互联大师——
工业互联网智能网关
项目详细方案

互联大师-工业互联网智能网关
设计文档

所在赛道与赛项： A

目录

一、目标问题与意义价值

1.1 项目背景

工业互联网是新生的通信与工业经济融合的生态技术,对新一轮科技革命和产业变革机遇有着重大意义。工业互联网的跨界融合特征带来了一系列技术创新,还支撑着大规模个性定制、开放式协同制造、服务型制造等新模式、新业态得以深度应用和全面普及。

1.2 目标问题

现今工控设备设备多种多样,常用的工控设备通信口有 RS-232、RS-485、CAN 和以太网,常用的工控协议有 Modbus、Fins、S7、PPI 等。常出现异构设备通信的总线接口和协议不匹配的问题,这极大程度地限制异构设备的联动交互,大幅增加了设备更新成本。同时,当前工控协议种类繁多,缺乏统一的标准。

国家高度重视工业设备上云,明确提出要“实施工业设备上云‘领跑者’计划”。通过推动工业设备上云,并开展运行监测、能效优化等服务,带来直接经济效益数额巨大。工控协议作为云平台与设备交互的语言,提出一种普适性强,开源通用的工控协议,将设备接入云平台,应用前景广泛。

1.3 项目介绍

基于多协议交互与工业设备上云两个痛点问题。制定统一标准、开源通用的工控协议,开发支持多协议自适应交互同时连接云平台的工控网关应用前景广泛。本项目抽象主流工控协议,提炼工控协议模板,改进并提出 ICCP 协议作为工控协议标准。以 ICCP 协议为核心,实现多协议自适应交互模型,开发智能网关与配套系统。

1.4 项目价值

对用户企业而言,协议转换网关能降低设备更新成本,降低设备采购耦合度。设备联云,可实现工业设备运行状态、运行环境等数据的云端汇聚,形成以数据驱动为特征的业务决策闭环,加速业务流程重组和生产方式优化,助力企业提质降本增效。

对平台运营商而言,推动工业设备上云,有利于进一步完善设备数据采集体系,突破协议解析、边缘智能等技术短板,巩固工业互联网平台技术支撑体系。

二、设计思路与方案

2.1 协议提出

工控协议种类众多,欲制定工控协议统一标准,需对多种工控协议有深入了解。通过前期调研,学习 ModBus、Fins、PPI、PowerLink 在内的多种协议,了解协议的通信模式与报文格式。基于对工控协议的抽象和适当改进,提出了工控协议 ICCP(Industrial Control Common Protocol)。具体协议图如图 1 所示。

图 1 ICCP 协议

2.2 协议转换

工控协议众多,N 种协议一对一转换需要 N*(N-1)/2 次,工作量巨大且不易扩展。以 ICCP 协议为中间协议实现南北向协议转换。能大幅减少工作量，减少协议转换冗余次数，同时后续方便扩展，网关引入新协议只需要实现协议与 ICCP 协议的转换，而非引入新协议与既有协议一对一转换。具体如图 2 所示。

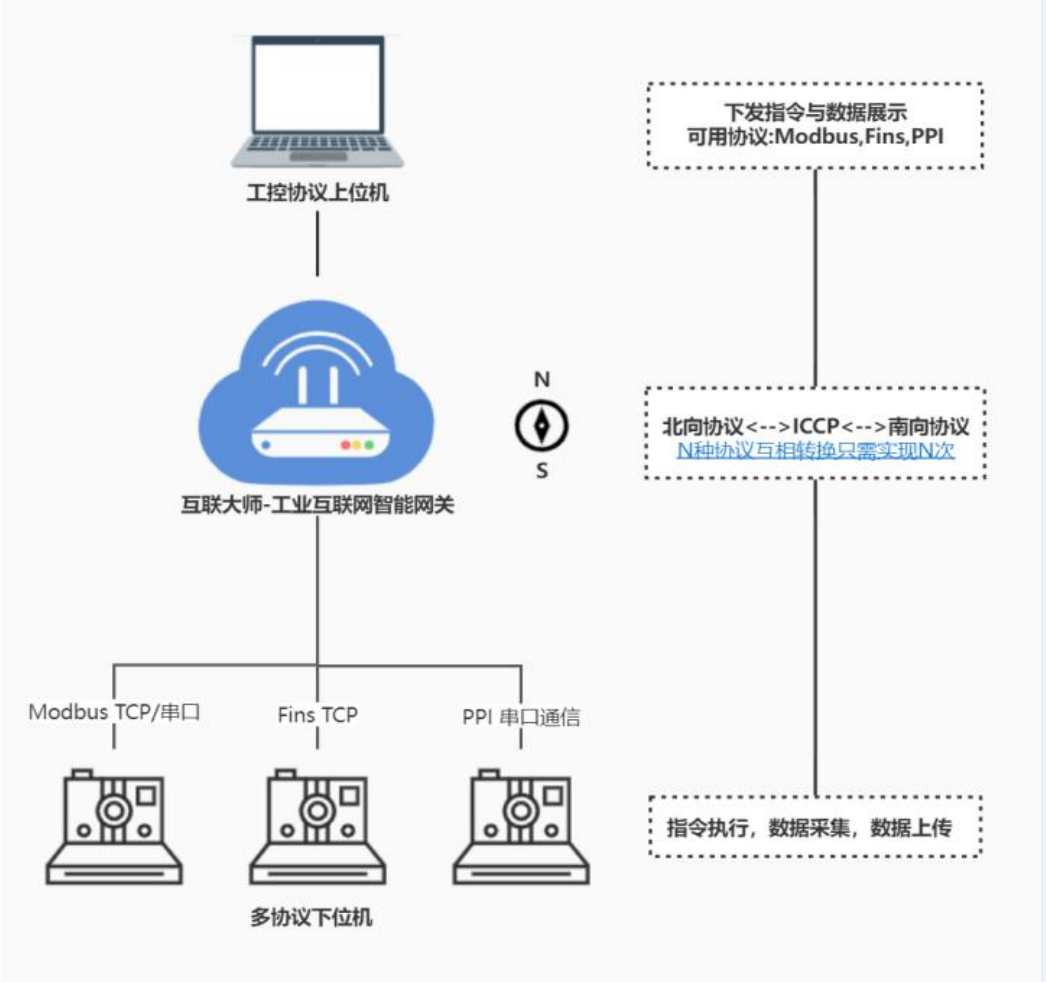

图 2 协议转换图

2.3 设备上云

网关北向通过 ICCP 协议接入互联网云平台，南向连接工业设备，借助网关将南向协议报文统一转换为 ICCP 协议报文，上位机统一协议为 ICCP，基于 ICCP 协议开发云端上位机。借助网关实现串行线路与互联网的交流,实现云端远程操控和监视下位机。具体如图 3 所示。

