广州中医药大学

医学信息工程学院

本科毕业论文

题　　目：基于 ARM 的物联网系统的开发

姓　　名：张健强

学　　号：2014081006

专业年级：2014 级

指导老师：高理文

指导教师单位：医学信息工程学院

论文提交日期：　　年　　月　　日

论文答辩日期：　　年　　月　　日

广州中医药大学学位论文原创性声明

本人郑重声明：所呈交的学术论文，是个人在导师的指导下，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，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人或集体，均已在文中以明确方式标明。本人完全意识到此声明的法律后果由本人承担。

学位论文作者签名：

签字日期：

摘　要

物联网行业的终端、通信协议、技术标准、平台等不够统一，造成物联网的碎片化严重[1]。这是阻碍物联网终端连接量扩大，服务更广覆盖，数据的统一维护和运用的重要原因之一。加之很多公司都希望拥有自己的一整套物联网系统，以降低成本、数据私有化和提供定制化服务。物联网系统的实现就是为了处理这些问题。

本文旨在开发一个适用于多种终端、多种物联网标准协议的窄带物联网体系。该系统主要部分由云平台，单片机、ARM开发板以及相应的窄带物联网模块组成。本文详细介绍了该系统的组成和工作原理，并开放了相应的源代码。云平台使用经典的 RESTful 架构开发[2]，分为前端和后端。前端技术采用 Vue.js，UI 组件使用 elementUI 开源框架，而后端采用 Node.js，数据库采用 MongoDB，硬件终端采用 Stm32 做主控，bc95 做窄带物联网模块。整套系统的运行流程是 Stm32 向窄带物联网模块发指令，该模块启动基于 UDP/CoAP 通信协议的内置资源服务，与云平台建立注册连接，之后模块将采集的数据传送到云平台，云平台记录数据并下发数据到终端，达到终端与云平台数据交互的目的。

系统分离了软件、硬件的开发，各部分还针对性的有进行模块化开发，降低了开发者的开发难度和减少了开发周期，还支持了多种协议。开发者只要针对支持这些协议的硬件进行开发，就可以接入云平台，还可以针对性的进行定制化开发。

关键词： ARM　窄带　物联网　UDP　CoAP　云平台

ABSTRACT

The terminals, communication protocols, technical standards, and platforms of the Internet of Things industry are not unified enough, which causes fragmentation of the Internet of Things [1]. This is one of the important reasons that preventing IoT terminals' number from expanding and covering broader service, and maintaining and applying IoT data. In addition, many companies want to develop their own IoT systems to reduce costs, save data privately and provide customized services for their customer. The realization of the Internet of Things platform is precisely to resolve these conflicts.

This article aims to establish a Narrow Band IoT system that is applicable to multiple terminals and multiple IoT standard protocols. The main part of the system consists of a cloud platform, a microcontroller or an ARM development board, and a narrow band Internet of Things module. This article explains in detail the structure and working principle of the system. And I provide the open source. The cloud platform is developed using the classic architecture named RESTFul and is divided into front-end and back-end. The front-end technology uses Vue.js, UI component uses “elementUI” open-source framework, and the back-end uses Node.js, the database uses MongoDB, the hardware terminal uses Stm32 as master controller, and bc95 serves as the narrow band IoT module. The operating process of the entire system is that Stm32 sends commands to the Narrow Band module. The narrow band module starts the built-in resource service based on the UDP/CoAP communication protocol. And Then it registers and establishes a connection with the cloud platform. Afterwards, the narrow band module transfers the collected data to the cloud platform. The cloud platform receives and records data, then transfers data to the terminal to achieve the purpose of data exchange between the terminal and the cloud platform.

The system separates the development of software and hardware. Each

part is developed modularly, which reduces developer's development difficulty and reduces the development cycle. It also supports A variety of protocols. Developers can access cloud platforms as long as they develop hardware that supports these protocols, and they can also customize and develop them.

Keyword: ARM Narrow Band Internet of Things UDP CoAP Cloud Platform

目　　录

第 1 章 绪论

1.1 国内外研究现状和难题

2017 年 IoT Analytics 最新公布了一组数据：全世界的物联网平台持续高速增长，目前已经超过 450 家，比 2016 年增长近 30%。物联网平台有较为显著的行业化特征，工业制造、智慧城市和智能家居是应用最大的三大领域。其中，工业制造和智慧城市的典型特征是泛在连接的管理需求突出，而智能家居则是物联网体验优化的先锋场景。IoT Analytics 还指出，虽然物联网平台市场的碎片化与领先平台在高速增长，但只有 7%的物联网平台每年可获得超过 1000 万美元的营收。

概括起来，物联网平台的市场现状就是这几个词：市场高涨、行业分化、结构碎片、前景不明[3]。

物联网系统应该是终端连接、终端管理、数据采集分析、应用定制等多个方面能力的组合。而从能力综合的角度来看，BCG 认为只有 14%的平台能够提供全方位的能力，姑且称之为物联网综合平台[4]。来自不同领域的参与方，都在依托自己的核心能力优势，意图构建一个新的平台枢纽。以运作模式来分类，可以分为两类：定制化服务与标准化服务。顾名思义，定制化服务是按照自定义的需求对物联网平台针对性修改。标准化服务则是云平台提供统一的接口和套件，客户自行调用。在物联网综合平台在平台生态的打造上，面临待破局的难题：1.多边市场难题。2.边际成本问题。3.生态流转问题。物联网平台的未来在多边市场、边际成本以及生态流转三个问题解决之前，物联网的“平台”特性都将受到限制，B 端定制化市场特征仍将占据优势。这也意味着出身于传统 ICT 领域的软件服务商与垂直行业巨头在市场中获得更多的竞争地位。

目前低功耗物联网业务所面临的挑战主要包括：在网络和终端方面，模组成本高，耗电量大，网络覆盖不理想；在运营方面，增量不增收，业务碎片化。窄带物联网，即 NB-IoT 能有效解决当前低功耗应用方面遇到的矛盾。 NB-IoT 作为物联网授权频谱的主流技术之一[5]，具备远距离、低功耗、低速度的特性，适用于智慧城市以及工业、农业等多方面物联网应用的场景。国内外影响力比较大的四个标准组织 3GPP、GSMA、GCF、CCSA 都在积极推动，制定 NB-IoT 的相关技术标准。国内三大运营商：移动、联通、电信也积极跟进，华为，ofo，摩拜等多家企业都在结合 NB-IoT 研发相关产品。

所以针对热门的NB-IoT开发的物联网系统，拥有较好的应用前景。

1.2 目的与意义

硬件、连接、平台及应用服务为物联网产业生态的四个主要层次，各层之间紧密耦合、互为支撑。其中，硬件和连接作为底层支持，平台是核心组成与底层相配合，价值兑现则要凭借应用服务。作为物联网产业生态中的关键组成部分，物联网平台是万物互联时代软硬结合的枢纽位置，一方面肩负管理底层实体硬件、支撑并赋能上层应用服务的重任，另一方面，汇聚硬件实体属性、感知信息、用户身份、交互指令等静态及动态信息，不断爆发式增长的数据[6]，正在驱动其成长为实体世界的虚拟镜像。由硬件和物联网云平台构成的系统毋庸置疑是至关重要的。

NB-IoT 是第三代伙伴计划（3GPP）针对低功耗广域物联网业务推出的一种新型基于窄带（200KHz）的蜂窝物联网技术标准[7]，是专门为低功耗、广覆盖物联网业务设计的。作为低功耗广域物联网无线接入技术，NB-IoT (Narrow Band Internet of Things)具有广覆盖、大连接、低功耗、低成本 4 个方面的优势[8][9]。

由于 NB-IoT 具有的多方面优点，所以本文针对新兴的 NB - IoT 技术，通过单片机控制 NB-IoT 模组与 NB-IoT 基站进行通信，与云平台完成连接。该云平台构建一个针对 NB - IoT 相关协议的物联网平台，北向提供 RESTful API 给行业应用，南向采用 CoAP 协议完成 NB-IoT 设备管理及业务数据交互[2][10]。该系统适配了 NB 网络，并针对性开发了可接入云平台的硬件平台，降低了其他开发者的开发难度，确保了系统的开放性和可应用性。

1.3 本文创新点

第一，支持专为物联网设计的基于 REST 构架的 CoAP 协议。第二，针对窄带物联网调节总体的通信流程。第三，极大减少了二次开发的难度。只需要参考给定的硬件嵌入式开发例程，将之移植到自己的嵌入式平台，再部署该套系统的云平台，即可直接完成整套系统的开发，大大降低了开发的难度。第四，代码开源，可供私人开发者或者企业需求二次开发。

第 2 章　物联网系统总体架构及相关靠技术

2.1　物联网系统系统架构

2.1.1 系统需求分析

一个完整的物联网系统包括三部分，感知层，网络层和应用层[11]。感知层主要作用是使用各类传感器采集环境数据，网络层用于传输数据，主要由无线网络、有线网络和处理数据的云平台构成该层，应用层是各种具体的物联网服务。

由于本文提出的物联网系统目标是建立感知层和网络层，应用层可按需进行扩展，所以该物联网平台需要建立一个可视化的 Web 云平台[12]，按照物联网协议与设备终端直接建立通信，并管理设备终端；需要支持 HTTP 通信保证云平台的正常运行，还需要支持 CoAP/UDP 协议访问窄带物联网设备终端。平台内部需要根据用户的操作传递消息，以达到管理设备，订阅推送设备数据和其他信息显示给用户的目的。

2.1.2 系统的总体功能和框架设计

在上一节需求分析的基础上，对窄带物联网的总体功能和框架进行设计，如图 2.1 所示，该物联网系统包括两大部分，窄带物联网设备终端，云平台端[13]。

云平台端提供服务和显示，与设备终端通信，并将相关信息展示到云平台前端。其中又细分为通信服务、数据存储、SocketIO 消息推送。通信服务主要用于窄带物联网设备终端注册进入平台，与平台就建立连接通道和双方的数据传输。数据存储是将用户信息，设备终端有关信息等存储进 MongoDB。而 SoocketIO 则是前端和后端建立通信连接的另一个通道，最主要用来将服务端收到的设备信息推送到前端显示。

