服务。本文边缘网关系统在支持服务层加入了日志服务和异常提醒服务。

日志是现代软件应用程序重要部件。日志记录可以监控系统的运行状况，了解系统中各个服务之间的相互作用，通过详细的日志记录可以及时发现系统存在的问题[16]，有利于系统性能不断改善。

本系统中使用Spring Boot应用框架设计日志服务，同时使用LOGBack作为底层日志框架，如图xx-xx所示



系统中其它服务通过RESTful API向日志服务提交日志记录请求、查询日志和删除日志记录。日志服务为每一个调用设置相应的配置信息，如时间戳、日志级别、标签和对应服务等，以便于快速查询获取对应服务的日志；支持多个级别的日志记录，例如Debug、INFO、WARN、ERROR和 FATAL；日志可以存储到文件和数据库中。

## 应用服务

本文边缘网关系统可以在不连接北向应用的情况下独立运行在家庭局域网中，同时不受其它外界环境的控制和监管。为了给底层节点提供更好的服务，可以将节点数据发送到北向应用如云端，应用服务层就是为这个目的而设计的。在应用服务层主要有两个服务客户端注册服务与客户端分配服务。

### 客户端注册服务

无论是网关外部还是网关内部的服务都可以通过客户端注册服务注册为应用服务层的客户端，然后接收核心数据服务传输的数据

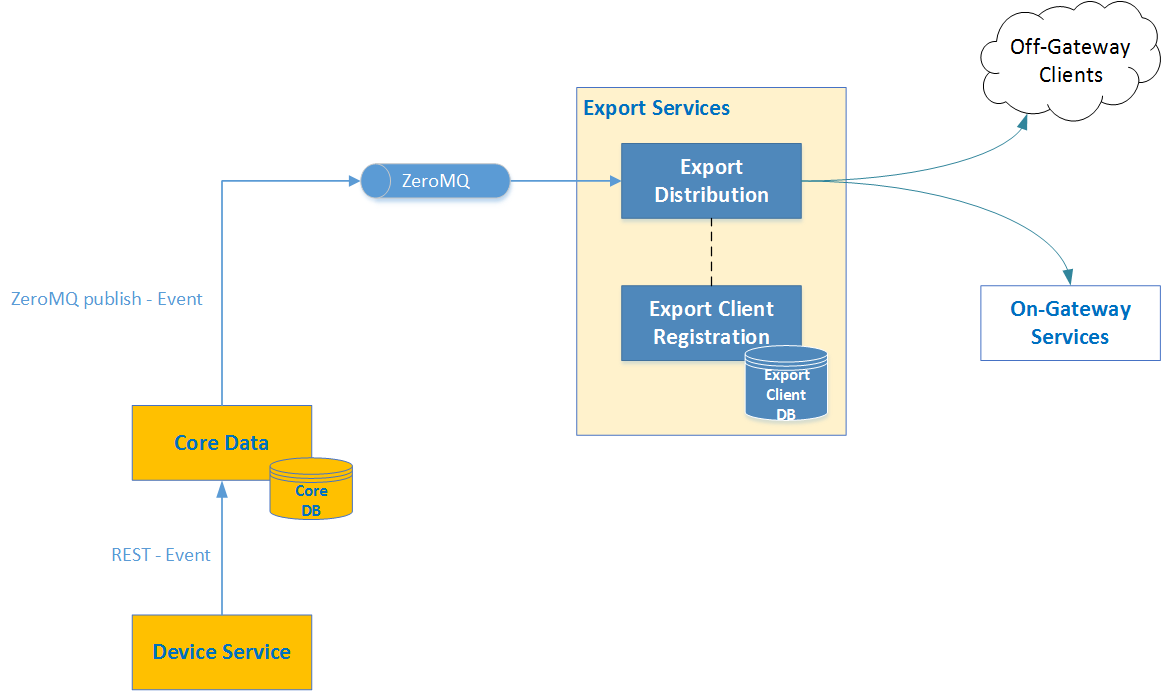
客户端注册时需要提供必要的信息：默认情况下，核心数据服务将所有接收到数据都发送到应用服务层，因此客户端注册时需要指定所要接收信息的内容或范围，通过指定节点对应的名称或ID使客户端只接收对应节点的数据，也可以指定所要获取数据的描述符，即数据的类型和内容，使客户端仅接收到使用对应描述符的数据；客户端在注册时可以指定数据发送到客户端的方式，包括数据的格式、加密方式和压缩方式等，例如将数据编码成JSON格式并对数据进行AES加密后发送到客户端；客户端注册时必须提交客户端相关的IP地址和端口号等信息。