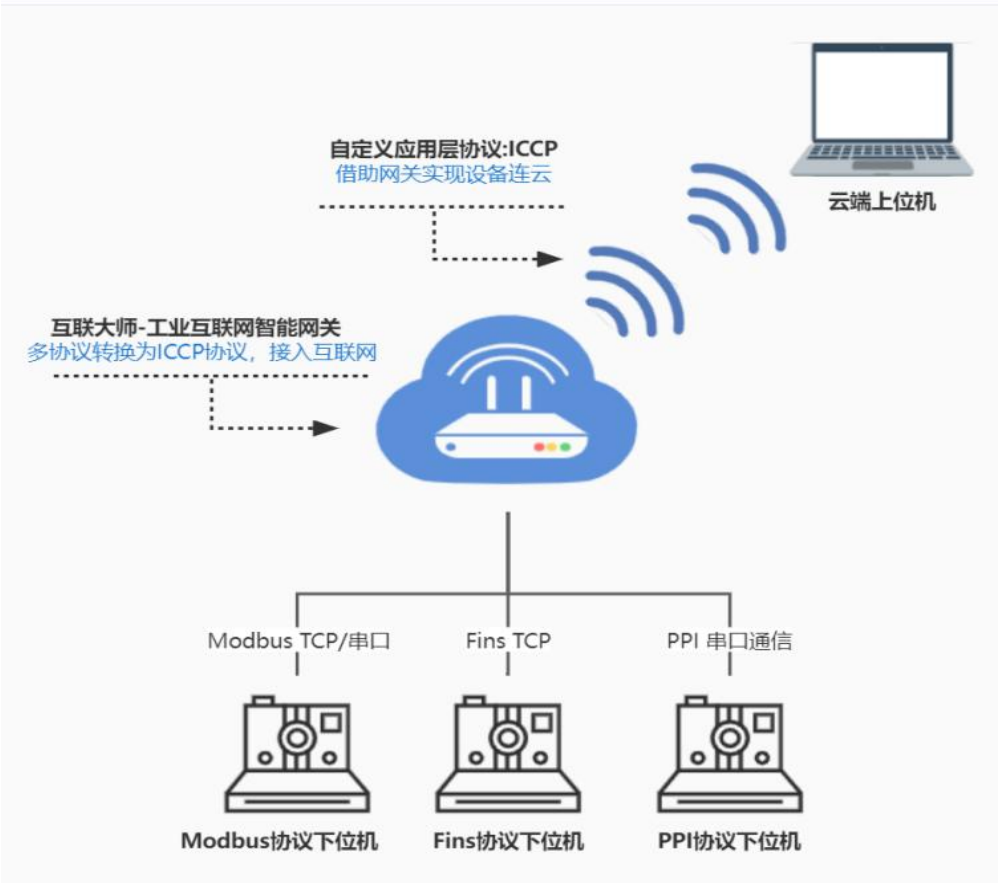

图 3 设备上云图

2.4 网关可视化

除了基础网关系统研发，开发网关系统配套的 Web 软件，实现网关系统的基础配置，设备注册，日志展示，安全过滤等功能。具体如图 4 所示。

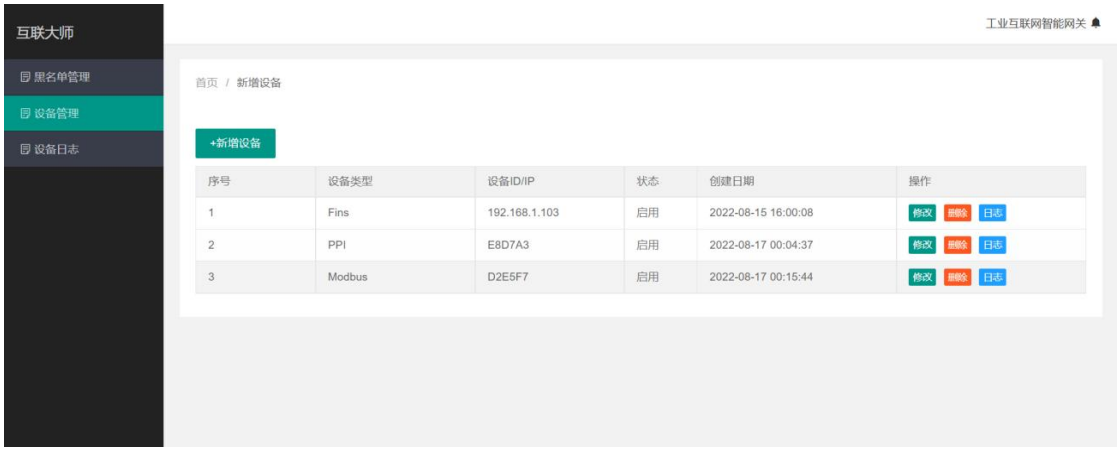

图 4 网关界面

2.5 场景还原

考虑到网关的实用性与效率，采用树莓派与温湿度传感器、LED 灯等设备，分别开发支持 ModBus，Fins，PPI 协议的下位机。同时开发支持多种协议的云端上位机。通过 WiFi、交换机、网线、串行线连接云端上位机、基于树莓派的网关、基于树莓派的下位机。还原真实工控场景，验证网关可用性。具体如图 5 所示。

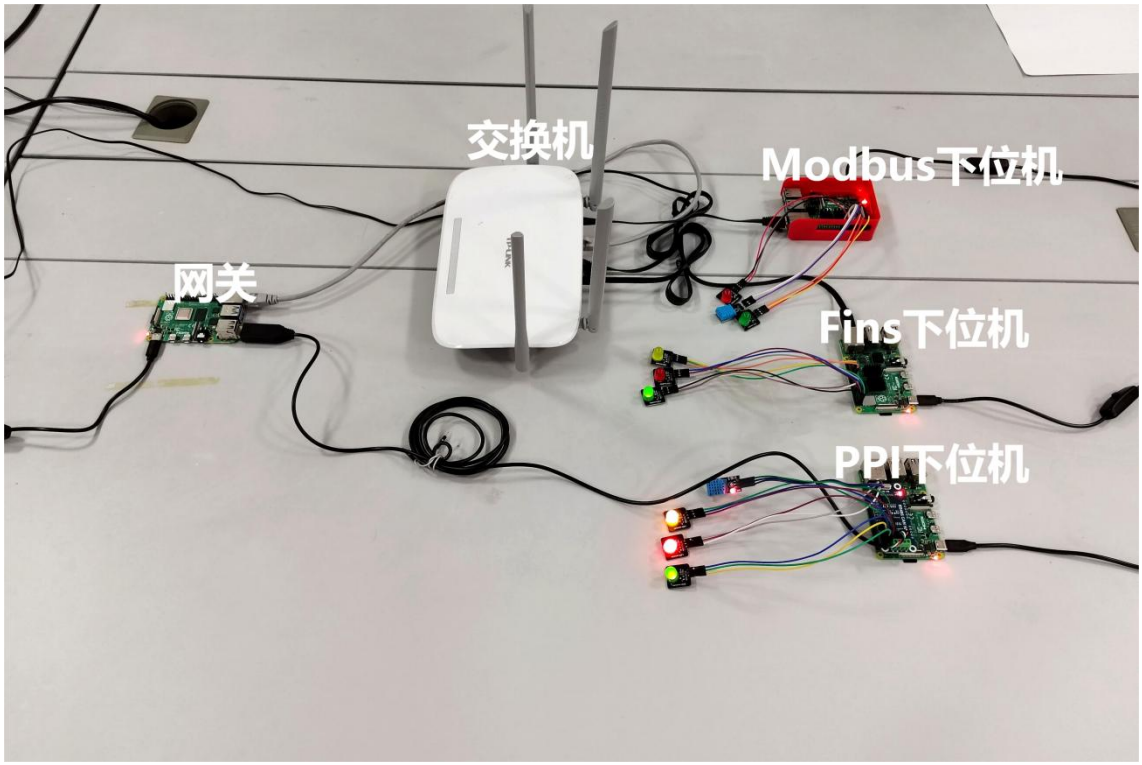

图 5 现实工业场景模拟

三、方案实现

3.1 协议制定

3.1.1 场景调研

（1）现实场景

工业控制系统,是由各种自动化控制组件以及对实时数据进行采集、监测的过程控制组件共同构成的确保工业基础设施自动化运行、过程控制与监控的业务流程管控系统。其核心组件包括数据采集与监控系统(Supervisory Control and Data Acquisition,SCADA)、分布式控制系统(Distributed Control Systems,DCS)、可编程控制器(Programmable Logic Controller,PLC)、远程终端(Remote Terminal Unit,RTU)、人机交互界面设备(Human Machine Interface,HMI),以及确保各组件通信的接口技术。

一个 SCADA 控制中心是长期对现场进行集中监控的通信网络,包括监控报警和处理状态数据。可以将自动化或操作者驱动的监控命令推送到远程站控制设备，远程站设备通常被称为现场设备。现场设备控制本地诸如打开和关闭阀门和断路器的操作，从传感器系统收集数据，以及监控当地环境的报警条件等。

DCS 集成为一个控制架构，包含监督多个集成子系统的监督级别的控制它们负责控制本地化流程的细节。DCS 用于控制工业过程，如发电，油气炼油，水和废水处理，以及化学，食品和汽车生产。

PLC 是基于计算机的固态设备，用于控制工业设备和工艺。而 PLC 是整个 SCADA 和 DCS 系统中使用的控制系统组件,它们通常是较小的控制系统配置中的主要组件用于提供监管控制汽车装配线和发电厂吹灰器控制等离散工艺。 PLC 是广泛应用于几乎所有的工业过程。

HMI 也就是操作人员面前的显示屏，HMI 和 plc 进行组态之后，可以通过 HMI 对 PLC 通信控制以达到控制终端设备的目的。

（2）常用协议

在众多公开或私有工控协议中可分为如下几类：

- 标准协议：国际标准或公认的标准协议，如 ModBus、DNP3、IEC104 等。
- 私有公开：只有厂商自己设备支持并提供官方协议文档，如 Omron Fins 协议、三菱 Melsec 协议。
- 私有不公开：只有厂商自己设备支持且官方不提供协议文档，如 S7、西门子 PPI 协议、GE SRTP。

3.1.2 协议分析

（1）ModBus-TCP

Modbus 协议是应用层（协议层）报文传输协议，它定义了一个控制器能认识使用的消息结构，而不管它们是经过何种网络进行通信的。它描述了一控制器请求访问其它设备的过程，如何回应来自其它设备的请求，以及怎样侦测错误并记录。它制定了消息域格局和内容的公共格式。

Modbus 协议采用主从模式（或 C/S 模式），协议决定了每个从设备（也称下位机、控制器、slave、server）须要知道它们的设备地址，识别主设备（也称上位机、client，Master）按地址发来的消息，决定要产生何种行动。如果需要回应，从设备将生成反馈信息并用 Modbus 协议发出。Modbus 协议分为三种，基于以太网的 Modbus-TCP，基于串行传输的 Modbus-RTU 和 Modbus-ASCII。

通信过程：connect 建立 TCP 连接；准备 Modbus 报文；使用 write 命令发送报文；在同一连接下等待应答；使用 read 命令读取报文，完成一次数据交换；通信任务结束时，关闭 TCP 连接。

Modbus 协议的报文（或帧）的基本格式是：表头 + 功能码 + 数据区 + 校验码。功能码和数据区在不同类型的网络都是固定不变的,表头和校验码则因网络底层的实现方式不同而有所区别。表头包含了从站的地址，功能码告诉从站要执行何种功能，数据区是具体的信息。

ModbusTCP 的数据帧可分为 MBAP（报文头）和 PDU 两部分：MBAP+PDU = MBAP + 功能码 + 数据域，MBAP 7byte，功能码 1byte，数据域不确定，由具体功能决定。