窄带物联网设备终端又细分为三个部分，一个是 MCU 做主控，所有的硬件的动作流程全部由 MCU 控制。第二部分是 NB 模块，该模块与 MCU 使用串口协议通信。MCU 通过串口向 NB 模块发送 AT 指令完成入网，和云平台注册并通信和数据传输的过程。第三部分是传感器感知环境部分，主要是通过 MCU 控制采集环境数据返回给 MCU。

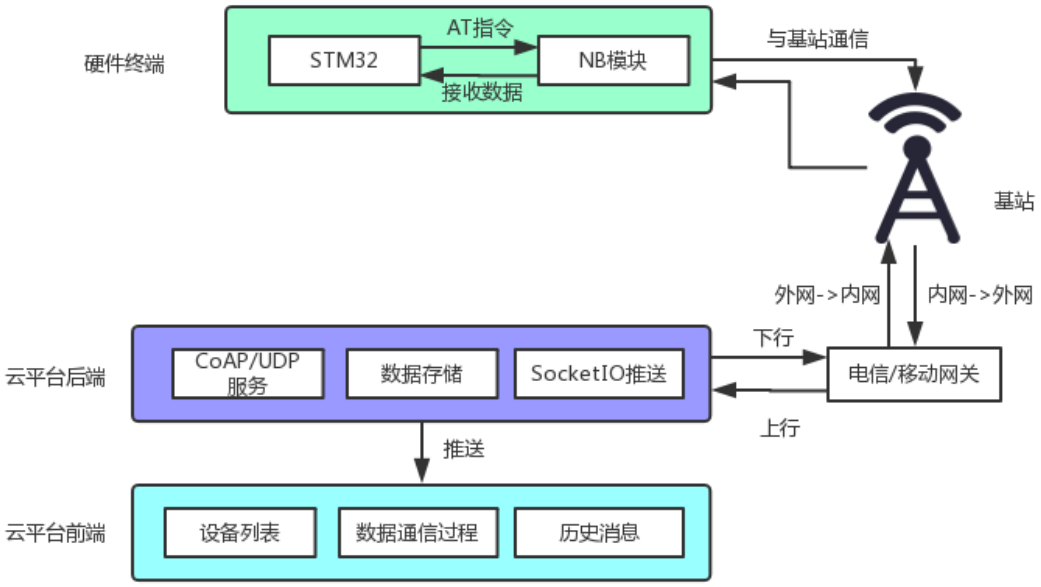


图 2.1 系统框架

整个系统的通信流程图 2-2 所示。

(1)进入云平台，根据需要创建一个通过 CoAP 或 UDP 协议通信的设备，之后进入数据调试页面，此时服务端已建立相应的 CoAP 或 UDP 服务。

(2)如果设备终端有连接传感器，则先读取传感器的数据，否则 MCU 直接通过串口向 NB 模块(BC95)发送一系列 AT 指令，模块会根据 AT 指令与基站无线通信，之后注册到网络，此时入网成功。之后 MCU 根据需要继续发送指令指定 NB 模块使用 CoAP 或 UDP 通信协议。

(3)如果使用 UDP 协议通信，则设备终端需要先发送一个云平台数据调试页面指定的注册包给云平台，云平台收到后判断收到的注册包是否正确，如果正确则设备终端成功注册进云平台，接下来双方可直接进行数据通信。

(4)如果使用 CoAP 协议通信，则设备终端不需要先发送注册包给云平台。可通过 MCU 发送 AT 指令设置云平台的 ip 和 port。之后先通过设备终端发送任一数据到云平台，该过程云平台会执行一系列的响应和请求操作，之后设备终端成功注册到云平台，且云平台订阅了设备终端的资源，此时，双方可直接进行数据通信。需注意的是如果采用 CoAP 通信，则必须是先由设备终端主动向云平台发起通信，而如果一段时间不通信后，NB 模块会进入深度睡眠状态，此时唤醒模块也需要设备终端主动向云平台发起通信。

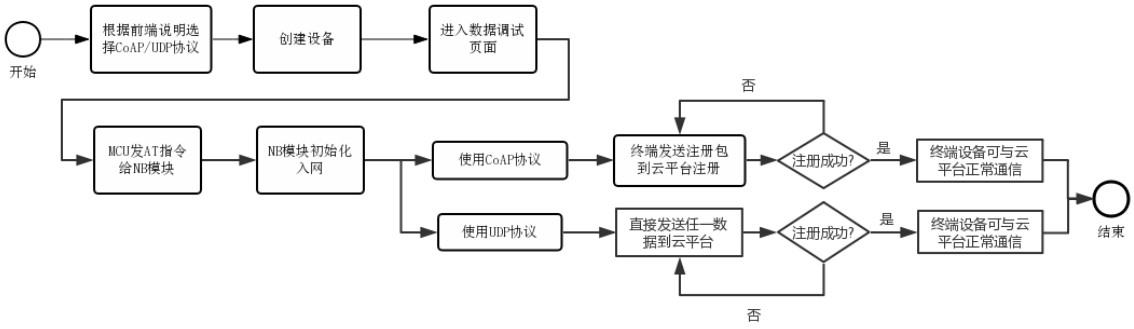


图 2.2 总体流程

2.1.3 总体技术架构

系统架构分为软件和硬件两部分。

下位机即硬件采用 STM32 作为 MCU，本文所用的 STM32 是 STM32F103C8T6 型号，所开发的模块库可针对不同 STM32 型号稍做修改即可移植到其他型号上面。使用的无线模块是窄带物联网(NBIoT)技术开发的 BC95-b8 模块。

上位机云端平台由前端后台以及数据库构成。前端采用 Vue.js 开发，UI 集成模块采用的是饿了么的开源框架 Element UI。后台采用 Node.js 技术开发，也使用了一个框架名为 Express。数据库方面采用了非关系数据库 NoSql 系列下的 MongoDB，数据主要用来存储用户的账号信息、设备详细信息、数据收发的记录。每种记录都是由一个 Collections(文档)构成。由于前后端都部署在一台服务器上，所以它们之间通信需要跨域处理。

2.2 窄带物联网及其协议

2.2.1 NB-IoT

窄带物联网（Narrow Band Internet of Things, NB-IoT）成为物联网的一个越来越重要的分支。NB-IoT 是作低功耗广域网(LPWAN)里面的一种，是近几年逐渐兴起的 IoT 技术，支持低功耗设备在广域网的蜂窝数据连接，只消耗大约 180KHz 的带宽，具有广覆盖，低成本，低功耗，低速率的特性，适用于智能抄表、智能停车、智能垃圾回收等智慧城市应用[14]。

NB-IoT 适合应用于要求待机时间超长、对网络连接要求较高的设备。NB-IoT 有望解决当前物联网标准不统一、终端成本高、接入能力

不足等核心问题，从而带来物联网（尤其是低功耗广覆盖接入市场）的快速发展[15]。

2.2.2 CoAP 协议

1. CoAP 的定义和起源

受限制的应用协议(Constrained Application Protocol)的简称为CoAP。在低功耗受限制的设备里面实现 TCP 和 HTTP 协议是超过要求的。为了接入低功耗设备，小巧的 CoAP 应运而生。

2. CoAP 针对物联网所做的设计和改进

CoAP 是 TCP/IP 协议族的一员，最小的数据包只有 4 个字节。它具有与 HTTP 类似的特征，资源抽象、REST 式交互以及可扩展的头选项等是它的核心内容。由于 HTTP 基于 TCP 传输协议，采用点对点的通信模型，不适合于推送通知服务，而且对于受限设备（如 8-bit 微处理器）HTTP 过于复杂。

CoAP 协议基于 REST 构架[16],REST 是指是一种 Web 服务实现方案，是互联网资源访问协议的一般性设计风格。为了克服 HTTP 对于受限环境的劣势，CoAP 既考虑到数据报长度的最优化，又考虑到提供可靠通信。一方面,CoAP 提供 URI,REST 式的方法如 GET,POST,PUT 和 DELETE,以及可以独立定义的头选项提供的可扩展性。另一方面，CoAP 基于轻量级的 UDP 协议，并且允许 IP 多播。而组通信是物联网最重要的需求之一，比如说用于自动化应用中。CoAP 还定义了带有重传机制的事务处理机制，弥补了 UDP 传输的不可靠性，并且提供资源发现机制，并带有资源描述。

考虑到资源受限设备的低处理能力和低功耗限制，CoAP 重新设计了 HTTP 的部分功能以适应设备的约束条件，而不是盲目的压缩 HTTP 协议。另外，为了使协议适应物联网和 M2M 应用，CoAP 协议改进了一些机制，同时增加了一些功能。图 2.3 比较了 HTTP 和 CoAP 的协议栈。CoAP 和 HTTP 在传输层有明显的区别[17]。HTTP 协议的传输层采用了 TCP 协议，而 CoAP 协议的传输层使用 UDP 协议，开销明显降低，并支持多播。

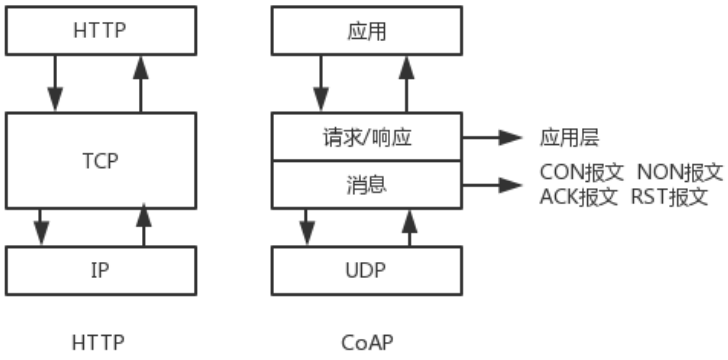


图 2.3 HTTP 和 CoAP 协议栈

2.3　硬件平台

2.3.1 STM32 平台的介绍和优势

1. STM32 产品介绍

STM32 系列基于专为要求高性能、低成本、低功耗的嵌入式应用专门设计的 ARM Cortex-M0，M0+，M3，M4 和 M7 内核的 32 位处理器。按内核架构分为不同产品：

主流产品（STM32F0、STM32F1、STM32F3）、超低功耗产品（STM32L0、STM32L1、STM32L4、STM32L4+）、高性能产品（STM32F2、STM32F4、STM32F7、STM32H7）。