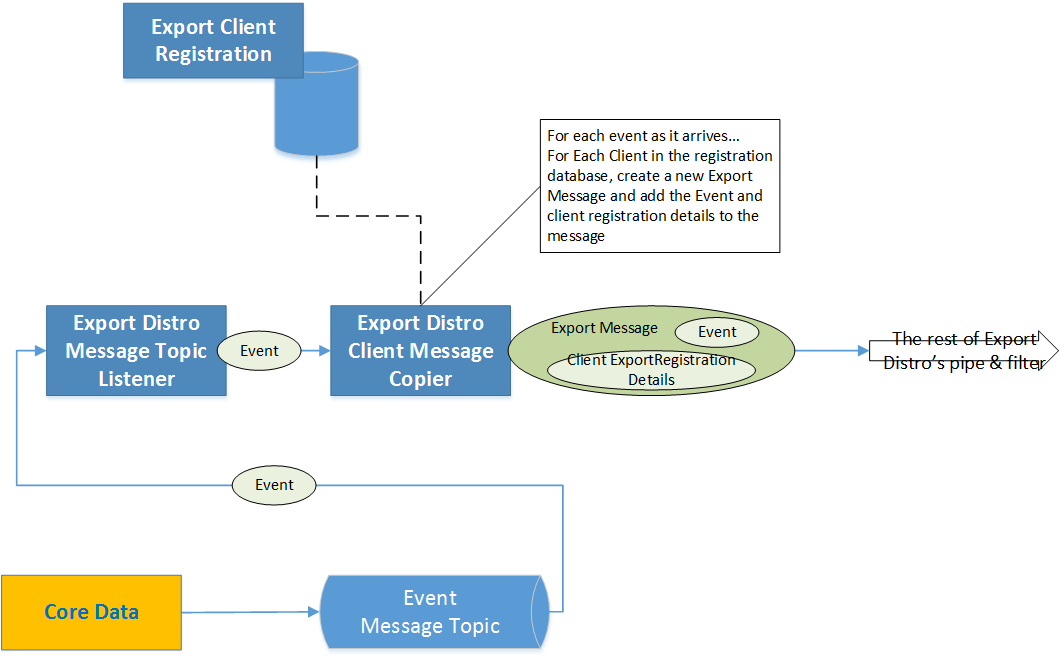
客户端注册成功之后，客户端的信息被存储在本地的MongoDB数据库中。

### 客户端分发服务

客户端注册服务负责客户端信息注册，并将客户端信息存储到数据库。客户端分发服务通过调用数据库获取数据流向、数据发送方式等信息。如图xx-xx所示，节点数据通过设备服务进入到核心数据服务，然后核心数据服务通过消息队列将数据发送到客户端分配服务，客户端分发服务将数据进行过滤、转换和格式化，然后通过MQTT、REST等通信方式分发到对应的客户端。

本文边缘网关系统的客户端分发服务是基于Spring Integration消息传递框架设计。在客户端分发服务中设置ZerMQ的客户端，该客户端订阅具有特定主题Topic的消息，当消息到达时，客户端分发服务访问客户端注册服务的数据库，查询那些客户端是该消息的接收者，然后按照对应客户端的信息与消息组合在一起发送到客户端分发服务的消息通道中，利用Spring Integration的管道和过滤器将数据分发到对应的客户端。





如图xx—xx所示，

## 本章小结

本章根据EdgeX Foundry框架的层次结构设计应用于智能家居的边缘网关。其中对设备服务层做了重点介绍和分析，包括设备服务层的功能，设备服务的结构和工作流程以及不同通讯协议对应驱动的设计过程；接着依次对核心服务层、支持服务层和应用服务层的服务部件功能及设计方法做了详细介绍。