MBAP 为报文头，长度为 7 字节，组成如下表所示。

序列号	协议标识	数据长度	单元标识符
2byte	2byte	2byte	1byte

其中：

内容	解释
序列号	报文的序列号，一般每次通信之后就要加 1 以区别不同的通信数据报文
协议标识	Modbus-TCP 协议为 00 00
数据长度	表示接下来的数据长度，单位为字节（包括单元标识符）
单元标识符	设备 ID

Modbus 中，数据类型可以分为两大类，分别为位变量(Coil)和整形变量(Register)，每一种数据，根据读写方式的不同，又可细分为两种（只读，读写）。

Discretes Input（离散输入寄存器）	位变量	只读
Coils（线圈）	位变量	读写
Input Registers（输入寄存器）	16-bit 整型	只读
Holding Registers（保持寄存器）	16-bit 整型	读写

Modbus 协议常用功能码：

功能码	名称	功能
0x01	读线圈状态	读位（读 N 个 bit）---读从机线圈寄存器，位操作
0x02	读输入离散量	读位（读 N 个 bit）---读离散输入寄存器，位操作
0x03	读多个寄存器	读整型、字符型、状态字、浮点型（读 N 个 words）---读保持寄存器，字节操作
0x04	读输入寄存器	读整型、状态字、浮点型（读 N 个 words）---读输入寄存器，字节操作
0x05	写单个线圈	写位（写一个 bit）---写线圈寄存器，位操作
0x0F	写多个线圈	写位（写 n 个 bit）---强置一串连续逻辑线圈的通断
0x06	写单个保持寄存器	写整型、字符型、状态字、浮点型（写一个 word）---写保持寄存器，字节操作
0x10	写多个保持寄存器	写整形、字符型、状态字、浮点型（写 n 个 word）---把具体的二进制值装入一串连续的保持寄存器

（2）Fins

Fins（factoryinterface network service）通信协议是一种应用层协议，是欧姆龙公司开发的用于工业自动化控制网络的指令/响应系统。使用 FINS 指令可实现各种网络间的无缝通信,包括用于信息网络的 Ethernet(以太网),用于控制网络的 ControllerLink 和 SYSMAC LINK。通过编程发送 Fins 指令，上位机或 PLC 就能够读写另一个 PLC 数据区的内容，甚至控制其运行状态，从而简化了用户程序。Fins 协议支持工业以太网，这就为 OMRON PLC 与上位机以太网通信的实现提供了可能。Omron Fins 协议缺省 TCP/UDP 端口号为 9600。

Fins 协议采用主从模式（或 C/S 模式），协议决定了每个从设备（也称下位机、控制器、server）须要知道它们的设备 IP 地址，识别主设备（也称上位机、client，Master）按 IP 地址发来的消息，决定要产生何种行动。如果需要回应，从设备将生成反馈信息并用 Fins 协议发出。Fins 协议分为两种，运输层使用 TCP 协议的 Fins TCP 和运输层使用 UDP 协议的 Fins UDP.

通信过程（以Fins TCP为例）：构造Fins建立连接指令报文，发送建立连接指令；收到下位机的连接回复后，通过所需的功能构造相应的指令格式并发送至下位机，收到下位机回复获取对应的数据，通信完成。

Fins协议的报文（或帧）的基本格式是：协议标识 + 指令长度 + 命令类型 + 错误码 + 网络标识 + 服务器节点地址 + 客户端节点地址 +FF+ 数据区。协议标识是固定不变的，指明该协议为Fins。指令长度指示从命令类型到数据区的总长度。命令类型区分不同功能的指令。网络标识在不同的网络中会有差异。服务器节点地址和客户端节点地址取决于上位机和下位机的IP地址。数据区是具体相关操作的信息。

Fins TCP协议各字段详细说明如下：

字段	说明
协议标识	4byte，固定46 49 4E 53
指令长度	4byte，说明指令从命令类型到最后的长度
命令类型	4byte，如果是建立连接指令则是00 00 00 00. 建立连接的回复指令是00 00 00 01. 如果是读写指令则是00 00 00 02.
错误码	4byte，如果指令无错误则是00 00 00 00. 如果协议标识不是46 49 4E 53则错误码是00 00 00 01，如果指令长度过长则错误码是00 00 00 02……
网络标识	3byte，第一个字节指明是上位机发送的指令还是下位机回复的指令，80表明上位机发送的指令，C0表明下位机回复的指令。后两个字节常为固定的00 02.
服务器节点地址	3byte,第一个和第三个字节为固定的00. 第二个字节指示服务器的节点。假设服务器IP地址为192.168.1.34，该IP地址的16进制为22，则第二个字节的值为22
客户端节点地址	3byte，同服务器节点地址，第一个和第三个字节为固定的00. 第二个字节指示服务器的节点，计算方法同服务器节点地址。
数据区	长度不固定，取决于具体的功能和数据个数等参数。

Fins协议常用功能码：

功能码	功能
0x0101	内存区域读取
0x0102	内存区域写入
0x0103	内存区域填充
0x0104	多内存区域读取
0x0105	内存区域传输

（3）PPI

PPI通讯协议是西门子公司专为S7-200系列PLC开发的通讯协议。该协议内置于S7-200CPU中。PPI协议物理上基于RS-485口，通过屏蔽双绞线就可以实现PPI通讯。

PPI 协议是一种主-从协议。主站设备发送要求到从站设备，从站设备响应，从站不能主动发出信息。主站靠 PPI 协议管理的共享连接来与从站通讯。PPI 协议并不限制与任意一个从站的通讯的主站的数量，但在一个网络中，主站不能超过 32 个。PPI 协议最基本的用途是让西门子 STEP7-Micro/WIN 编程软件上传和下载程序和西门子人机界面与 PC 通信。

PPI 上位机和 PPI 下位机都是用特定的 ID 进行标示，上位机要找到特定的下位机必须先知道其 ID.

通信过程：构造 PPI 建立连接指令报文，发送建立连接指令；收到下位机的连接回复后，通过所需的功能构造相应的指令格式并发送至下位机，收到下位机回复获取对应的数据，通信完成。

PPI 协议的报文（或帧）的基本格式是：起始符 + 长度 + 起始符 + 目的地址 + 源地址 + 功能码 + 固定字段 + 数据类型 + 数据个数 + 数据区。长度字段表明指令的长度，功能码字段表明指令的功能，读或者写等，数据类型字段表明使用什么单位操作数据。

PPI 各字段详细说明如下：

字段	说明
起始符	1byte，固定的 0x68
长度	1byte，指示从目的地址到指令结束的总长度，会重复一次
目的地址	1byte，指示服务器端的 ID
源地址	1byte，指示客户端的 ID
功能码	1byte，指示指令的功能。常见的有读功能对应 0x6C，写功能对应 0x7C
固定字段	15byte，一般较为固定，和操作方式，冗余识别，参数个数等相关
数据类型	2byte，第二个字节为固定的 0x00。第一个字节指示对数据操作的单位，位操作对应 0x01，字节操作对应 0x02，字操作对应 0x04，双字操作对应 0x06
数据个数	2byte，第二个字节为固定的 0x00。第一个字节指示数据的个数
数据区	长度不固定，根据功能和数据个数等有所变化

3.1.3 协议提出

通过实际应用场景调研与学习多种工控协议，现提出工控协议 ICCP（Industrial Control Common Protocol），ICCP 协议为应用层协议，与底层传输网络无关。考虑到协议的长远发展，参考 RFC3232 号文档后，选定 TCP4983 端口作为 ICCP 协议特定的 TCP 端口。同时本项目基于此端口进行通信。

与众多工控协议相似，ICCP 协议采用主从模式，主设备向指定编号的从设备发送指令，每个从设备须知道它们的设备编号，识别决定要产生何种行动。如果需要回应，从设备将生成反馈信息并用 ICCP 协议送回。

以下为 ICCP 协议结构：

字段	说明
协议标识	2byte，固定为 0xdd 0xdd
序列号	2byte，报文序列号，标识业务编号

源 ID	1byte，上位机在网络拓扑中的编号，通常为 0
目的 ID	1byte，下位机在网络拓扑中的编号，不超过为 255，255 为广播
状态码	1byte，标识指令执行状态与报文传输过程中出现的异常
功能码	1byte，当前仅支持读与写功能，待后续扩展
数据长度	2byte，标识数据区与检验码长度之和，以 byte 为单位
数据区	长度待定，具体的数据，包括操作单位、数量、寄存器类型、偏移量、数据等
校验码	2byte，采用网际校验算法

其中，数据区结构：操作单位+操作数量+寄存器类型+偏移量+数据：

字段	说明
操作单位	1byte，按照单位大小依次分为位、字节、字、双字
操作个数	1byte，一次操作的寄存器数量，不超过 255
寄存器类型	1byte，操作的寄存器的类型
起始地址	3byte，操作寄存器的起始地址，也称为偏移量
数据区	长度不定，当读取数据时则为读到的具体数据，当写入数据时则为上位机写入的数据，无数据可空缺

状态码:协议提供状态码标识下位机指令执行状态以及报文传输过程中出现的常见问题：

状态码	说明
0x00	未出错
0x01	未找到目的设备
0x02	未找到目的寄存器
0x03	寄存器偏移量溢出
0x04	网关异常
0x05	访问权限异常
0x06	校验未通过，报文损坏
0x07	其他异常
...	预留，待扩展