STM32 型号的说明：以 STM32F103RBT6 这个型号的芯片为例，该型号的组成为 7 个部分，其命名规则如下(以 STM32F10xAyBz 为例)：

(1)STM32 代表 ARM Cortex-M 内核的 32 位微控制器。

(2)F 代表芯片子系列。

(3)10x 后三位数字代表增强型系列。

(4)A 这一项代表引脚数。

(5)y 这一项代表内嵌 Flash 容量。

(6)B 这一项代表封装。

(7)z 这一项代表工作温度范围。

它和 8 位单片机最大的不同是，它不仅可以使用寄存器进行编程，还可以使用官方提供的库文件进行编程，这样便于移植，也降低了很多开发成本。

本文采用 STM32F103C8T6 作为底层硬件进行嵌入式开发。它是一款采用 Cortex-M3 内核，CPU 最高速度达 72 MHz，具有 48 个 I/O 口，64KB

Flash 存储器，采用官方推荐的 HAL 库进行嵌入式软件开发。

2. STM32 的优势和选用该平台的原因

STM32 的优异性体现在如下几个方面：

(1)超低的价格。用 8 位机的价格购买 32 位机，是 STM32 最大的优势。

(2)超多的外设。FSMC、TIMER、SPI、IIC、USB、CAN、SDIO、ADC、DAC、RTC、DMA 等都是 STM32 拥有的外设及功能，具有极高的集成度。

(3)丰富的型号。STM32 仅 M3 内核就拥有 F100、F101、F102、F103、F105、F107、F207、F217 等 8 个系列上百种型号，具有 QFN、LQFP、BGA 等封装可供选择。同时 STM32 还推出了 STM32L 和 STM32W 等超低功耗和无线应用型的 M3 芯片。

(4)优异的实时性能。84 个中断，16 级可编程优先级，还可以引入 RTOS，并且所有的引脚都可以作为中断输入。

(5)杰出的功耗控制。STM32 各个外设都有自己的独立时钟开关，可以通过关闭相应外设的时钟来降低功耗。

(6)极低的开发成本。STM32 的开发不需要昂贵的仿真器，只需要一个串口即可下载代码，并且支持 SWD 和 JTAG 两种调试口。SWD 调试可以为你的设计带来跟多的方便，只需要 2 个 IO 口，即可实现仿真调试。

2.3.2 几种不同无线标准的比较

目前主流的无线标准[18]主要有 NB、蓝牙、WIFI、Zigbee、3G/4G。以下分五个方面来进行比较说明：速率，传输距离，功耗，传输的数据量，传输形式。

表 2-1 几种不同无线标准的比较

无线标准	速率	传输距离	功耗	传输的数据量	传输形式
WIFI	速度很快(300Mbps)	传输距离短(200-300m)	高功耗(10-50mA)	一次性可传输的数据量大	只需要一个设备端，连接上 WiFi 网络即可
Blueto oth	速度较高(1Mbps 左右)	传输距离超短(2-30m)	低功耗	一次性可传输的数据量大	两个设备端建立 socket 后进行数据传输
3G/4G	速度飞快	传输距离超	超高功耗	一次性	需要一个设备

		长		可传输的数据量非常多	端连接到附近的基站后传输数据。
Zigbee	速度一般(250kbps)	传输距离短(50-300m)	较低功耗(5mA)	一次性可传输的数据量较少	多个设备组网通信
NB	速度一般	传输距离超长	超低功耗	一次性可传输的数据量较少	需要一个设备端连接到附近的基站后传输数据。

其中传输距离上看，NB>=3G/4G>WIFI>=Zigbee> Bluetooth；功耗比较上， 3G/4G>WIFI>Zigbee>Bluetooth>NB。 由于 NB 网络具有以上的特点，所以特别适合用来应用于一些远距离，低功耗，传输数据少的场景上，如智能泊车、自行车联网防盗、车联网、智慧城市、智慧建筑、环境监控等应用。

2.3.3 串口通信

串行通信是一种全双工通信，一般传输过程是每个字节，每个字节的位一个一个进行的，并且传输一个字符时，总是以“起始位”开始，以“停止位”结束。在进行传输之前，双方一定要使用同一个波特率设置。波特率就是每秒钟传输的数据位数。

常用的两种基本串行通信方式包括同步通信和异步通信，一般使用异步通信，它规定传输的数据格式由起始位、数据位、奇偶校验位和停止位组成。

异步通信是指收发双方各自使用自己的时钟，无需用同一个时钟信号同步，而同步通信需要收发双方同步各自的时钟再进行通信。它以数据帧的方式传输数据，为了能让接受方正确接受一个帧，需要在帧首加上起始位，同时，为了让接收方正确识别一个完整的帧，需要在帧尾加上停止位，代表一帧的结束。因此，只要双方遵守这个协议，他们就可以在任意时刻发送和接收一个帧，发送方可以在发送完第一个帧后，等 1 个小时再发送第二个帧，而接受方会不断采集线路 RX 引脚的电平，如果突然采集到下降沿，由 1 变为 0 ，则接收方确认传送方有新的数据要传输过来，于是开始以设定的波特率获取数据。

第 3 章　云平台组成和工作原理

3.1　系统架构设计，支持的协议和工作原理

由于云平台针对的是窄带物联网控制与通信，所以需要支持主流的窄带物联网协议，包括 UDP，CoAP 等，考虑支持 MQTT、TCP 协议。

工作原理：

上位机云端建立相应的 UDP 服务或 CoAP 服务器。当 NB 模块向云端发出数据，云端根据实际协议与模块建立通信 Socket，接收数据并将之存储到数据库，之后可发送下行数据。

云平台的系统架构主要由前端 UI，后端，中间跨域件，数据库等组成，如图 3-1 所示。后面几节会详细介绍这几部分内容。

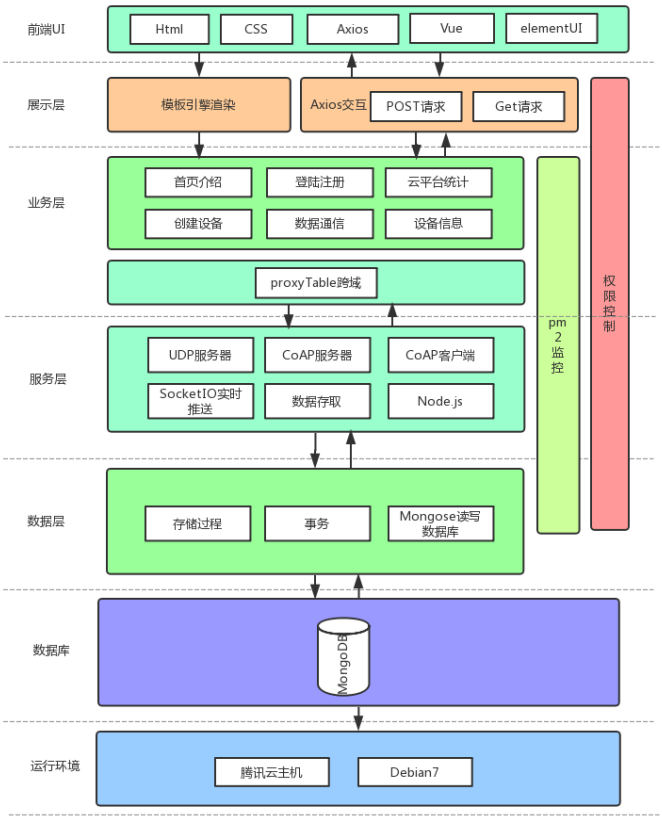


图 3-1 云平台系统架构设计

3.2 云平台前端设计

3.2.1 前端功能和架构的选用

在设计前端页面时，需要考虑前端良好可用的结构，界面适当的美化以及和后端交互的易用性。前端的主要功能是展示一些介绍信息，用户登陆注册页面，云平台后台控制页面，包括显示设备总体信息，显示设备的历史信息，创建设备，显示云平台数据的收发，显示用户的基本信息。

Vue 是一款友好的、多用途且高性能的用于构建用户界面的渐进式、响应式框架[19]。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。基于以上的功能需求和 Vue 优异的性能，前端技术架构采用 Vue.js。

3.2.2 前端功能介绍

1 首页展示基本信息

首页分为三部分构成，如图 3-2 所示。第一，云平台的介绍信息，包括介绍 NBIot 的知识点，CoAP 协议的解读，所使用的 STM32 平台，NB 模块等硬件的介绍，分析 NBIoT 和其他无线技术的异同以及 NBIoT 在特定领域的优异之处。第二，提供云平台的使用说明，包括操作说明，开放的 API 的说明，提供这些文档说明。第三，提供一个注册和登陆的入口，给每位用户有自己的独立的操作后台。

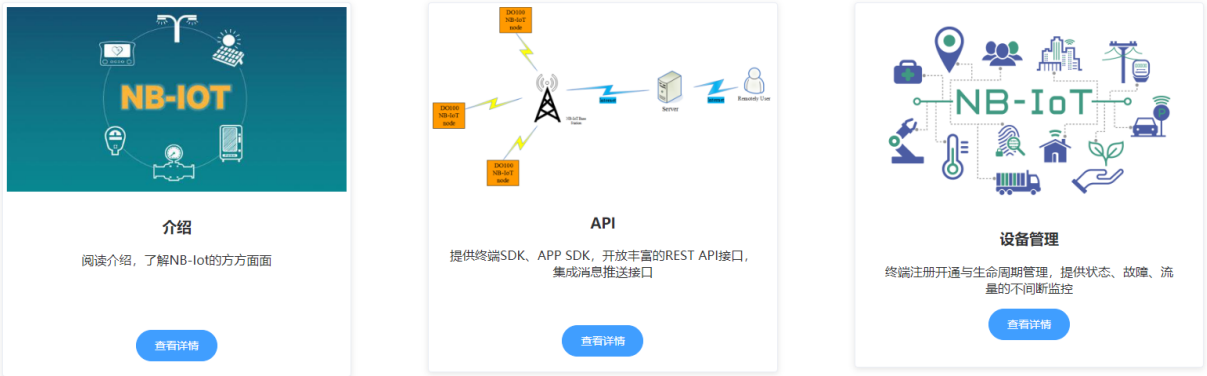


图 3-2 首页介绍

2 注册和登陆功能

(1) 注 册 功 能

如 图 3-3 a)所 示 是 注 册 页 面， 该 页 面 有 三 个 基 本 的 输 入 框： 账 号， 密 码， 确 认 密 码。 输 入 框 添 加 了 逻 辑 校 验， 三 个 输 入 框 局 部 可 不 为 空， 账 号， 密 码 长 度 必 须 在 6 - 20 个 字 符 之 间。 确 认 密 码 框 和 密 码 框 表 单 填 写 必 须 一 致， 当 这 些 条 件 满 足， 则 可 以 提 交 填 写 的 表 单， 将 注 册 信 息 发 送 到 后 台， 记 录 到 数 据 库， 当 前 端 收 到 后 端 响 应 的 用 户 名 和 状 态 时， 注 册 页 面 显 示 成 功 如 图 3-3 c)所 示， 并 跳 转 到 登 陆 界 面。 当 输 入 不 符 合 输 入 框 的 逻 辑 校 验 时 则 报 错。 如 图 3-3 b)所 示。