# 实验结果与分析

## 系统测试环境搭建

### 测试环境

1. 硬件环境

本文测试实验搭建了一个智能家居系统，搭建该系统需要以下硬件设备：

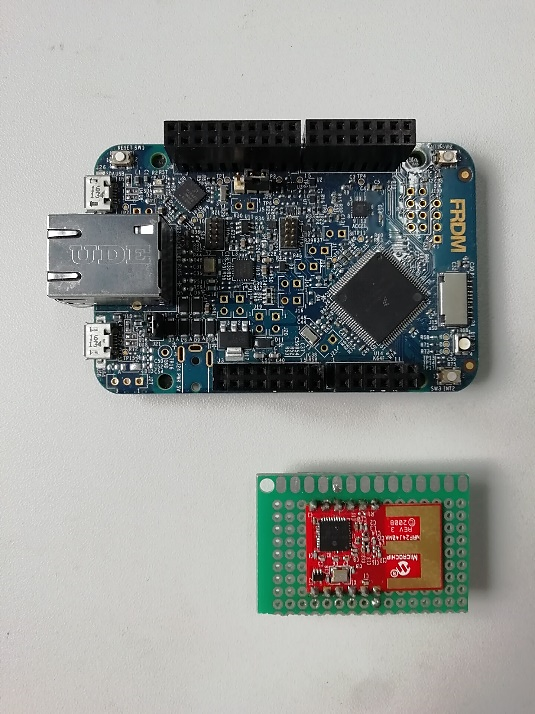
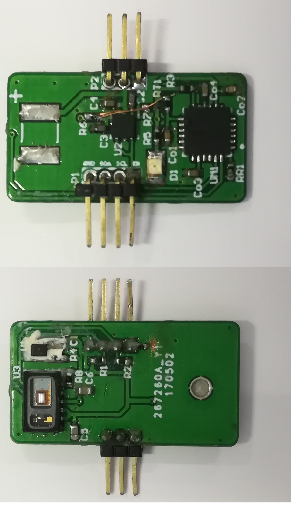
  

图5-1 智能家居测试系统硬件

测试系统硬件主要分为三个部分：

网关部署平台：本文所设计的两个网关都部署在树莓派上。如图A是树莓派，型号是3B。树莓派搭载1.4GHz 、64位四核Broadcom Arm Cortex A53架构处理器。使用32GB的SD卡用于存储操作系统、文件和网关程序。

节点主控平台：图B是底层节点主控FRDM-K64开发板，节点的传感器和执行器将数据通过串行外设接口SPI发送到主控，然后主控通过IEEE802.15.4无线收发无线收发器MRF24J40将处理后的节点数据发送到网关，图C是MRF24J40模块。

节点：如图D是一个可穿戴的硬件节点的正反面示意图，可以采集用户的体温、脉搏、运动量等数据。图E是一个智能台灯节点，由普通台灯改造而成。

1. 软件环境

软件环境包括操作系统和影响系统功能与性能的主要框架、工具。详细信息如表5-1所示：

表 5-1 网关软件环境

|  |  |
| --- | --- |
| **软件环境** | **版本号** |
| 操作系统 | Raspbian 2018-06-27-raspbian-stretch-lite |
| MongDB版本 | MongoDB shell 2.4.14 |
| Golang版本 | go1.10.3 linux/arm |
| Nodejs版本 | Node10.15.1 linux/arm |
| Consul版本 | Consul1.4.3 linux/arm |

### 可穿戴节点硬件参数测量

可穿戴节点作为监测婴幼儿睡眠状况的智能手环进行设计，通过腕带固定在手环上。节点尺寸为，与手腕接触面积小，重量为，佩戴轻便，基本不影响手腕正常运动。可穿戴节点与主控之间通过杜邦线连接，同时由主控为可穿戴节点供电。

可穿戴节点实物图如下：

图5-2 可穿戴节点实物图

**正面** **反面**

可穿戴节点最终佩戴效果如下图所示：

图5-3节点佩戴效果图

台灯节点实物图如下：

### 测试软件环境搭建流程

本文所设计的边缘网关软件环境搭建过程比较复杂，因此在这里给出详细的搭建步骤：

1. 在树莓派上安装Raspbian操作系统，并安装依赖包Git、ZeroMQ库等。
2. 下载Golang源代码并编译。为了缩短编译时间，本文基于Golang1.4.3 linux/arm二进制文件对Golang1.10.3 linux/arm的源代码进行编译。下载下载Glide包管理工具，Glide用于下载程序需要的包。
3. 下载网关程序依赖库，编译网关程序。
4. 安装Mongo DB数据库，然后初始化数据库。
5. 安装Consul，启动Consul代理。