单位：在以 ICCP 协议操作下位机的过程中，将操作单位分为如下若干种：

	0x01	0x02	0x03	0x04
说明	bit（位）	1byte（字节）	2byte（字）	4byte（双字）

寄存器类型：对寄存器按照单位与访问权限分类，可分为以下 4 类：

数据	说明
0x01	开关寄存器（单位为 bit，可读可写）
0x02	状态寄存器（单位为 bit，仅读）
0x03	数据寄存器（单位为 word，仅读）
0x04	控制寄存器（单位为 word，可读可写）

校验码：ICCP 协议校验采用网际校验算法：

在发送方，先把被校验的数据划分为许多 16 位字的序列。如果数据的字节长度为奇数，则在数据尾部补一个字节的 0 以凑成偶数。用反码算数运算把所有 16 位字相加后，然后再对和取反码，便得到校验和。

在接收方，将收到的数据报（包括校验和字段），将所有 16 位字再使用反码算数运算相加一次，将得到的和取反，即得出校验和的计算结果。如果数据报在传输过程中没有任何变化，则此结果必为 0，于是就保留这个数据报。否则即认为出差错，并将此数据报丢弃。

3.2 网关研发

3.2.1 协议转换映射关系

本项目中，协议相关关系对比如下：

	ModBus-TCP	Fins-TCP	PPI	ICCP
编址信息	使用 ID 进行上位机和下位机的识别	使用 IP 进行上位机和下位机的识别	使用 ID 进行上位机和下位机的识别	使用 ID 进行上位机和下位机的识别
协议标识	指令第 3 和 4 字节固定为 0x00，0x00 以及特定 502TCP 端口	指令开始部分的协议标识字段四字节 46 49 4E 53 以及特定 9600TCP 端口	指令的起始符、结束符等	指令开始部分的协议标识字段，特定端口 4983
传输介质	以太网通信	以太网通信	串口通信，基于 RS-485 口	以太网通信
检验方式	无校验	无校验	指令结束的校验码字段	指令结束的校验码字段
通信流程	直接通信，无建立连接过程	需建立连接才能通信	需建立连接才能通信	直接通信，无建立连接过程

寄存器类型方面，Modbus 下位机有四种寄存器类型：离散输入寄存器、线圈、输入寄存器、保持寄存器；Fins 下位机主要有三种存储类型：CIO 区、W 区和 DM 区，均可以读写

位和字；PPI 下位机主要有八种寄存器类型：I、Q、AI、AQ、V、M、S、SM. 其中 AI 和 AQ 只能对字进行操作，其他寄存器均可以操作位和字。

基于上述下位机寄存器特点进行寄存器标准的考虑，将 ICCP 协议寄存器分为按照操作单位与访问权限分为四种类型：开关寄存器与状态寄存器操作单位为 bit，数据控制器与控制寄存器操作单位为字，其中开关寄存器与控制寄存器可读可写，状态寄存器与数据寄存器则只可读。换言之，ICCP 协议下位机寄存器类型与 Modbus 协议下位机的寄存器类型匹配。同时，为了实现协议的转换，例如 PPI 协议转换为 ICCP 协议后再转换为 Fins 协议以实现 PPI 上位机和 Fins 下位机的通信，需要实现不同协议下位机的寄存器的映射和匹配。

通过分析 Modbus 下位机寄存器、Fins 下位机寄存器和 PPI 下位机寄存器后，我们归纳出的不同协议下位机寄存器映射关系如下所示：

ICCP 寄存器	Modbus 寄存器	Fins 寄存器	PPI 寄存器
状态寄存器	Input Register	无	无
开关寄存器	Coil	CIO、W、DM 位操作	I、Q、V、M、S、SM
数据寄存器	Input Register	无	AI
控制寄存器	Holding Register	CIO、W、DM 字操作	I、Q、V、M、S、SM、AQ

对于功能码字段，现在工业上最常用的功能为读和写。读功能，工业上可以通过传感器等获取温度、湿度等信息并存储在下位机的寄存器中，上位机发送读指令从而获取这些数据。对于写功能，工业上可以通过写 0 和 1 控制灯或二极管等设备的开关，上位机发送写指令，下位机读取后并控制相应的设备以达成目的。Modbus 协议、Fins 协议和 PPI 协议均具有读和写功能，因此本项目提取出这些协议的共同点，构造的标准协议设置功能码字段指示命令的功能。

3.2.2 架构

本项目架构图如图 6 所示。

图 6 项目架构图

3.2.3 转换流程

不同协议的通信流程不完全相同，以 ModBus，Fins，PPI 协议为例，Fins 与 PPI 协议在指令报文之前还存在建立连接的过程，而 ModBus 协议则不具有建立连接的过程，因此网关需要根据各种协议的特性，执行代理建立连接的动作。其中，转换流程图如下图 7 所示。

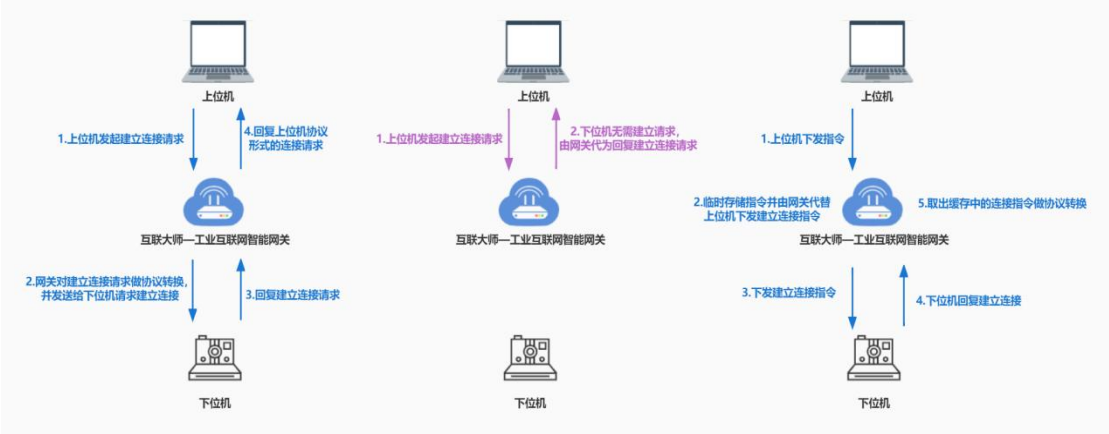

图 7 协议转换流程图

3.2.4 网关开发关键技术

（1）通信

ModBus、Fins、PPI 协议的交互需要实现以太网通信与串口通信，北向借助 ICCP 与网关可实现以太网与串口异构网络之间的交流。

网关与 Fins、Modbus 下位机的通信中，采用 Socket 编程，下位机为 Server，网关为 Client，下位机持续接受 TCP 连接用于读取指令与回复指令，当网关成功实现协议转换后则需要与下位机进行通信，对于 Modbus，Fins 下位机，网关调用 Socket API 与下位机建立连接并交互。对于使用串口通信的 PPI 协议下位机。网关使用 Java 开源串口通信库 jSerialComm 进行串口通信。

（2）多线程

网关作为上位机与下位机交互的中间站，需要维护上下位机的通信对，最佳办法为单独启动线程，线程内部维护上下位机通信的 Socket，上位机与特定下位机在某一时刻的所有指令由一个线程单独处理，线程内部可进行协议转换，与上位机通信，与下位机通信，日志记录等功能。

独立线程解决一对通信对的交互，可避免单点阻塞而引起网关全面阻塞，一个线程阻塞，其他上下位机的通信不受影响，通过配置网关信息，可限制网关最大线程数，当新的线程启动，网关可用线程数量减少，线程结束网关可用线程数增加。通过限制线程数的方式可避免网关资源紧张。

（3）面向对象

采用 Java 中一切皆对象的思想，每一个协议的每一个指令报文可对应到唯一一个对象，对协议报文进行对象封装，将协议封装为协议类，各个协议类均提供构造方法与生成方法，借助构造方法，只需传入 byte 数组指令，协议类自动解析并封装为对象。借助序列化方法，对每一个协议对象，可生成唯一的 byte 数组用于在网络中传输。

将协议报文封装为对象，可避免对网络中的报文直接操作，协议转换简化为对象属性赋值，不同协议的转换，只需按照特定映射规则，即可实现对象之间的转换，且以协议对象为单元，方便数据在网关不同部分之间参数传递。

（4）回调函数

为最大限度的进行协议转换解耦，方便网关后续引进协议进行协议转换。利用 Java 继

承与多态的特性，制定协议转换接口，包括协议标准化接口 ProtocolsToICCP 与标准协议目标化接口 ICCPToProtocols。

ModbusToICCP、FinsToICCP、PPIToICCP 三个类均实现 ProtocolsToICCP 接口，按照各自的属性以及与 ICCP 协议的映射关系，即可实现三种协议到 ICCP 协议的转换。其中，回调函数转换图如图 8 所示。