a） 注 册 界 面　　b） 校 验 错 误

c） 注 册 成 功

图 3-3 注 册 功 能

(2)登 陆 界 面 是 同 样 类 似 的 设 计， 如 图 有 3-4 a)所 示。 该 页 面 有 两 个 输 入 框， 也 同 样 添 加 了 逻 辑 校 验， 输 入 账 号 和 密 码 后 点 击 登 陆 即 可。 如 果 不 符 合 校 验 要 求 则 报 错， 如 图 3-4 b)所 示。 登 陆 成 功 页 面 弹 出 提 示， 如 图 3-3 c)所 示。 否 则 页 面 会 提 示 错 误， 如 图 3-4 d)所 示。

a) 注册页面　　b) 校验错误

d) 登陆成功　　c) 登陆错误

图 3-4 登录功能

(3)该模块的重点和难点

登陆和注册的请求接口采用了 Vue 的 Axios 模块。Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 Node.js 中。本文主要用该模块完成从 node.js 创建 http 请求、拦截请求和响应、转换请求数据和响应数据、自动转换 JSON 数据等工作，使之能够提交请求给后台以及接收该请求相应的响应数据，达到前后端数据交互的目的，是该系统前端方面的重点和难点。以下以注册接口为例说明 Axios 的使用。

表 3-1 关于注册的 Axios API

```
1. axios.post("/users/register", {
2.   userName: this.loginInfo.userName,
3.   userPwd: this.loginInfo.userPwd
4. }).then((response) => {
5.   let res = response.data;
6.   if (res.status == "0") {
7.     this.loginInfo.nickName = res.result.userName;
8.     this.$notify({
9.       title: '成功',
10.        message: this.loginInfo.nickName + '注册成功',
11.        type: 'success'
12.     });
13.     this.$router.push('/users/login');
14.   } else {}
```

```
15. })
```

分析：

通过这部分代码，读者可以清晰了解到如何利用 Axios 中的 API 来发送请求和接收响应，其中需要注意的是：

1. 第 1 行表示用 Axios 发送一个 post 请求，请求的路径是/users/register，2-3 行是请求的参数，请求参数用 JSON 数组来表示。

2. 第 5-14 行是对接收到的响应做解析，其中第 6 行判断接收到的 status 是否是 0，如果是则表示注册成功，第 7 行记录用户名信息并在第 8-12 行发出通知，之后通过第 13 行跳转到登陆页面。

3 后台设备管理功能

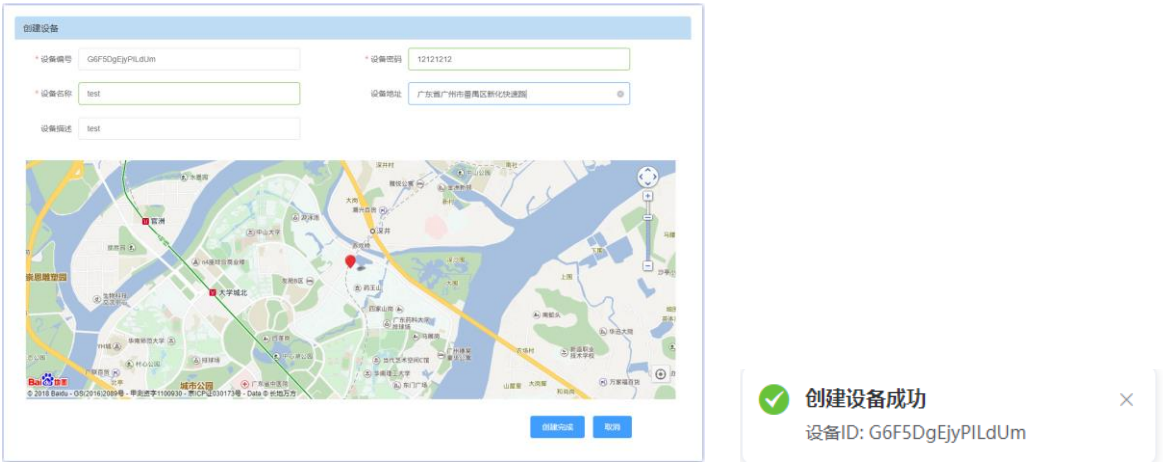
该功能分为 3 部分。

(1) 第一部分是后台主控，相当于后台主页，展示了当前设备的基本信息，包括设备数量，上行消息数量，下行消息数量，用户数量。后期增加设备在线情况统计，近段时间的流量统计，UDP 数据流量统计，CoAP 数据的流量统计。如图 3-5 所示。

图 3-5 后台主控界面

(2)第二部分是设备管理部分，也就是云平台的核心部分。该部分分为三个小功能，第一个小功能是创建设备。如图 3-6 a)所示，进入创建设备页面后系统会自动生成 15 位设备编号，此时还需要填写设备密码，设备名称，另外的设备地址和设备描述为选填。此处应注意如果设备使用 UDP 通信时，可使用云平台自动生成的设备 ID，也可手动填写 15 位设备 ID，但必须是英文加数字的组

合；而如果设备使用 CoAP 通信时，需填写设备的 IMEI 码，否则通信不成功。如何查询设备的 sim 卡请查看附录 A。填写完毕后点击创建完成则会等待后端响应，如果后端响应成功则提示成功，如图 3-6 b)并跳转到设备列表页面。



a） 从创建设备页面　　b）创建设备成功

3-6 创建设备

第二个小功能是查看设备列表，该页面展示了该用户创建的所有设备的信息。如图 3-7 所示。

图 3-7 查看设备列表页面

这些信息包括了设备 ID 设备名称，设备密码，设备所在位置，状态(在线/离线)，和可以进行的操作。这些操作包括数据传输，编辑和删除，其中编辑功能是跳转到创建设备页面，此时页面是该行设备的信息，可以在此基础上修改，修改后再次点击创建完成即可。删除功能是指点击每一行的删除按钮可以永久删除该行的设备信息。而数据传输功能是和终端设备交互的核心功能，该部分功能可具体分为支持 UDP 的数据传输和支持 CoAP 协议的数据传输。如果该行设备对应的是使用 UDP 通信的设备的话，点击数据传输按钮，获取该设备的所有信息并进入数据传输调试页面，如图 3-8 所示。

图 3-8 数据调试页面

进入该页面后会同时会向后端提交一个 post 请求，主要作用是建立后端的 CoAP/UDP 服务，具体如下：

表 3-2 关于数据调试的 Axios API

```
1. Bus.$on('getDeviceInfo', (row) => {
2.   this.$nextTick(function () {
3.         global.DEVINFO = row;
4.   });
5.});
6.this.deInfo = global.DEVINFO;
7.axios.post("/debugDevice?deviceID="+this.deInfo.deviceID+"&devicePwd=" + this.deInfo.devicePwd).then((response) => {
8.   this.status = response.data.status;
9.   if (this.status == "1") {
10.      this.$notify("设备"+ this.deInfo.deviceID+"已上线");
11.   } else if (this.status == "0"){
12.      this.$notify("注册包发送错误");
13.   } else if (this.status == "2") {
14.    this.$notify("收到消息:"+ response.data.receData);
15.      });
16.   }
17.})
```

分析：

通过这部分代码，读者可以清晰了解到进入数据调试页面后会先获取设

备信息，然后提交一个 post 请求，之后根据收到的后端返回的数据进行判断并通知用户，其中需要注意的是：

1.点击数据调试按钮时会触发一个 Bus 实例发出一个'getDeviceInfo'事件信号。

2.第 1-5 行表示用使用监听 Bus 实例上的自定义事件'getDeviceInfo'，并将其赋值给全局变量。第六行将全局变量再赋值给当前页面实例的 deInfo 数据。

3.第 7 行根据设备的 ID 和密码提交一个 post 请求，请求 url 以设备 ID 和设备密码构建了一个 query 参数。第 8 行接收后端返回的数据。第 9 行到第 17 行根据收到的数据进行判断和通知用户。

4.需注意使用 UDP 通信时需要先发送一条注册包，注册成功后终端设备才可与云平台通信。

该部分功能需要注意的是，如果终端设备使用 CoAP 协议与云平台后端通信，那么设备 ID 必须是设备的 IMEI 号，具体可以查看附录 A。设备使用 CoAP 与云平台后端通信时，不需要发送注册包到后端，可在输入框输入需要发送的数据点击发送按钮直接发送，CoAP 的通信过程会比较慢，可以稍等几秒即可看到。具体的原理和通信过程参考第 4 章。

(3)第三个小功能是查看历史信息，如图 3-9 所示。该部分会显示终端设备和云平台通信过程产生的数据这些信息包括信息的序号、设备 ID，通信产生的数据、数据方向，通信数据采用的协议，数据产生的时间。