## 基于单块架构模式的智能家居网关测试

本节基于单块架构模式的智能家居网关完全由本人自主设计并实现。

### 协议测试

编写Node.js脚本，模拟CoAP、LwM2M、MQTT三种节点。节点首先向网关发送注册请求，注册成功以后每5秒钟发送一个体温数据。

CoAP协议测试

首先，可穿戴节点向网关发送注册请求

### 功能测试

功能测试，在软件测试中称为黑盒测试，主要是从用户应用的角度出发，验证系统的各个功能是否能够正常地运行。只要各种输入情况下系统输出均正确，就可以得出系统功能测试通过。

本节网关的功能测试主要从两方面展开：模拟节点向网关发送请求，在网页中人工点击网关节点管理界面，验证网关节点管理功能是否符合预期功能；设置不同的场景，验证由可穿戴带节点、台灯节点和网关组成的智能家居系统能否做到实时监测体温、心率和睡眠状况，异常场景触发报警，两个节点之间互联互通等。验证系统节点管理功能的测试用例如表5-2所示：

表5-2 节点管理功能测试用例

|  |  |  |
| --- | --- | --- |
| **序号** | **测试步骤** | **预期结果** |
| 1 | 底层节点向网关发送注册请求 | 可新增节点并显示在节点列表中 |
| 2 | 注册成功后，节点发送体温数据 | 节点的资源列表中新增一个温度传感器，同时该温度传感器有一条数据 |
| 3 | 节点列表页面点击删除图标 | 可删除节点以及节点相关的资源和数据 |

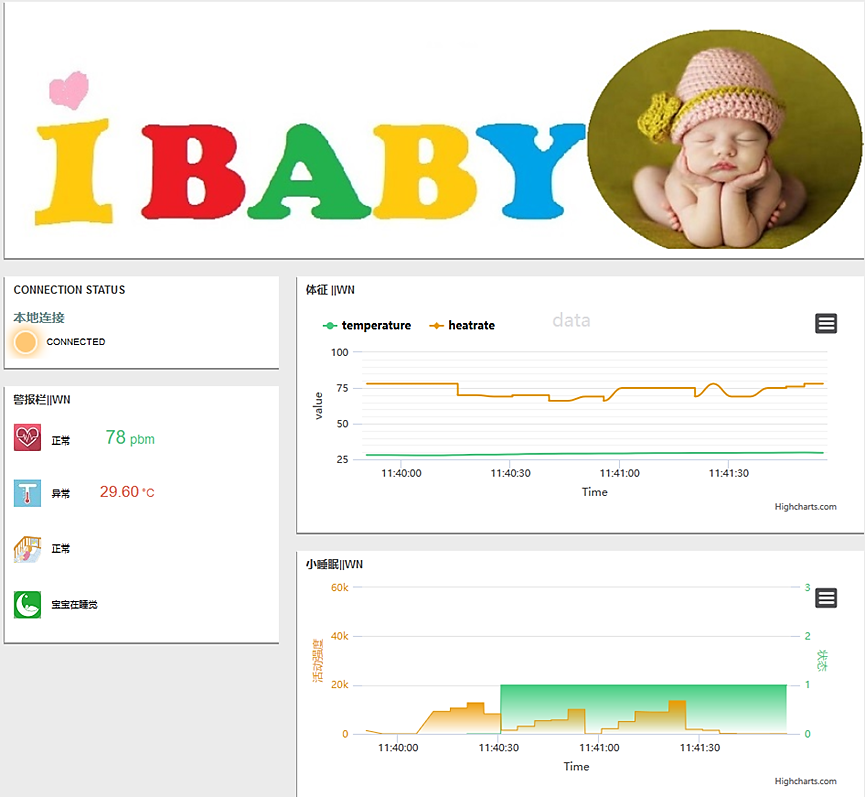
智能家居系统功能测试方案:

1. 模拟婴幼儿突然体温异常的场景，体温过高为发烧，体温过低为着凉；

模拟发烧：将手环模块靠近热水袋，网页中警报栏将显示体温异常报警信息；模拟着凉：将手环环模块靠近冰水袋，网页中警报栏将显示体温异常报警信息。

1. 由于无法模拟人体心率异常，可通过网页查看体温心率曲线；
2. 翻身俯睡报警：翻转智能手环模块，使其正面逐渐朝下，达到一定倾角时，网页警报栏将显示俯睡报警信息；
3. 睡醒提醒：在一段时间内保持智能手环模块静止不动，系统将认为被测试者已经处于睡眠状态，之后连续活动，模拟婴幼儿醒来时身体活动的场景，网页警报栏将显示婴幼儿睡醒的提醒信息；
4. 睡眠状态监测：为了方便测试，把睡眠状态监测的时间单位由1min改成了5s。在网页睡眠曲线中会绘制出人体活动量曲线和状态曲线。模拟婴幼儿睡眠场景，测试者要轻微活动或者保持不动，此时活动量很低甚至为0；模拟婴幼儿清醒的场景，测试者手部持续做出较大幅度动作；网页将实时显示活动强度和睡眠-清醒的状态。

### 测试结果



根据测试：

1. 验证节点管理模块的三个测试用例均通过。

2）网关感知层能够接入不同通信协议的节点，实时上报节点数据；模型层能够正确转换节点数据和解析视图层指令，缓存节点信息和节点资源数据，定时将缓存数据同步到Mysql数据库进行持久化存储；视图层能够正确解析指令以管理节点，实时显示节点数据。如图xx-xx所示是网关视图层的网页界面，图中可以看到节点的实时数据，以及异常体温警报。

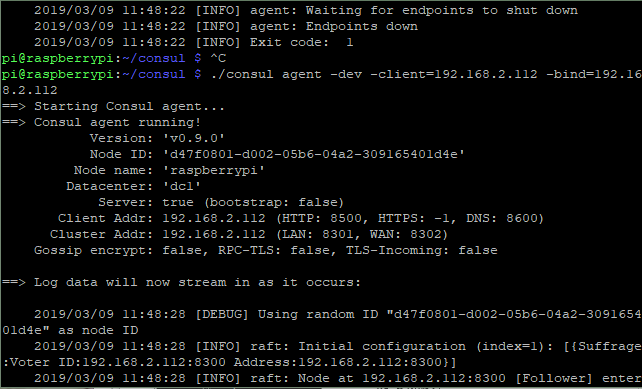
## 基于微服务架构模式的智能家居网关系统网关系统测试

### 功能测试

1. 注册和配置服务功能测试

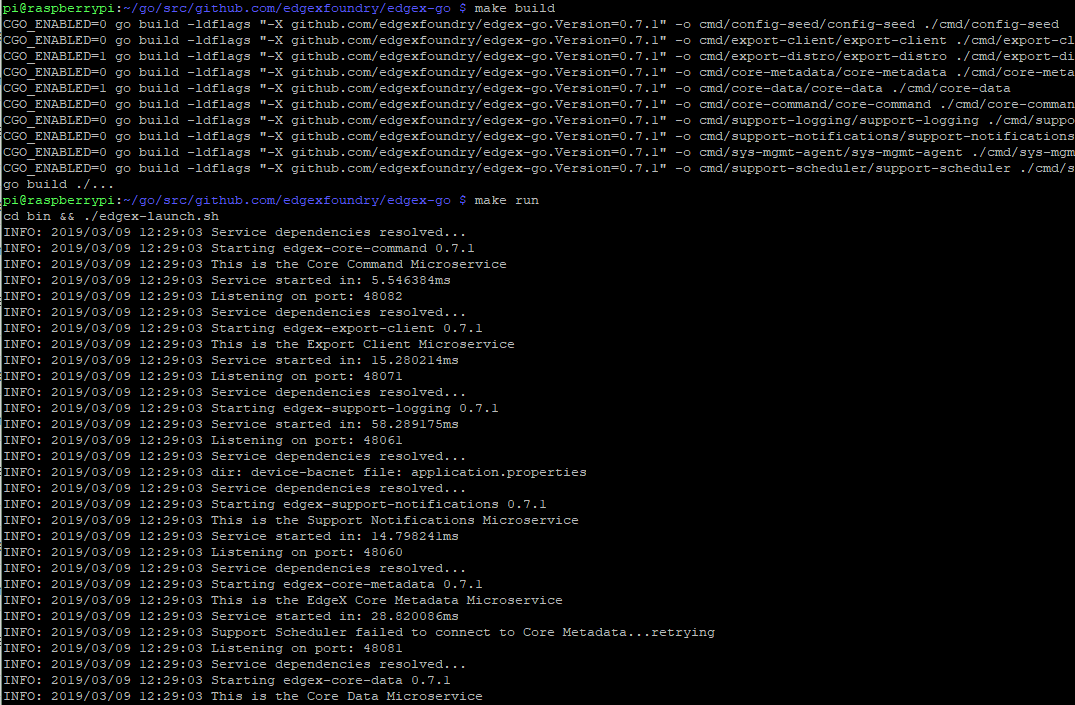
网关启动注册与配置服务之前，首先需要启动Consul代理。Consul代理可以维护成员信息，注册服务，执行健康检测，响应查询请求等。代理必须在Consul集群中的每个节点上运行。代理可以以客户端或服务器两种模式运行。代理节点上在客户端模式下，所有注册的服务信息均会被转发到服务器模式的代理，代理节点无法持久存储信息。相对于客户端模式，服务器模式除了具有相同的功能之外，还能够持久存储注册的服务信息。