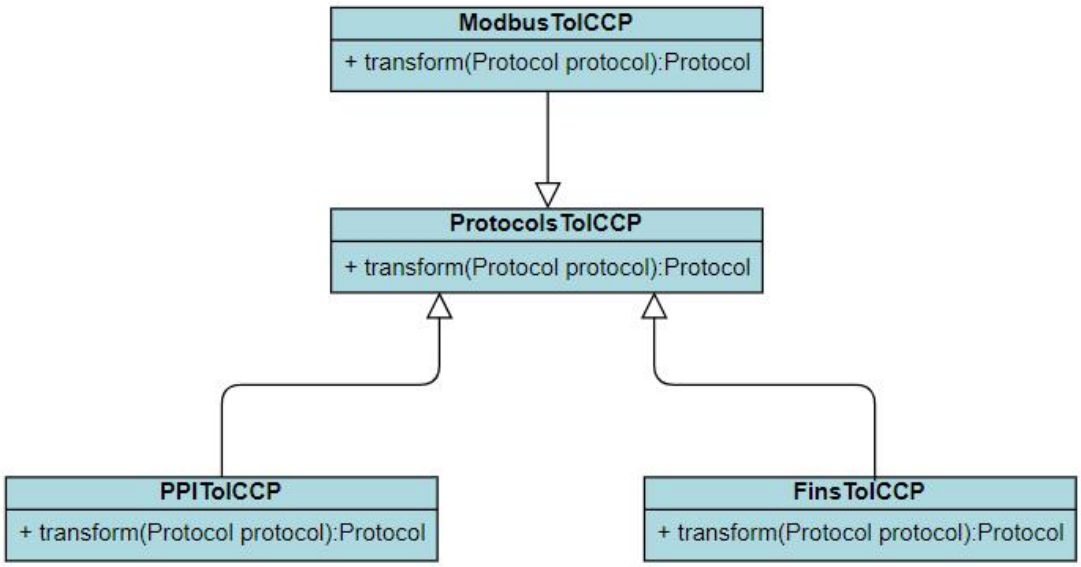

图 8 回调函数转换图

同理，ICCPToModbus、ICCPToFins、ICCPToPPI 三个类实现 ICCPToProtocols 接口，按照各自的属性与映射关系，实现接口，如图 9，即可实现 ICCP 协议到三种协议的转换。

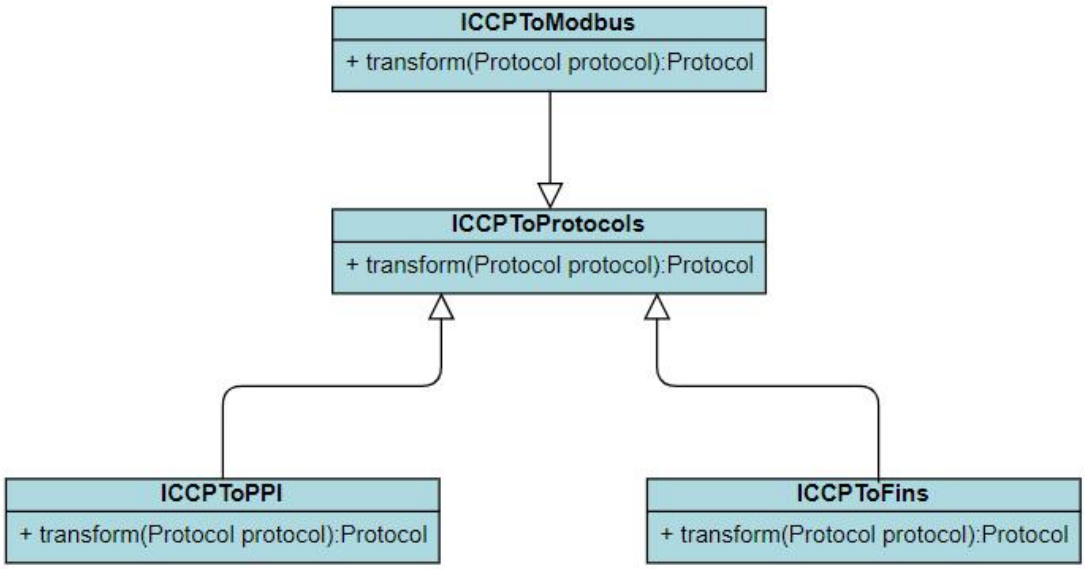

图 9 ICCP 协议到三种协议的转换

对于新协议的引入，无需实现新协议与现有生态中每一种协议的转换，只需实现新增协议到 ICCP 协议的转换接口以及 ICCP 协议到新增协议的转换接口即可实现新增协议与现有生态中的每一种协议任意转化。对于工作量有明显提升，可大幅度减少代码重复使用。

（5）日志记录

网关作为数据交互的关键节点，会产生大量日志信息。项目对所有日志分为两类，网关系统信息与协议转换信息，考虑到上位机唯一，协议转换信息可按照下位机节点进行分类。每一类为该下位机节点所产生的所有协议转换信息。

在网关一些关键处记录日志，包括网关启动，网关初始化信息配置，网关收到 socket 连接等。对协议转换日志，记录协议转换详细流程，以 Modbus 协议转 Fins 协议，记录 Modbus->ICCP、ICCP->Fins 的转换流程，同时还应记录网关代理建立连接所产生的报文以及下位机的回复报文转换。

（6）开发关键技术栈

网关使用 Maven 构建工程，使用原生 Java 语言开发，以太网通信采用 Socket 编程，串口通信采用开源框架 jSerialComm，以下为 jSerialComm 的 maven 依赖。

```
<dependency>
  <groupId>com.fazecast</groupId>
  <artifactId>jSerialComm</artifactId>
  <version>[2.0.0,3.0.0)</version>
</dependency>
```

网关的配置信息，设备节点信息以及日志信息均存储在 MySQL 数据库中，持久层使用 Mybatis 框架简化数据库操作，将数据库表映射为 Device、Log 等 pojo。

网关系统最终以 jar 包形式上传到网关树莓派并运行,仅需在树莓派上提前布好数据库文件与语言环境，即可一键启动。

3.3 网关配套软件

3.3.1 功能详情

（1）设备管理

借助网关配套 web 界面，可在网关上注册下位机设备，将下位机设备接入网关生态进行协议转换，通过设备的网络拓扑 ID、IP 和协议类型等信息就可以实现简单设备注册，同时提供设备增删改查接口用于设备信息更新。配套软件对不同的下位机采取统一的管理操作，包含基础的增删改查和日志查询，简化了管理模式，提高控制效率，降低管理成本。

（2）日志记录

为了使日志输出一目了然，包含下位机 ID 和 IP，操作发生时间、操作内容和日志类型。日志信息按照日志类型可以分为网关系统日志与协议转换日志。并且为了切合实际使用的需求，日志记录查询可分为部分查询和整体查询两类。

部分查询是通过设备管理模块中,下位机信息中的日志功能模块输出该下位机对应的日志记录，使得专门查阅某个下位机的操作日志时迅速便捷，整体查询则是展示全部的下位机整体的日志记录，用以处理需要观察下位机群整体的运行状态的情况。

（3）黑名单管理

切合网关安全需求，提供了上位机黑名单功能，根据 IP 地址屏蔽所选上位机的使用，并记录封禁的原因，用以配合后续进行解禁或修改备注原因等操作，以保障网关的安全和正常使用。

（4）数据库表设计

设备表(Device)

字段	设备 ID	设备 IP	设备类型	注册时间	设备状态
编码	idp	ip	type	time	status
类型	varchar	varchar	varchar	datetime	varchar
约束	非空，主键 1	非空，主键 2	非空	非空	非空

日志表(Log)

字段	设备 ID	设备 IP	日志时间	日志内容	日志类型
编码	idp	ip	time	content	type
类型	varchar	varchar	datetime	text	varchar
约束	非空，主键 1	非空，主键 2	非空，主键 2	非空	非空

黑名单表(Blacklist)

字段	上位机 IP	封禁原因
编码	ip	reason
类型	varchar	text
约束	非空，主键	非空

3.3.2 开发技术

（1）前端

Web 前端页面方面，在主 JavaScript 的应用框架下，通过链接 html 和 CSS 文件，使用 JQuery 和 Vue 渐进式 JS 框架进行构建，使得页面可以极大程度上分割成可复用的组件，提高开发效率。同时应用 ColorUI 等组件库，实现按钮、长文本输入框、弹窗等组件功能实现。在前后端通信方面，前端利用后端提供的服务器地址，在 Vue 中嵌套 Axios 异步框架使用 ajax 发起对后端的请求，返回 JSON 数据对象进行交互。

（2）后端

项目通过 Maven 进行构建，部署于 Tomcat 服务器上运行，进行功能调试和纠错，在持

久层使用 MyBatis 框架，配置连接 XML 文件，构造 Mapper 接口，构造方法实现对数据库的操作映射。

使用 SqlSessionFactoryUtils 工具类，减少重复冗余的 JDBC 原生代码编写，并且可以避免由于多次调用而重复创建 SqlSessionFactory 对象的弊端，以此减少内存压力。

通过 Servlet 类接收来自前端的请求和携带的 JSON 数据，并创建相应的 Service 对象执行对数据库的操作，将结果集封装为对应类的对象，再转化为 JSON 数据格式发送回前端进行响应。

3.4 多协议上位机

3.4.1 动态构造报文

借助上位机软件的自动构造报文的逻辑控制,通过使用者的选择项动态生成所要发送的报文，简化构造报文的过程，避免由频繁手动构造报文和手动输入造成的易错和繁杂，使用起来事半功倍，便捷高效。