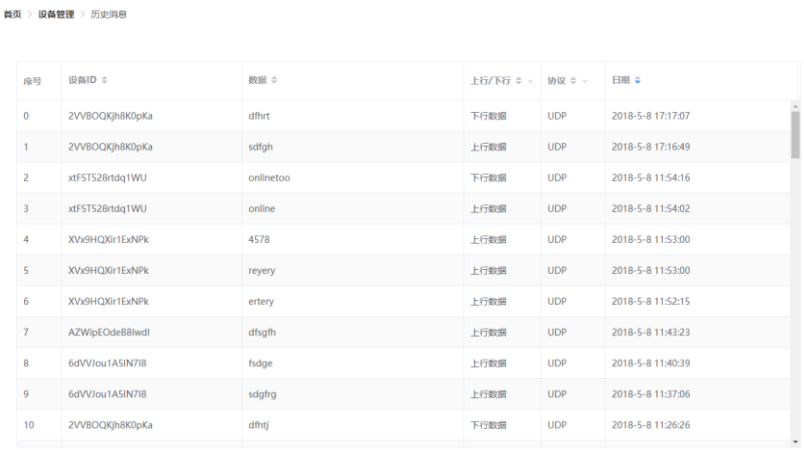


图 3-9 设备历史信息页面

第三部分是显示用户的基本信息，即账号，提供可以填写邮箱，手机等的入口

和修改密码的入口。

3.3　云平台后端设计

3.3.1 后端功能和架构的选用

后端主要是配合前端，需要操作 MongoDB 数据库，处理 UDP 相关逻辑，处理 CoAP 相关逻辑，开启 SoketIO 和前端交互，而且具有一定的实时性要求。

Node.js 具有以下特性[20][21]：

Node.js 库的异步和事件驱动的 API 全部都是异步就是非阻塞。它主要是指基于 Node.js 的服务器不会等待 API 返回的数据。服务器移动到下一个 API 调用，Node.js 发生的事件通知机制后有助于服务器获得从之前的 API 调用的响应。

非常快的内置谷歌 Chrome 的 V8 JavaScript 引擎，Node.js 库代码执行是非常快的。

单线程但高度可扩展 - Node.js 使用具有循环事件单线程模型。事件机制有助于服务器在一个非阻塞的方式响应并使得服务器高度可扩展，而不是创建线程限制来处理请求的传统服务器。Node.js 使用单线程的程序，但可以提供比传统的服务器(比如 Apache HTTP 服务器)的请求服务数量要大得多。

Node.js 还提供了各种丰富的 JavaScript 模块库，它极大简化了使用 Node.js 来扩展 Web 应用程序的研究与开发。

基于以上的特性和功能需求，Node.js 具有非常好的服务器性能，而且具有开源的 UDP、CoAP 库，异步属性使得与物联网部分要求实时的需求相得益彰，所以服务端代码采用了 Node.js。

3.3.2 后端主要模块的详细设计

1 用户登陆和注册模块

该模块首先初始化 MongoDB 数据库，连接成功后，监听路由发送过来的 post 请求，如果请求的路由是/users/login，则进入登录验证逻辑。首先取得 post 请求中的 userName，userPwd 参数，之后调用 API 在 mongoDB 数据库的 Users Collections 进行查询，如果查到该条记录，则将记录取出来，并将用户名和状态放在一个 json 数组里面，将该数组返回。如果请求的路由是/users/register，则进入注册逻辑。首先取得 post 请求中的 userName、userPwd 参数，之后调用 API 将用户数据插入到数据库的 Users Collections。

2 创建设备后端接口

如果请求的路由是/createDevice，则进入创建设备逻辑。首先取得 post 请求中的设备 ID、设备密码、设备名称、设备地址、设备描述等信息，将这条信息按 MongoDB DeviceDetailInfo Collections 的格式进行格式化，之后插入到改 Colletions，如果插入成功则返回设备 ID 和成功状态码。

3 设备列表后端接口

如果请求的路由是/deviceList，则进入读取设备列表逻辑。调用 get_device_list 方法，调用 MongoDB API 查询 DevicesDetailInfo Collections 所有数据，并将所有数据通过 JSON 格式化后返回给客户端。

4 设备历史信息后端接口

如果请求的路由是/deviceHistory，则进入读取设备历史信息逻辑。调用 get_device_list 方法，调用 MongoDB API 查询 DevicesInfo Collections 所有数据，并将所有数据通过 JSON 格式化后返回给客户端。此接口和设备列表后端接口逻辑类似。

5 设备数据传输调试接口

如果请求的路由是/debugDevice，进入设备数据传输调试接口逻辑。之后判断 post 请求的 query 参数的设备 ID 是否全为数字，是则开启 CoAP 服务器，进入 CoAP 服务器的通信流程；否则开启 UDP 服务器，进入 UDP 服务器的通信流程。关于云端和设备终端的 UDP 和 CoAP 通信流程参考第 4 章。

3.4 数据库设计

3.4.1 数据库选用

MongDB 是一个高性能，开源，无模式的文档型 NosQL 数据库。其特点是查询速度快、高并发、具有敏捷性和可扩展性。由于本文设计的物联网云平台后期需要扩展支持 MQTT，TCP，且 NB 物联网终端设备非常多，有一定的并发行要求，还有一定的实时性要求，所以 MongDB 适合用来做云平台的数据库存储数据。

3.4.2 数据库设计

MongoDB 中的集合类似于关系型数据库中的表，mongodb 中的文档类似于关系型数据库中的行。关系型数据库中的一条记录就是一个文档，是一个数据结构，由 field 和 value 对组成。MongoDB 文档与 JSON 对象类似。字段的值有

可能包括其它文档、数组以及文档数组。多个文档组成一个集合，多个集合组成一个数据库。

本文使用了 MongoDB 的一个数据库 NBIot，数据库有四个集合，以下详细介绍：

表 3-3 DevicesDetailInfo 集合存储设备的详细信息。

Collection(集合)	Field	数据类型	默认值	说明
DevicesDetailInfo	_id	ObjectId	系统自动生成	相当于主键
	deviceID	String	“”	设备 ID
	devicePwd	String	“”	设备密码
	deviceName	String	“”	设备名称
	deviceAddr	String	“”	设备地址
	deviceDetail	String	“”	设备详细描述

表 3-4 DevicesInfo 集合存储设备数据传输的详细信息。

Collection(集合)	Field(列)	数据类型	默认值	说明
DevicesInfo	_id	ObjectId	系统自动生成	相当于主键
	deviceID	String	“”	设备 ID
	deviceData	String	“”	设备上下行数据
	dataDir	String	“”	数据方向
	protocol	String	“”	通信协议
	date	String	“”	数据产生时间

表 3-5 Users 集合存储用户的详细信息。

Collection(集合)	Field(列)	数据类型	默认值	说明
Users	_id	ObjectId	系统自动生成	相当于主键
	userId	String	“”	用户 ID
	userName	String	“”	用户名
	userPwd	String	“”	用户密码

表 3-4 Ids 集合主要用来配合 Users 集合递增生成用户 ID。

Collection(集合)	Field(列)	数据类型	默认值	说明

	_id	ObjectId	系统自动生成	相当于主键
	name	String	“”	用户名
Ids	id	String	“”	用来生成用户ID
	amount	Int32	0	用户总数

第 4 章　硬件平台设计

4.1　总体架构和工作原理

硬件平台共包括了三个部分，MCU 主控，NB 模块和传感器感知。如图 4-1 所示。

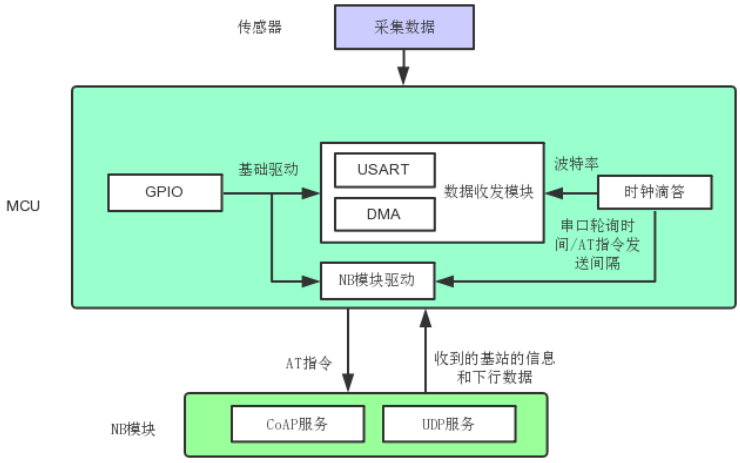


图 4-1 硬件系统架构设计

硬件控制逻辑是 STM32 作为控制器，使用串口协议向 NB 模块(BC95)发送 AT 指令，BC95 内置了 UDP/CoAP 协议栈，当接收到 AT 指令时，会根据命令与最近的基站通信，进行初始化入网，之后建立 UDP 服务或 CoAP 服务与云平台建立通信。之后模块可主动向云平台发送上行数据，而模块接收的消息也会通过串口传输给 MCU。

4.2　基于 NBIoT 的数据通信服务的设计和实现

4.2.1 主流程模块设计

主流程模块主要包含两部分，第一部分是初始化功能，第二部分是控制 NB 模块与基站通信。下面详细介绍：

初始化功能，首先调用 STM32 库的库函数进行硬件层面初始化，顺序依次为：系统初始化，系统时钟配置，GPIO 口初始化，DMA 初始化，串口功能初

始化，中断控制器初始化；

控制 NB 模块与基站通信：对 NB 模块操作函数进行初始化，配置软件方面的各个参数，之后进入 while 循环，开启定时器，开启 USART1 的串口轮询，NB 主函数进行判断指令的执行情况，事件的触发情况，之后根据 NB 模块的状态标识去执行相应的分支，每个分支对应了一种功能，包括 NB 硬件初始化，NB 模块注册入网，获取 NB 模块的信息，创建 UDP Socket 连接云平台，使用 UDP 发送数据，使用 UDP 接收消息，关闭 UDP Socket，将 NB 模块通过 CoAP 注册到云平台，使用 CoAP 发送数据，使用 CoAP 读取数据，NB 模块重启。在本文中，NB 模块的状态标识可以通过读取另外一个串口 USART2 的数据去设置的，以此达到人机交互的目的，当然也可以通过按键或者定时器定时等去设置这个标识，从而控制 NB 模块的行为。

4.2.2 串口通信设计

STM32 具有多个全双工的串行通信接口,可以作为 UART 使用,或者 USART 一般都使用异步收发模式(UART)。STM32 使用串口通信一般有三种方法：轮询、中断、DMA。

轮询模式是 CPU 不断查询 IO 设备，如设备有请求则加以处理。例如 CPU 不断查询串口是否传输完成,如传输超过则返回超时错误。轮询方式会占用 CPU 处理时间，效率较低。

而中断模式是当 I/O 操作完成时,输入输出设备控制器通过中断请求线向处理器发出中断信号，处理器收到中断信号之后，转到中断处理程序，对数据传送工作进行相应的处理。