启动Consul代理的命令如图xx-xx所示，测试时使用开发模式运行consul，使用参数绑定地址192.168.2.112，以允许consul集群中成员之间通信，使用参数绑定的地址允许consul集群中client成员访问。

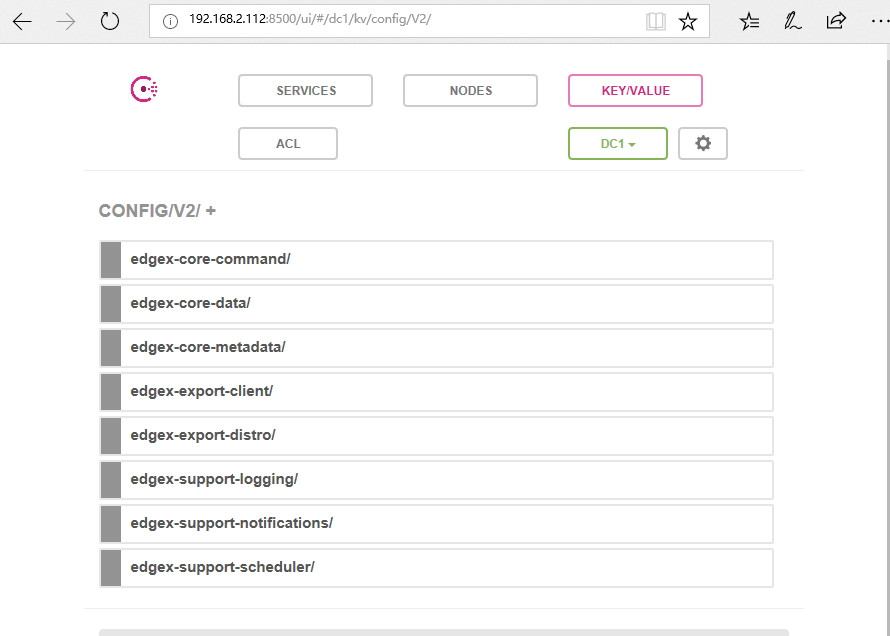


Consul代理以服务器模式运行，使用xx文件将存储Consul代理的数据。

启动边缘网关，如图xx-xx是网关启动时控制台输出，根据输出的日志信息，可以看到网关中，各个服务依次启动并注册到Consul代理中。为了在PC端执行树莓派上的网关程序，测试时将PC和树莓派接入到同一个局域网中，使用PuTTY远程访问树莓派。PuTTY是SSH和telnet客户端，它支持SSH，telnet和原始套接字连接，具有良好的终端仿真功能。