3.4.2 前端交互

前端页面分为四个区域：导航栏，可选区，发送区和显示区。

左侧区域，实现协议导航栏，清晰化 PPI、Fins、Modbus 和 ICCP 四个协议的界面划分。

在可选区，实现报文构造，以各个协议报文逻辑构造方式为基础，用键选和列选的方式实现独立报文生成，并对不符合构造逻辑的行为进行弹窗提示。

在发送区，接收点击生成报文按钮传递来的字符串并显示在长文本输入框中，或直接在输入框中输入和修改报文，点击发送按钮可进行协议发送通信。

在显示区，利用 html5 浏览器缓存实现体现在发送区上的协议通信数据信息缓存，使得上下位机协议可视化更为清晰，使用 ajax 技术前端与服务器的通信迅速，无需刷新定时器即可体现协议的交互逻辑。

3.4.3 后端通信

为了实现网关和上位机的连接，通过 socket 编程实现上位机与网关的网络通信，将动态生成的报文发送至网关处，并等待来自网关的信息回复，待收到回复信息则展示到前端页面。

3.5 基于树莓派的下位机

3.5.1 组网与通信

ModBus-TCP 与 Fins 协议基于 TCP/IP 网络，借助 Socket 编程与网线和交换机即可实现组网通信，而 PPI 协议基于 RS-485 串行总线，树莓派本身并不提供串行接口，PPI 协议解析的第一步需要实现串口通信。通过在树莓派上加装 RS-485 串行口扩展板，为树莓派提供串行接入的能力，采用 USB 转 RS-485 总线，USB 端连接网关，RS-485 端连接树莓派 PPI 下位机，即可成功实现串行链路组网。

Python 因其类库丰富而应用广泛，支持 Socket 网络编程同样也支持串口通信，借助 python 内置 pyserial 库，开启树莓派串口功能，网关端口（USB Serial）与 PPI 下位机端口（/dev/ttyAM0）同时开启，write、read 特定 PPI 协议报文实现通信。

3.5.2 协议解析

（1）Modbus 协议解析

Modbus 协议解析前需对目标单元号（第 7 字节）进行验证，对于目标单元号与自身单元号不符的报文一概忽略不做处理，其次则为数据提取，包括功能码提取，起始地址提取，操作数量提取，序列号提取。针对不同的功能码以不同的格式提取具体数据，同时针对不同的功能码执行不同的操作，下位机采用多个数组分别维护离散输入寄存器（Discreate Input），线圈（Coils），输入寄存器（Input Register），保持寄存器（Holding Register），对于寄存器的操作实则是对相应数组的操作。

以位（bit）为单位的寄存器通常为 Boolean 变量，标识着开与关两个状态，工控场景中常用于控制电源开关等两极信号。在本项目中，将位寄存器映射到二极管 LED 灯，每一个位寄存器被映射到指定的 GPIO 口用于控制特定的 LED 灯，读位寄存器则位读取 LED 灯开关状态，而写位寄存器则为控制 LED 灯开关，以写 00 00 号线圈为例，写入数据 1 则为点亮 0 号线圈对应的 LED 灯，写入数据 0 则为熄灭 0 号线圈对应的 LED 灯。

以字（16bits）为单位的寄存器通常为实际生产中的各种物理量，如温湿度传感器采集到的温湿度，CO2 传感器采集到的 CO2 浓度等。对于以字为单位的寄存器的读写操作，则被映射到对各种传感器的操作，若 00 号输入寄存器对应了 dht11 传感器的温度部分，则 04 功能码读取 00 号寄存器则为读取 dht11 传感器采集到的温度。同样写寄存器操作被映射到具体的物理操作，例如温度控制等。

（2）Fins 协议解析

本节将从建立连接和读写功能结合一个建立连接指令、一个读指令和一个写指令对 Fins 协议进行解析。下述指令中假设上位机 IP 地址为 192.168.1.24，下位机 IP 地址为 192.168.1.23。

首先对建立连接指令进行解析。建立连接指令如下：

```
46494E53 0000000C 00000000 00000000 00000018
```

其中：

字段	说明
46494E53	Fins 协议的标识，指示这是 Fins 协议
0000000C	长度字段，指示后面还有 0x0C(12)个字节
00000000	命令码，表示这是建立连接指令
00000000	错误码，代表指令无误
00000018	上位机的地址，0x18 是上位机 IP 最后一个字节的十六进制

建立连接指令的回复指令如下，该指令中前五个字段和建立连接指令字段意义相同，最后一个字段是下位机的地址，0x17 是下位机 IP 最后一个字节的十六进制：

```
46494E53 00000010 00000001 00000000 00000018 00000017
```

然后对读指令进行解析。具体指令如下，其功能是对下位机的 DM 区进行读取，读取两个字，偏移量为 0：

```
46494E53 0000001A 00000002 00000000 80 00 02 001700 001800 FF 0101 82 000000 0002
```

其中：

字段	说明
46494E53	Fins 协议的标识，指示这是 Fins 协议
0000001A	长度字段，指示后面还有 0x1A(26)个字节
00000002	命令码，表示这不是建立连接指令或建立连接回复指令
00000000	错误码，代表指令无误
80	ICF
00	RSV
02	GCT
001700	服务器端地址
001800	客户端地址
FF	SID
0101	读数据的功能码
82	DM 区域读取，且单位为字
000000	偏移量，指示偏移量为 0
0002	读取数据个数

下位机对上述指令的回复指令如下，前部分字段同读取指令，0000 字段代表读取数据成功，AABB 是读取到的第一个字，CCDD 是读取到的第二个字：

```
46494E53 0000001A 00000002 00000000 C0 00 02 001800 001700 FF 0101 0000 AABB CCDD
```

最后对写指令进行解析。具体指令如下，其功能是写入下位机 CIO 区，写两个位，分别写入 0 和 1，偏移量为 0，部分字段同读指令，不再进行说明：

```
46494E53 0000001C 00000002 00000000 80 00 02 001800 001700 FF 0102 30 000000 0002 00 FF
```

其中：

字段	说明
0102	写数据的功能码
30	CIO 区域，且单位为位
000000	偏移量，指示偏移量为 0
0002	写入数据个数，代表写 2 个数据
00 FF	写入的数据值，00 和 FF 分别表示写入 0 和 1

下位机对上述指令的回复指令如下，其中最后的字段 0000 代表写入成功：

```
46494E53 00000016 00000002 00000000 C0 00 02 001800 001700 FF 0102 0000
```

（3）PPI 协议解析

本节将从建立连接和读写功能结合一个建立连接指令、一个读指令和一个写指令对 PPI 协议进行解析。下述指令中假设上位机 ID 地址为 00，下位机 ID 地址为 02。

首先对建立连接指令进行解析。

建立连接指令如下：

```
10 02 00 49 4B 16
```

其中：

字段	说明
10	起始符
02	下位机 ID
00	上位机 ID
49	上位机站号
4E	校验码
16	结束符

下位机在收到上述指令后发送连接回复指令 E5，表示确认收到连接。

然后对读指令进行解析。具体指令如下，其功能是对下位机的 V 寄存器进行读取，读取三个字节，偏移量为 0x320：

```
68 1B1B 68 02 00 6C 320100000000000E00000401120A10 0200 0300 0184 000320 8D 16
```

其中：

字段	说明
68	起始符

1B1B	长度，指示从下位机 ID 到指令结尾的长度，长度字段重复一次
02	下位机 ID
00	上位机 ID
6C	读功能码
320100000000000E00000401120A10	固定字段
0200	数据单位，代表读取单位是字节
0300	数据个数，代表读取三个
0184	寄存器，代表读取 V 寄存器
000320	偏移量
8D	校验码
16	结束符

下位机对上述指令的回复指令如下，部分字段与上述指令相同，其中 0400 字段代表指令是进行的字节的读取，18 字段代表十进制的 24，表示读了三个字节（24 个位）。99、34、56 字段是读取到的三个字节的数据：

68 1818 68 00 02 08 32030000000000002000700000401FF 0400 18 99 34 56 8B 16

最后对写指令进行解析。具体指令如下，其功能为写入下位机 V 寄存器，写入一个字值为 0x1234，偏移量为 0x320，部分字段同读功能相同：

68 2121 68 02 00 7C 320100000000000E00060501120A10 0400 0100 0184 000320 0004 0010 1234 FE 16

其中：

字段	说明
7C	写功能码
0004（第二个）	其中 0003 代表以位为单位写入，0004 代表以字节或字或双字为单位写入
0010	表示十进制的 16，含义是写入一个字节（十六个位）
1234	写入的一个字的数据
FE	校验码

下位机在收到上述指令后发送连接回复指令 E5，表示写入数据成功。

3.5.3 硬件功能执行

（1）dht11 应用

dht11 是一款含有已校准数字信号输出的温湿度复合传感器；包括一个电阻式感湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接；单线制串行接口，信号传输距离可达 20 米以上。

dht11 一次完整的数据传输为 40bit,高位先出依次为:8bit 湿度整数数据+8bit 湿度小数数据+8bi 温度整数数据+8bit 温度小数数据+8bit 校验；DHT11 数据分小数部分和整数部分,当前小数部分用于以后扩展,现读出为零；测量分辨率分别为 8bit（温度）、8bit（湿度）数据传送正确时校验位等于“8bit 湿度整数数据+8bit 湿度小数数据+8bi 温度整数数据+8bit 温度小数数据”之和。温度和湿度数据均为一个字，正好映射到下位机两个以字为单位的寄存器，对寄存器的读实则为读取下位机环境温湿度。