DMA 技术则是直接内存存取技术。所谓直接传送，即在内存与 IO 设备间传送一个数据块的过程中，不需要 CPU 的任何中间干涉，只需要 CPU 在过程开始时向设备发出“传送块数据”的命令，然后通过中断来得知过程是否结束和下次操作是否准备就绪。由于 DMA 优越的性能，能够极大减少 CPU 的负担，所以本文采用 DMA 串口通信机制。

关于 DMA 模块的设计主要分为四个部分：DMA 初始化、DMA 写数据、DMA 读数据、DMA 轮询。DMA 初始化是配置 DMA 方式的 USART1 的接收和发送的回调函数。DMA 写数据是将发送缓冲区数据发送 AT 指令。DMA 读数据和 DMA 串口轮询相配合，当接收缓冲区有串口数据时，将之取出来。

接收缓冲区的数据有三种，第一种是 NB 模块收到的来自基站的回应指令，第二种是云服务器通过 UDP 协议发给 NB 终端的数据，第三种是云服务器通过 CoAP 协议发给 NB 终端的数据，也即是下行数据过程。串口轮询是否有数据到

达以及根据收到的数据判断接下来选择进行哪一个流程，对整个通信过程至关重要。

4.2.3 NB 模块驱动设计

1 NB 终端初始化和入网

当 NB 状态标识是 NB_Init 时，进入初始化和入网的流程。首先设置 NB 模块的流程处理状态为 PROCESS_INIT，设置最大超时时间为 2s，调用 NBbc95SendCMD_Usart 函数，将对应的 AT 指令发送给 NB 模块。通过调用 bc95Main NB 主函数判断执行下一条指令，期间不断轮询串口收到的来自基站的回应。顺序执行的指令依次是 AT、AT+CMEE=1、AT+CFUN=1、AT+CIMI、AT+CGSN=1、AT+CEREG=1、AT+CSCON=1、AT+CGATT=1、AT+CGATT?、AT+NSMI=1、AT+NNMI=2，当发送完 AT+CGATT?指令后收到基站的回应是 +CGATT:1 OK 说明 NB 模块入网成功，可进行下一步动作。如图 4-2 所示。具体指令的作用可参考附录 A。

```
/* Start AT SYNC: Send AT every 1s, if receive OK, SYNC st
AT
OK

/* Use AT+CMEE=1 to enable result code and use numeric
AT+CMEE=1
OK

/* Use AT+CFUN=1 to open Full functionality*/
AT+CFUN=1
OK

 /* Use AT+CGSN=1 to query the IMEI of module */
AT+CGSN=1
+CGSN:863703039252817
OK

/* Use AT+CIMI to query the IMSI */
AT+CIMI
460042207106991
OK

/* User "AT+CGATT=1"to activate context profile */
AT+CGATT=1
OK

/* Query the status of the context profile,You may have to
AT+CGATT?
+CGATT:1
OK
```

a） 入网

```
/* Use AT+NSMI=1 to turn on sent message indications */
AT+NSMI=1
OK

/* Use AT+NNMI=2, set new messages indications */
AT+NNMI=2
OK
```

b） 续 a）入网

图 4-2 AT 指令控制 NB 模块入网

2 基于 UDP 数据通信服务的设计

(1) 创建 UDP Socket

当 NB 状态标识是 NB_UDP_Create 时，进入创建 UDP Socket 的流程。首先设置 NB 模块的流程处理状态为 PROCESS_MODULE_INFO，设置最大超时时间为 1s，调用 NBbc95SendCMD_Usart 函数，将对应的 AT 指令发送给 NB 模块。通过调用 bc95Main NB 主函数判断执行下一条指令，期间不断轮询串口收到的来自云服务器的回应。顺序执行的指令是 AT+NSOCR=DGRAM,17,5683,1，当发送完该指令后收到 0-7 的回应时，说明成功建立了序号为回应值的 UDP 协议的 Socket，可进行发送注册包注册到云平台了。

(2) 使用 UDP 协议发送数据

当 NB 状态标识是 NB_UDP_Send 时，进入使用 UDP 协议发送数据流程。首先根据传输进来的参数设置发送缓冲区和需要发送的数据长度，之后格式化发送缓冲区，设置 NB 模块的流程处理状态为 PROCESS_UDP_SEND，最大超时时间为 5s，调用 NBbc95SendCMD_Usart 函数，将对应的 AT 指令发送给 NB 模块。通过调用 bc95Main NB 主函数判断执行下一条指令，期间不断轮询串口收到的来自云服务器的回应。执行指令 AT+NSOST=0,ip,port,msgLen,Hex String，当发送完该指令后收到 Socket ID, msgLen OK 格式的消息时，说明消息已经发送出去了。此过程为 NB 终端到云平台的上行数据流程。如图 4-3 所示。

3 基于 CoAP 数据通信服务的设计

(1) 终端 NB 模块初始化 CoAP 服务

当 NB 状态标识是 NB_Coap_Server 时，进入使用终端 NB 模块初始化 CoAP 服务的流程。首先设置 NB 模块的流程处理状态为 PROCESS_COAP，调用 NBbc95SendCMD_Usart 函数，将对应的 AT 指令发送给 NB 模块。通过调用 bc95Main NB 主函数判断执行下一条指令，期间不断轮询串口收到的来自云服务器的回应。执行的指令是 AT+NCDP=ip,5683，当发送完该指令后终端设备收到 OK 的回应消息时，说明已设置了远程 CDP 服务器的 IP 和端口。如图 4-4 所示。

(2) 终端 NB 模块初始化 CoAP

当 NB 状态标识是 NB_Coap_Send 时，进入使用终端 NB 模块使用 CoAP 协议发送数据的流程。首先根据传输进来的参数设置发送缓冲区和需要发送的数据长度，之后格式化发送缓冲区，设置 NB 模块的流程处理状态为

PROCESS_COAP_SEND，将调用 NBbc95SendCMD_Usart 函数，将对应的 AT 指令发送给 NB 模块。通过调用 bc95Main NB 主函数判断执行下一条指令，期间不断轮询串口收到的来自云服务器的回应。执行的指令是 AT_NMGS=MsgLen,Msg，当发送完该指令后收到 OK，和+NSMI:SENT 的回应消息时，说明已将数据成功发送出去。此过程为 NB 终端到云平台的上行数据流程。如图 4-5 所示。

```
[21:04:06.191]收←◆
OK
->:AT+NSOCR=DGRAM,17,5683,1

[21:04:06.287]收←◆

[21:04:06.319]收←◆
0

[21:04:06.351]收←◆
OK
UDP Create:S
UDP初始化成功
 BC95 Module Send Messsage to UDP Server
->:AT+NSOST=0,193.112.57.70,18777,10,746573740D0A00000D0A

[21:04:06.447]收←◆

[21:04:06.511]收←◆
0,10

OK
```

图 4-3 AT 指令控制 UDP 数据上行

```
/* Use AT+NCDP to Configuration CDP server */
AT+NCDP=193.112.57.70,5683

OK
/* Use AT+CFUN=1 Configuration CFUN become a fully fu
AT+CFUN=1

OK
/* Use AT+NCDP? to query the server IP address and port
AT+NCDP?

+NCDP:193.112.57.70,5683

OK
```

图 4-4 AT 指令初始化 CoAP 服务

```
/* Use AT+NMGS to  send a message from t
AT+NMGS=6,74796975796F

OK

+NSMI:SENT
```

图 4-5 AT 指令控制 CoAP 数据上行

第 5 章　物联网各协议的通信流程和设计

此章节是本套系统的精华之处，也是难点设计之处。

5.1　UDP 通信流程设计

5.1.1 上行数据过程

如图 5-1 所示，首先单片机向终端 NB 模块发 AT 指令，终端模块与基站通信，初始化，入网。此时 NB 终端尝试连接指定的服务器，连接成功后建立一个 UDP socket 与服务端通信。

首先进入云平台数据调试页面，此时云平台已经在服务端建立 UDP 服务，并监听了一个指定的端口，本文采用的端口是 18777。之后单片机按照数据调试页面提示控制 NB 模块发送注册包到云平台服务器，如果收到云平台的响应的状态码为 1，则注册成功，此时设备就已正式进入云平台，可以进行正常的数据收发。此时使用终端 NB 模块发送数据给云平台，是上行数据传输过程；而云平台发送给 NB 终端是下行数据传输过程。

5.1.2 下行数据过程

如图 5-2 所示，云平台可直接向 NB 终端传递数据。NB 终端收到数据后会在串口打印相应的字符串。此时设备终端判断数据发送成功。

5.1.3 上下行过程需要注意的几点

(1) 上下行过程中设备终端一直是 UDP Cllient 端，云平台一直是 UDP Server 端。

(2) 只能先进行上行数据传输过程，注册到平台后，才可以进行下行数据传输过程。

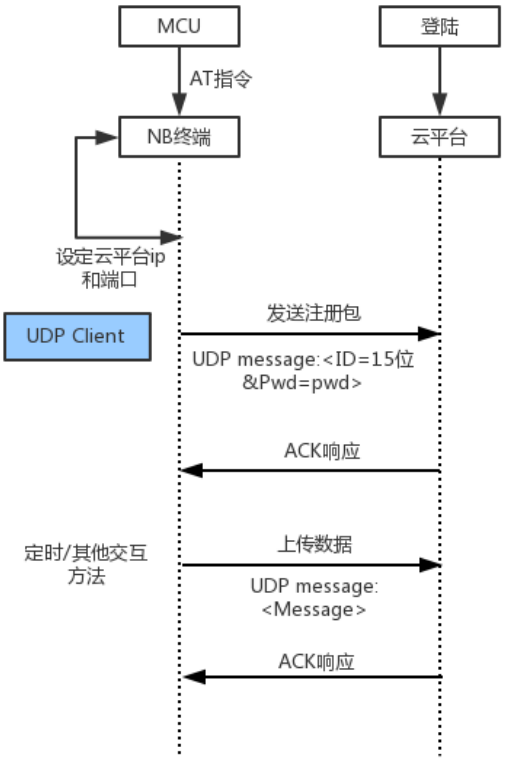


图 5-1 UDP 上行过程

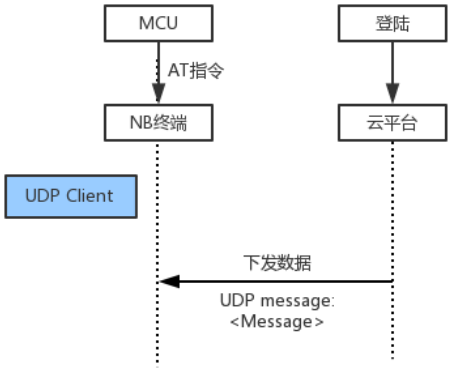


图 5-2 UDP 下行过程

5.2 CoAP 通信流程设计

5.2.1 NB 模块内部 CoAP 资源

在设计 CoAP 通信流程前，需要先了解 NB 模块内部的 CoAP 资源。

CoAP 资源其实是一个逻辑上的概念，也可以称为路由，或者称为 Endpoint。NB 模块中实现了 CoAP Server，也实现了 CoAP Client 用于向 CDP 服务器注册。NB 模块中包含一个 URL 为“t/d”的资源，该资源支持 GET 方法和 POST 方法。NB 模块中内的“t/d”支持观察者模式，可支持订阅。当 NB 模块需要与云平台沟通时则需要用到这些资源，具体参考下文。