访问<http://192.168.2.112:8500/ui>查看Consul注册表，如图xx-xx所示，边缘网关中的各个服务已经成功注册，并且服务以Key/Value的形式存储。点击服务列表，可以查看每个服务的配置信息。



curl是一种文件传输工具，在Linux环境中的按照URL规则在命令行中工作。Curl可以作为REST的客户端，完成REST API测试。利用curl指令可以发送HTTP协议的GET、POST、PUT和DELETE请求。边缘网关为每个服务都定义了可访问的API列表。在网关启动之后，使用如表xx-xx所示的两个API分别测试服务是否可以访问，服务的配置参数是否正确。

|  |  |  |
| --- | --- | --- |
| **方法** | **访问URL** | **预期结果** |
| GET | 192.168.2.112: {service\_port}/api/v1/ping | 可返回所有字符串“pong” |
| GET | 192.168.2.112:{ service\_port }/api/v1/config | 可返回服务的配置 |

1. 设备服务功能测试

本文为边缘网关设计了三种设备服务，分别对应CoAP、LwM2M、MQTT三种设备服务。根据节点上的传感器编写设备配置文件，如节点名称，节点资源描述，访问节点资源的GET或PUT命令等，以下是节点温度传感器描述：

{

"name":"temperature",

"min":"35",

"max":"42",

"type":"F",

"uomLabel":"degree cel",

"defaultValue":"0",

"formatting":"%s",

"labels":["temp","health"]

}

依次启动三种设备服务。设备服务启动时，会加载设备配置文件获取对应的信息，边缘网关首先会检测设备服务是否已经存在，如果不存在，则将其注册该设备服务并启动设备服务驱动。在测试设备服务时，在设备服务的设备配置文件中添加预定义设备，设备服务在启动时会加载设备信息，然后将预定义设备注册为设备服务的客户端。如图xx-xx是控制台输出的日志信息。

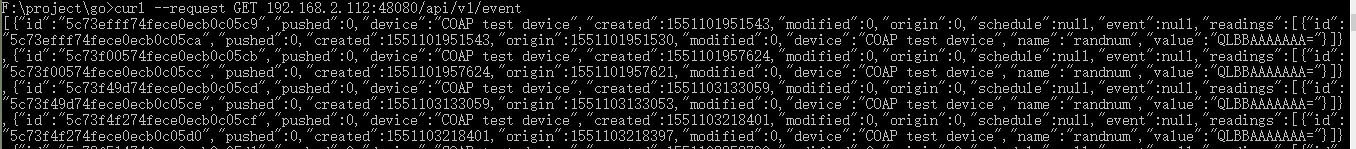
其中，A区的日志信息表示预定义设备已经通过设备服务注册到元数据服务。

为了验证设备服务的命令响应功能，在配置文件中预定义了一个定时事件：每30s调用一次命令控制服务，向CoAP设备服务发送一个GET请求，设备服务接到请求后，解析该命令并转发到底层节点。如图xx-xx中B区的日志信息所示。

底层节点按照设备服务定义的格式发送数据到网关。如图xx-xx的C区所示，当设备服务接收到底层节点发送数据后，

1. 核心数据服务：

核心数据服务主要功能实现对节点数据和节点读写事件的持久化存储，核心数据服务的IP地址是，端口号是。首先使用curl访问核心数据的API，查看测试设备服务时，节点发送的数据可对应的事件已经存储到核心数据中。如图xx-xx所示是多条对CoAP节点发送数据对应的时间事件，同时在存储之前，核心数据服务对节点数据进行了加密处理：



1. 元数据服务

元数据服务的主要功能是实现节点的资源及其属性的存储。使用curl工具访问如表所示的API可以查看节点的资源。

1. 应用服务层

为应用服务层添加客户端，首先启动MQTT 代理，以下是MQTT代理的配置:

Adress: 192.168.2.229

Port: 1883

publisher: EdgeXExportPublisher

user:hukfgtoh

password: uP6hJLYW6Ji4

topic: EdgeXDataTopic

首先使用curl工具发送POST请求，在应用服务层注册一个MQTT客户端，请求URL为http:// 192.168.2.112:48071/api/v1/registration，以下是请求内容：

{

"origin":1471806386919,

"name":"MQTTClient",

"addressable":{

"origin":1471806386919,

"name":"EdgeXTestMQTTBroker",

"protocol":"TCP",

"address":"192.168.2.229",

"port":1883,

"publisher":"EdgeXExportPublisher",

"user":"hukfgtoh",

"password":"uP6hJLYW6Ji4",

"topic":"EdgeXDataTopic"