由于上拉电阻，总线空闲状态为高电平。主机发出开始信号，把总线拉低大于 18ms,拉高电平延时等待 20-40us 后，读取 DHT11 的响应信号。在 dht11 接收到主机的开始信号后，等待主机开始信号结束，发送 80us 低电平响应信号，再把总线拉高 80us。主机发送开始信号结束后，可以切换到输入模式，总线由上拉电阻拉高或者输出高电平均可。每一 bit 数据都以 50us 低电平开始，高电平的长短决定数据位是 0 还是 1（26us-28us 为 0；70us 为 1）。当最后一 bit 数据传送完毕后，DHT11 拉低总线 50us，随后总线由上拉电阻拉高进入空闲状态。

将 dht11 的 GND 引脚与 VCC 引脚分别连接树莓派 5V 电源与，数据引脚则连接某一 GPIO 口，借助 python RPi 库，对 GPIO 口按照 dht11 指定规则电平变化与解析，即可读取现场温湿度，对采集到的温湿度数据则更新到下位机对应寄存器，为上位机提供数据。

（2）二极管 LED 灯应用

各种协议均存在以 bit 为单位的寄存器,实际生产中通常用于控制电源开关等两极信号,本项目中，采用二极管 LED 灯形象演示信号执行效果。下位机维护位寄存器到二极管 LED 灯和树莓派 GPIO 口的映射关系，对寄存器的读写操作则为对 GPIO 口的电平操作。

借助 python RPi 库，若下位机解析报文结果为读写位变量，构造回复报文同时，执行具体操作，若上位机对下位机线圈置 ON 状态，则下位机对相应 GPIO 端口发送高电平，效果为点亮对应二极管 LED 灯，若上位机对下位机线圈置 OFF 状态，则下位机对相应 GPIO 端口发送低电平，效果为熄灭对应二极管 LED 灯。

四、应用效果

4.1 网关运行效果

将网关代码打包为 jar 包部署到网关所用树莓派,同时将网关 web 系统部署到树莓派的 tomcat 服务器中，访问 8080 端口下的对应路径，即可查看网关各种信息。