5.2.2 CoAP 的通信流程

CoAP 的通信流程分为上行数据通信和下行数据通信。

1 上行流程的流程

如图-3 所示，开始时单片机向终端 NB 模块发 AT 指令，终端模块与基站通信，初始化，注册进入 NB 网络。

之后选择 NB 设备进入云平台数据调试页面，此时云平台已经在服务端建立

CoAP 服务，并监听了 5683 端口。

上行过程包括注册过程、服务器订阅过程、NB 终端向服务器发送指示等过程。

注册过程 NB 终端向服务器 t/r 路由进行注册，query 参数 ep=IMEI 号。服务器收到请求后响应一个状态码为 2.04 的 ACK 包。之后 CDP 服务器发送一个 CON 类型的 CoAP 请求包，该包需要设置三个参数，第一设置请求方法为 GET，请求路由为 t/d，第二设置远程模块 IP 和端口，第三设置 token 为自定义生成的 7 个字节，第四设置观察者模式，此时云平台为观察者，发送成功后终端 NB 模块会发送一个 ACK 包回应，注意该包 token 需和前一个 CON 包一致，状态码为 2.05。此时服务器接收这个回应将订阅 NB 终端 t/d 资源，等待 NB 终端向它发送指示。然后 NB 终端主动向云服务器发送 NON 包，状态码为 2.05，并且 payload 里面携带了上行数据。云服务器收到 CoAP 包后给 NB 终端回应一个空的 ACK 包，通信完成。

2 下行过程

如图 5-4 所示，下行通信过程比较简单，CDP 服务器将使用 POST 方法向 NB 终端传递数据。CDP 服务器作为 CoAP Client，NB 终端作为 CoAP Server。NB 终端相应这个请求后并发送一个 ACK 包给 CDP 服务器。此时云平台判断数据发送成功。

终端设备接收成功后会自动打印出相应的信息，通信完成。

3 上下行过程需要注意的几点

(1) 上行下行数据过程使用了 CoAP 的观察者模式。

(2) 设备在云平台注册成功后的上行通信过程和下行通信过程不需要再进行注册。

(3) 上下行进行前需先使 NB 模块的 SIM 卡入网。

(4) 只能先进行上行数据传输过程，使得云平台订阅了终端模块的资源后，才可以进行下行数据传输过程。

(5)此处 CDP 服务器指的是云平台。

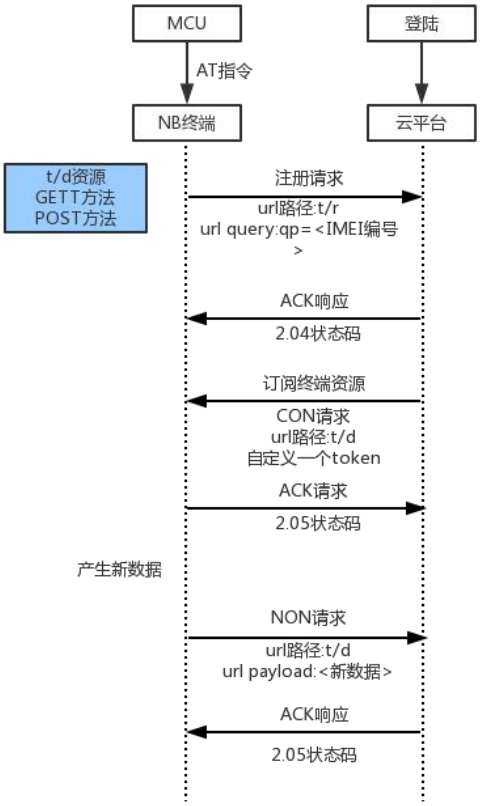


图 5-1 CoAP 上行过程

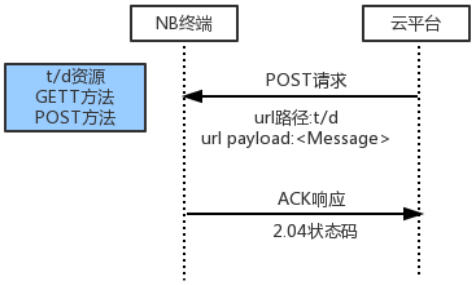


图 5-2 CoAP 下行过程

5.3 整套系统运行效果

5.3.1 云平台初始化

1 首先注册一个账号，注册成功后登陆进设备管理后台，此处效果参考第 3 章第 1 节前端设计。

2 创建一个 CoAP 设备。不管设备采用 UDP 或者 CoAP 协议与云平台通信，前端显示的效果是一样的，所以此处以采用 CoAP 为通信协议的设备为例。如图 5-3 所示。

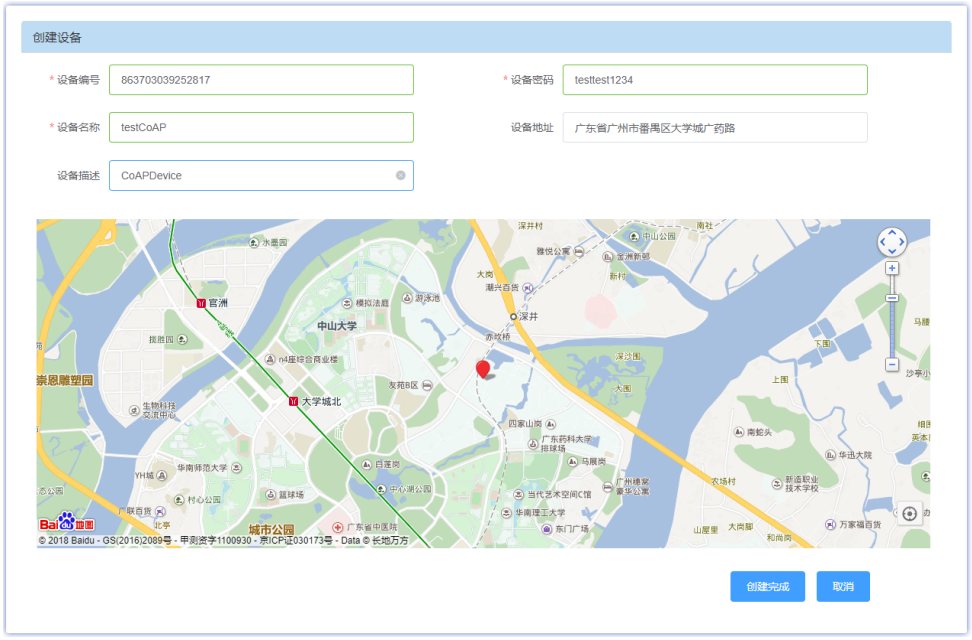


图 5-3 创建设备

3 如果创建成功，则会提示创建设备成功，如图 5-4 所示。

图 5-4 创建设备成功

4 此时设备列表里面已经有该设备信息，如图 5-6 所示。

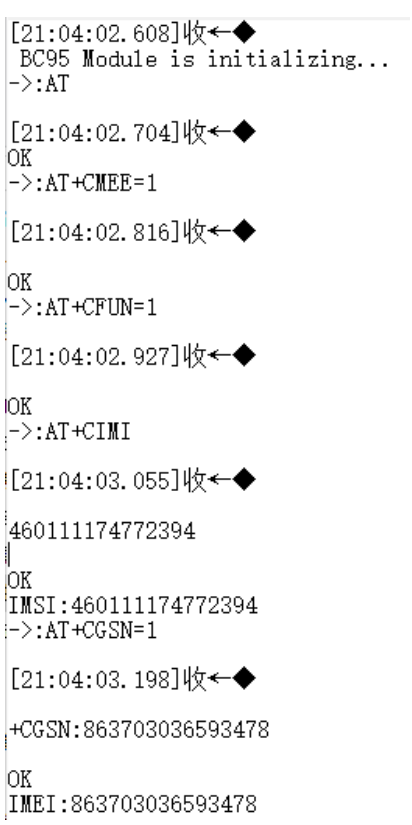
863703039252817	testCoAP	testtest1234	广东省广州市番禺区大学城广药路		数据传输　编辑　删除

图 5-5 设备列表信息

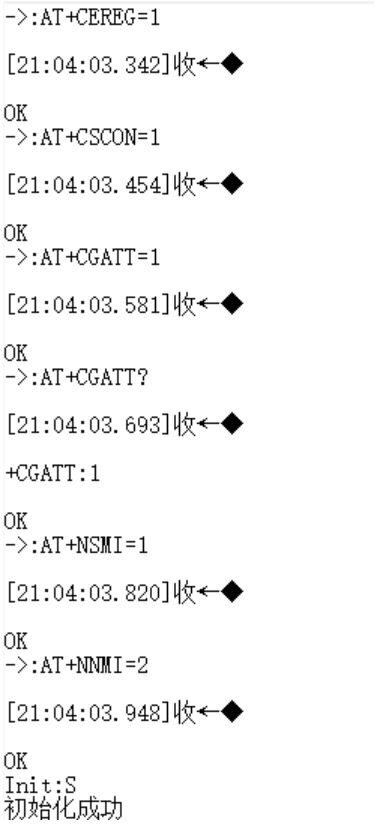
5 进入数据调试页面，此时云平台初始化工作已完成，等待设备终端的连接和数据收发。