},

"format":"JSON",

"enable":true,

"destination":"MQTT\_TOPIC"

}

在网关中启动一个http服务器，监听端口8111，在Freeboard中使用MQTT数据源插件，并修改插件的配置信息，就可以看到

## 本章小结

# 全文总结与展望

## 全文总结

针对xxx现状。

本论文的主要研究工作与成果如下：

（1）

（2）

（3）。

（4）。

## 研究展望

在本论文的工作基础上，xxx在以下几个方面进行优化和改进：

（1）。

（2）。

（3）。

致谢

三年的时光很快就过去，在硕士研究生即将毕业之际，首先我要感谢我的导师何顶新老师，他给了我们自由

其次我要感谢实验室的所有成员，感谢xxx学长

参考文献

1. 微服务架构与实践 王磊 2016.1 电子工业出版社
2. . An experimental study of a reliable IoT gateway Byungseok Kang, Hyunseung Choo∗ Department of Computer Science and Engineering, Sungkyunkwan University, accepted 15 April 2017 Elsevier是荷兰一家全球著名的学术期刊出版商
3. IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things Qian Zhu∗y, Ruicong Wang∗y, Qi Chen∗y, Yan Liu∗ and Weijun Qiny 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing
4. X. He, X.Y. Jing, “Design of data exchange system of smart home based on IOT,” Electronic Design Engineering, vol. 24, pp. 149-154, November 2016.
5. Uckelmann, Dieter, Mark Harrison, and Florian Michahelles. "An architectural approach towards the future internet of things." Architecting the internet of things. Springer Berlin Heidelberg, 2011.1-24.
6. Namiot, D., & Sneps-Sneppe, M. (2014). On M2M Software Platforms. International Journal of Open Information Technologies, 2(8), 29-33.
7. Namiot, D., & Sneps-Sneppe, M. (2014). On M2M Software. International Journal of Open Information Technologies, 2(6), 29-36.
8. Schneps-Schneppe, M., Namiot, D., Maximenko, A., & Malov, D.(2012, October). Wired Smart Home: energy metering, security, and emergency issues. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on (pp. 405-410). IEEE.
9. “ETSI Machine-to-Machine Communications info and drafts”
10. http://docbox.etsi.org/M2M/Open/ Retrieved: Jul, 2014.
11. Hassan, M., Zhao, W., & Yang, J. (2010, July). Provisioning web services from resource constrained mobile devices. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 490-497). IEEE.
12. Microservices <http://microservices.io/patterns/microservices.html> Retrieved: Aug, 2014
13. On Micro-services Architecture Dmitry Namiot, Manfred Sneps-Sneppe International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 9, 2014
14. 杨中书,刘臣宇.基于E-R模型的关系数据库设计方法[J].价值工程,2014,33(30):242-243..
15. 基于MLX90614的非接触式提问测量系统设计 软件导刊 2009.03
16. 基于MAX30102的穿戴式血氧饱和度检测系统 白鹏飞 刘强 2017-10-30 激光与红外
17. 睡眠监测手环在31例睡眠障碍患者中的应用研究 王司敏 秦浩2017-10-25 中国医疗设备
18. 新疆天文台天文观测数据传输日志系统设计与实现 朱艳 张海龙 天文研究与技术 2018-10-24
19. Sorapak Pukdesree,Vitalwonhyo Lacharoj,Parinya Sirisang. Performance Evaluation of Distributed Database on PC Cluster Computers using MySQL Cluster[J]. Lecture Notes in Engineering and Computer Science,2010,2186(1).
20. Ubaleht I. The design of relational database schemes based on the elementary relationships of attributes: Computing of the closure of a set of attributes for one type of relationship[C]. 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). IEEE; 2017:360-364.
21. 张长学,张伟,董智明. 移动推送技术面面观[J]. 移动通信. 2011(05) :21-27
22. Michael Bolin . Closure: The Definitive Guide: Google Tools to Add Power to Your JavaScript . O'Reilly Media 2010 9 27

附录 攻读学位期间发表论文目录

[1] 王静如, 何顶新. 基于Node.js的物联网网关系统的分层设计方法. 单片机与嵌入式系统应用. 2018.10