基础信息配置如图 10 所示：

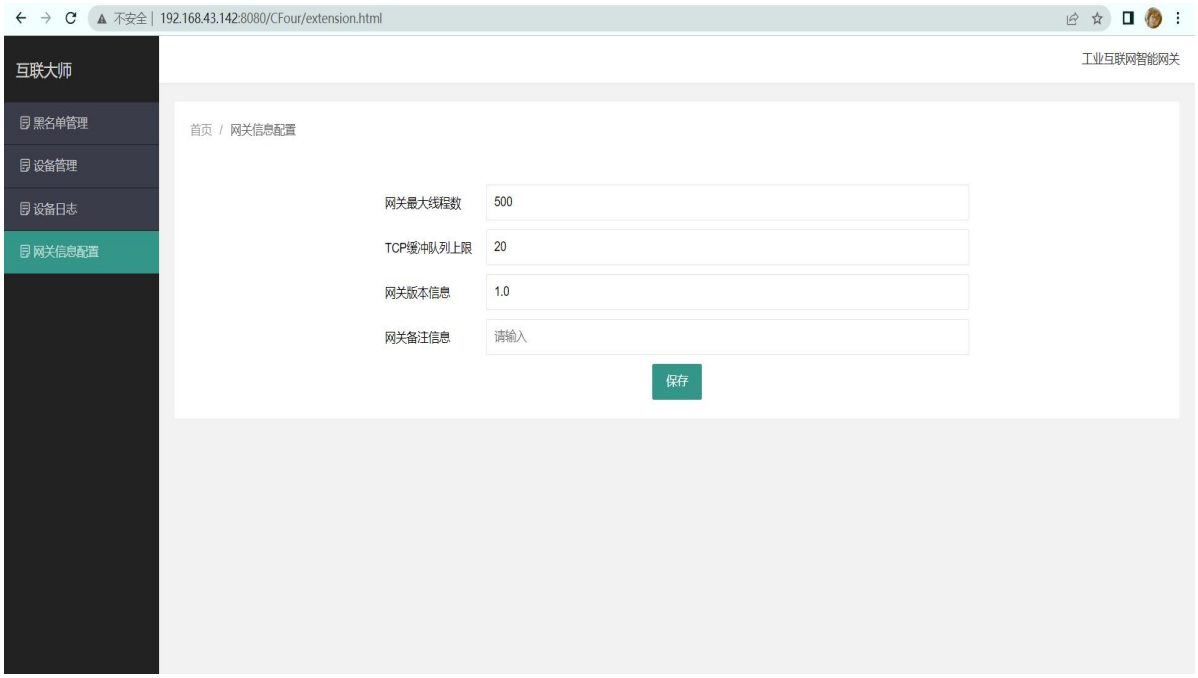

图 10 基础信息配置

设备管理如图 11 所示：

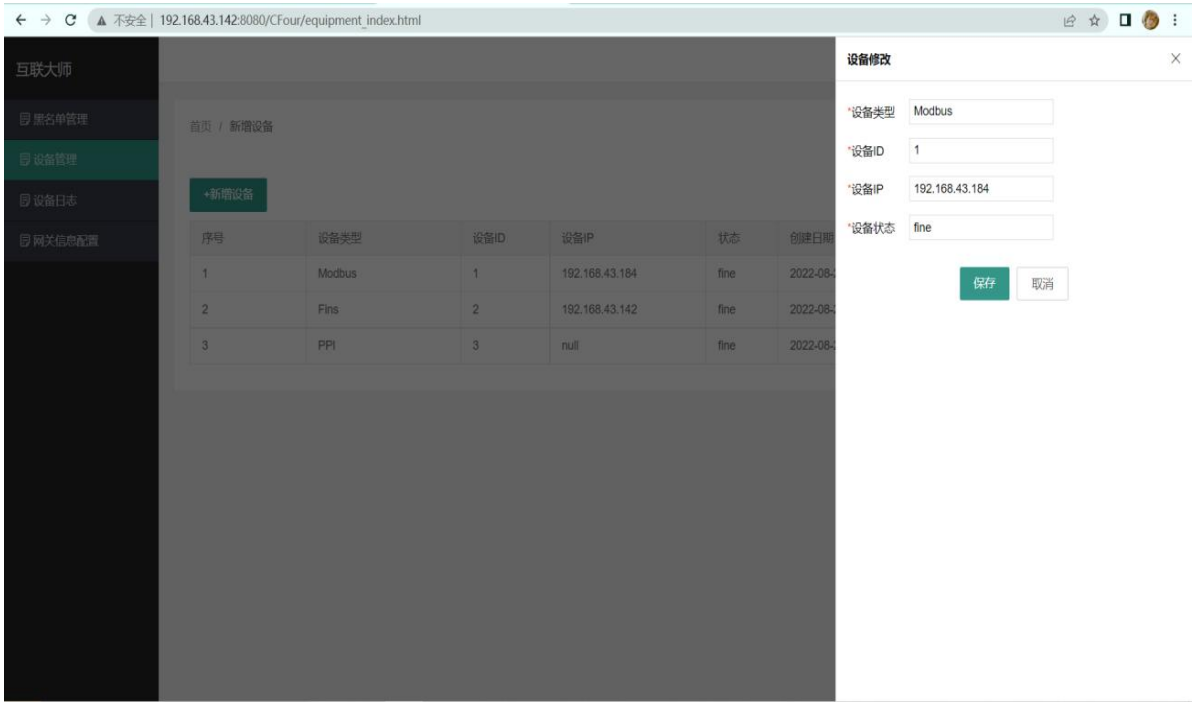

图 11 设备管理前端界面

系统日志：包括协议转换过程以及系统运行的关键日志。具体如图 12 所示。

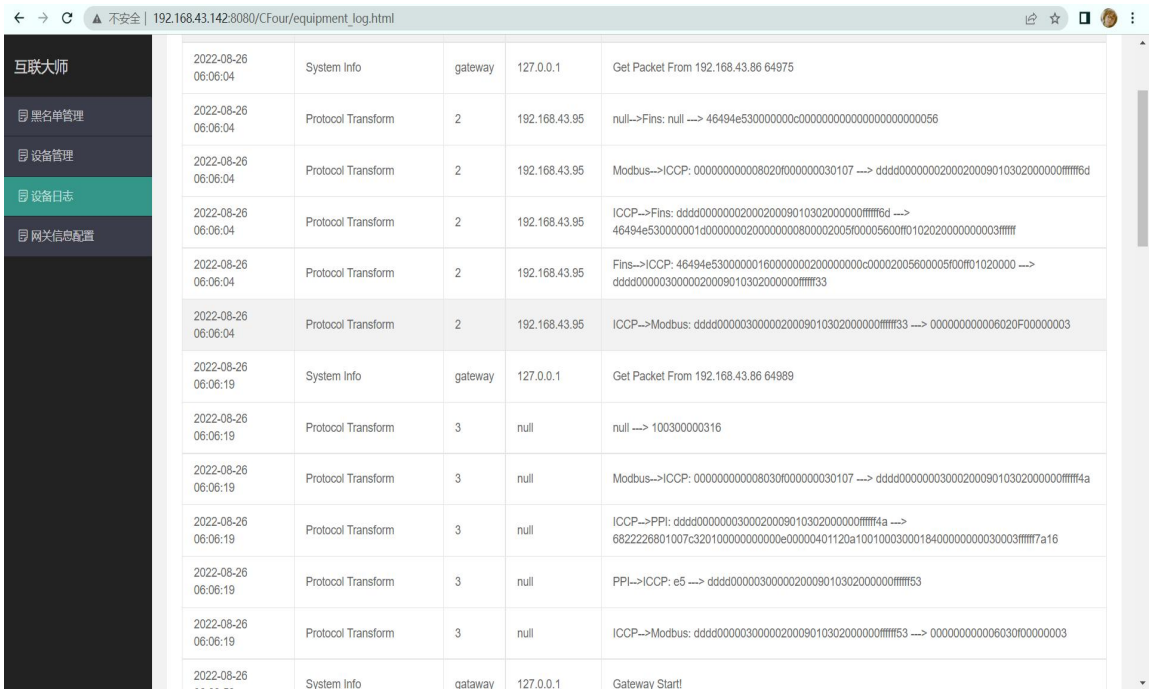

图 12 系统日志前端图

4.2 场景测试效果

项目还开发了云端上位机软件，并采用树莓派构建真实下位机。三个共同组建网络拓扑。上位机软件如图 13 所示：

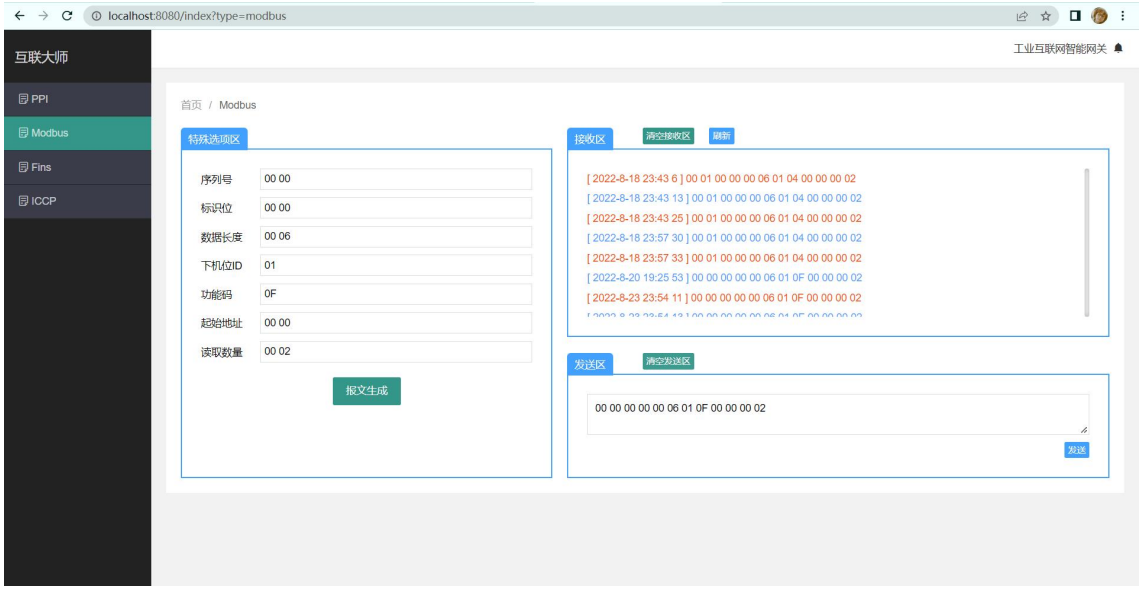

图 13 上位机软件

串口通信局部图如图 14 所示：

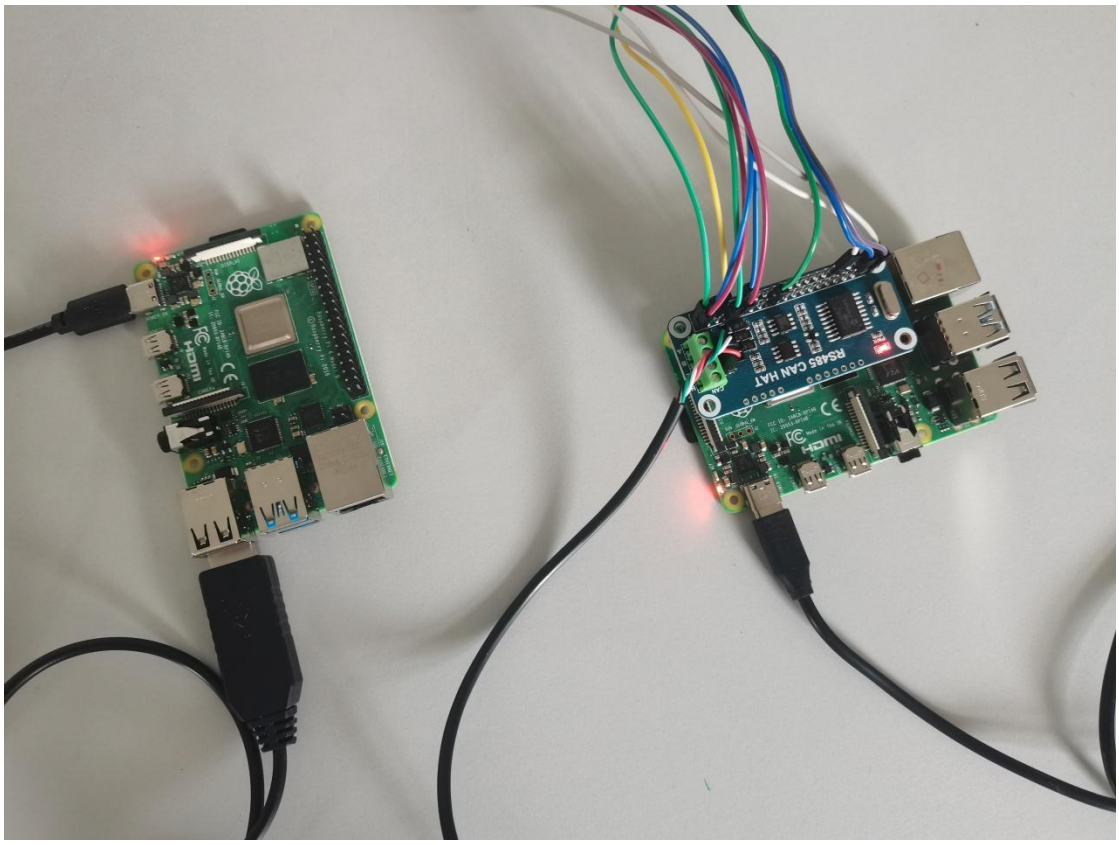

图 14 串口通信局部图

五、创新与特色

5.1 模式新颖

以自定义协议 ICCP 为中间协议实现协议转换，而非传统多协议一对一互转，大幅度减少协议转换冗余，做到协议转换可复用，可扩展。网关引入新协议，只需实现新协议与 ICCP 协议的互相转换，当网关支持协议数量达到一定规模，对协议转换和网关代码扩展有明显工作量的减少。

工业设备上云是当前行业刚需，网关北向只需采用 ICCP 协议对接上位机，与协议转换功能融合，无需新设备，新基础设施的引进即可轻松实现设备上云。基于北向协议 ICCP 的健壮性，开发云端上位机、工业设备接入物联网平台简单易行。

5.2 功能齐全

协议转换的核心为关系映射，从编址信息到核心数据，从差错检验到通信网络，网关对于不同协议的各种报文，能够做到正确解析，差错检验，协议对象封装，对应字段映射，目标报文构造，跨网络介质通信。

考虑到网关的健壮性，网关提供配套界面实现可视化控制与数据展示。提供网关基础信息配置接口。提供设备注册接口，可动态扩展下位机拓扑，接入新的下位机。通过黑名单控制实现上位机报文过滤，通过日志转换记录协议转换效果便于后续数据追踪。

5.3 架构合理

网关作为上下位机的中间站，维护着多个上下位机通信对，每一个通信对则启动新的线程处理业务，各个通信对之间互不影响，单点故障和阻塞不影响网关整体功能。

基于 Java 面向对象的思想，将报文解析的结果封装为协议对象，协议转换简化为对象间简单赋值。协议对象提供方法生成报文。一个对象与一个协议报文唯一对应，实现网络报文与主机上的协议对象随意切换。

协议转换以接口形式实现，分为协议标准化接口与标准协议目标化接口，各协议转 ICCP 协议仅需按自身协议特性实现协议标准化接口，ICCP 协议转其他协议也仅需按目标协议特性实现标准协议目标化接口。将接口以回调函数形式传入协议转换线程。实现协议转换解耦合。网关引入新协议，仅需实现协议标准化接口与标准协议目标化接口即可。

5.4 应用实际

网关当前支持 ModBus、Fins、PPI 协议的相互转换，同时支持 ModBus、Fins、PPI 协议与 ICCP 协议的相互转换，实现远程控制 3 种协议下位机。项目本着开源的精神，基于已有源代码可引入新协议实现互相转换，只需简单配置即可投入使用。

在网关系统基础上，项目开发了多个基于树莓派的不同协议下位机，以及基于 spring boot 的多协议上位机。借助串行线、网线、无线 WIFI、交换机、树莓派搭建网络拓扑，对网关系统实战测试。验证了网关的实用性与高效性。