5.3.2 设备终端初始化

1 使用串口连接单片机，使用串口助手发送数字 1 给单片机，单片机响应后直接进行初始化 NB 模块的一系列过程，如图 5-6 所示。



a）AT 指令以及收到的回复



b）续 a）

图 5-6 设备终端初始化

2 发送数字 8 给单片机，单片机响应后发送 AT 指令设置终端 NB 模块的 CoAP 服务，具体效果和命令可参照第 4 章第 2 节。

3 发送数字 9 给单片机，单片机使用 AT 指令控制 NB 终端向云平台发送数据。具体效果和命令也可参照第 4 章第 2 节。

5.3.3 云平台上和设备终端显示的数据交互

1 经过上述过程后，等待几秒时间，云平台会收到设备上线通知，并且显示收到的数据，如图 5-7、图 5-8 所示。

图 5-7 设备上线

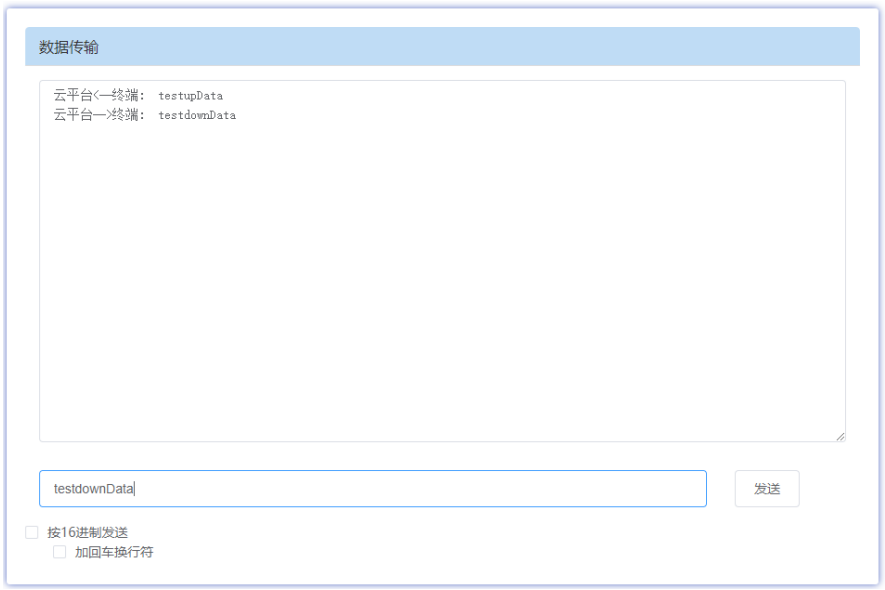


图 5-8 上行数据成功

2 下行数据过程只需要在输入框里面输入需要发送的数据，点击发送，等待几秒时间，设备终端即可收到云平台下行的数据如图 5-9 所示。

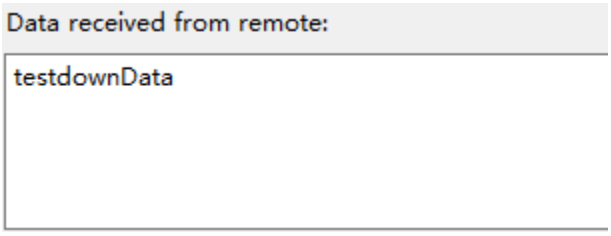


图 5-9 设备下行数据成功

总 结 与 展 望

此项目成功地构建了一个基于 ARM 的窄带物联网系统。该系统包含了云平台和设备终端两部分，通过的单片机的控制，NB 模块注册入网再配置相应的参数，按照本文设计的 UDP 和 CoAP 的通信流程，与云平台建立连接，实现了数据的上行和下行传输。

通过对窄带物联网系统的设计，该项目可作为开发窄带物联网平台的开源参考，也可以直接移植修改开发成小应用。

但是项目中还存在这一些不足之处。第一，云平台暂时只适配了 UDP、CoAP 协议，没有适配 MQTT、TCP 等其他广泛应用的标准化协议。第二，硬件平台只适配了 STM32 平台的。

为了解决此研究中的不足之处，后续会陆续适配 MQTT、TCP 协议，以及适配 ARM9、Arduino、树莓派等具有广泛开发者使用的平台。

在此也把项目开源，希望该项目能够继续发展完善，以及被更多人认可和使用。

参考文献

1. 陈宝亮. 电信业谋利物联网：NB-IoT 终结“碎片化”？. 21 世纪经济报道,2015-11-05(020)
2. 汤春明,张荧,吴宇平.无线物联网中 CoAP 协议的研究与实现.现代电子技术,2013,36(01):40-44
3. 贾雪琴,张云勇. NB-IoT 运营策略[J]. 中兴通讯技术,2017,23(01):21-24.
4. 平凡. 物联网平台纷争，谁将脱颖而出？[N]. 人民邮电,2017-08-31(006).
5. Olof Liberg,Mårten Sundberg,Y.-P. Eric Wang,Johan Bergman,Joachim Sachs. Chapter 1 - The Cellular Internet of Things[M].Elsevier Ltd:2018-11-30.
6. 刁兴玲. 物联网市场鏖战已起 中国联通确立“物联网平台+”生态战略[J]. 通信世界,2017,(26):43.
7. 黄海峰. NB-IoT、eMTC 与 LoRa 相争 谁将胜出?[J]. 通信世界,2017,(14):16.
8. 侯海风. NB-IoT 关键技术及应用前景[J]. 通讯世界,2017,(14):1-2.
9. Rashmi Sharan Sinha,Yiqiao Wei,Seung-Hoon Hwang. A survey on LPWA technology: LoRa and NB-IoT[J]. ICT Express,2017,3(1):.
10. 张康. 基于 CoAP 协议的数据采集系统设计与实现[D].内蒙古科技大学,2015.
11. 欧阳恩山. 基于开放标准的 NB-IoT 平台体系设计[J]. 福建冶金,2017,46(06):59-62+25.
12. 黄忠,葛连升. 基于 CoAP 的物联网 Web 服务统一访问方法[J]. 山东大学学报(工学版),2014,44(04):16-21+30.
13. 葛丹. 物联网传感器数据处理平台的设计与实现[D].南京邮电大学,2016.
14. 陈淑珍. NB-IoT 标准化进展及关键技术研究[A]. 广东省通信学会、中国电信股份有限公司广东研究院、《移动通信》杂志社.2016 广东蜂窝物联网发展论坛专刊[C].广东省通信学会、中国电信股份有限

公司广东研究院、《移动通信》杂志社:,2016:5

15. 钱小聪,穆明鑫.NB-IoT 的标准化、技术特点和产业发展[J].信息化研究,2016,42(05):23-26.
16. 武磊,张正炳,胡蓉华. 基于 Web 的家居设备远程控制系统设计与实现[J]. 微型机与应用,2017,36(19):66-69.
17. Tapio Levä,Oleksiy Mazhelis,Henna Suomi. Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications[J]. Decision Support Systems,2014,63:.
18. Hao WANG. Research of the Wireless Communication Technologies in Distribution Automation[A]. Science And Engineering Research Center.Proceedings of 2016 International Conference on Wireless Communication and Network Engineering (WCNE2016)[C].Science And Engineering Research Center:,2016:4.s
19. 朱二华.基于 Vue.js 的 Web 前端应用研究[J].科技与创新,2017(20):119-121.
20. 吴子然,李多,杨争辉,叶桦. 基于Node.js的家庭智能地暖远程监控系统[J]. 计算机科学与应用,2015,05(06).
21. Paul Krill. Node could bring JavaScript to the Internet of things[J]. InfoWorld.com,2016.

致　谢

在本次论文设计撰写过程中，高老师从选题指导、论文框架到细节修改，都给予了细致的指导，提出了很多宝贵的意见与建议。老师严谨的思维，化繁为简的实践方法给与了我很大启发。这篇论文是在老师的精心指导和大力支持下才完成的，感谢老师的传道授业解惑。

感谢所有授我以业的老师，没有这些年知识的积淀，我没有这么大的动力和信心完成这篇论文。

感谢在论文撰写过程中给予我帮助和支持的同学朋友们，正是你们热情的建议与热心的帮助，才让本文不断地完善。

最后，我要向百忙之中抽时间对本文进行审阅的各位老师表示衷心的感谢。

附　　录

附录 A：

部分 AT 指令说明

指令	作用
AT	开启 AT 同步。
AT+CMEE=1	上报设备错误，启用结果码。
AT+CGMI	查询设备制造商信息
AT+CGMM	查询设备 ID
AT+CGMR	查询设备固件版本
AT+NBAND?	查询模块是否支持 band
AT+NCONFIG?	查询用户配置
AT+CGSN=1	查询 IMEI
AT+CIMI	查询 IMSI
AT+CGATT=1	附着 NB 网络
AT+CGATT=0	分离 NB 网络
AT+CGATT?	查询附着状态，需等待几秒
AT+CSQ	查询信号强度
AT+COPS?	查询注册模式和运营商
AT+CEREG?	查询注册状态,返回 1 表示已注册本地网络
AT+NUESTATS	获取设备最新的操作统计
AT+NSORF	读取 UDP 发送回来的数据，返回实际返回的数据长度
AT+CFUN	功能限制。等于 1 时开启所有传输和接收的射频功能；等于 0 则关闭所有射频功能。
AT+CSCON	查询 sim 卡的基站连接状态。返回 1 表示已连接，0 表示空闲。
AT+NSMI	当上行数据发送到 CDP 服务器时设置或获取通知
AT+NNMI	当终端收到来自 CDP 服务器的下行数据时设置或接收通知
AT+NSOCL	关闭 socket(本文用来关闭 UDP 的 socket)

AT+NSOCR	创建 socket，第二个参数是设置协议，UDP 对应的是 17
AT+NSOST	发送一个固定长度的 UDP 数据报到服务器
AT+NCDP	设置 CDP 服务器的 ip 和端口并查询
AT+NMGS	通过 CDP 服务器从终端向网络发送消息
AT+NMGR	从 CDP 服务器接收消息

附录 B：

项目的 GitHub 地址是 https://github.com/SoftHardDevolop/NBIot-Cloud-Server