



网关配置工具使用手册

LMGateway Configuration Tool User Manual

版本：1.7

日期：2020-10-14

Table of Contents

前言	1.1
概述	1.2
第一章 工程管理	1.3
1.1 新建工程	1.3.1
1.2 打开工程	1.3.2
1.3 删除工程	1.3.3
第二章 网关管理	1.4
2.1 添加、搜索、查看网关	1.4.1
2.2 实时数据、设置Tag点值	1.4.2
2.3 系统设置	1.4.3
2.4 调试模式	1.4.4
2.5 HTTP接口	1.4.5
2.6 WEB服务器	1.4.6
第三章 数据采集配置	1.5
3.1 通道配置及协议选择	1.5.1
3.2 设备及点表编辑	1.5.2
3.3 用户点	1.5.3
3.4 计算点	1.5.4
3.5 系统点	1.5.5
第四章 数据存储	1.6
4.1 数据存储配置	1.6.1
4.2 添加存储点	1.6.2
第五章 数据服务	1.7
5.1 Modbus	1.7.1
5.2 BACnet	1.7.2
5.3 OPC UA	1.7.3
5.4 HTTP	1.7.4

5.5 远程数据库	1.7.5
5.6 OPC XML-DA Server	1.7.6
第六章 IoT	1.8
6.1 Mqtt Client	1.8.1
6.2 IoTDDC	1.8.2
6.3 Alink	1.8.3
6.4 Tlink	1.8.4
第七章 任务计划	1.9
第八章 报警与事件	1.10
8.1 设置报警条件	1.10.1
第九章 JavaScript脚本编辑	1.11
9.1 操作步骤	1.11.1
9.2 函数说明	1.11.2
9.3 使用示例	1.11.3
第十章 运行工程	1.12
第十一章 模板	1.13
第十二章 应用实例	1.14
12.1 新建工程	1.14.1
12.2 新建网关设备	1.14.2
12.3 新建通道	1.14.3
12.4 新建设备	1.14.4
12.5 添加数据点	1.14.5
12.6 数据服务	1.14.6
12.7 应用工程	1.14.7
第十三章 附录	1.15

写在最前面

Gateway Configuration（简称GC）是黄山罗米测控技术有限公司开发的用于配置和监控网关的一款配置软件，文档主要是介绍GC的使用。

LMGateway或网关均指的是罗米测控推出的各型号版本的数据采集网关。

GC能干吗

GC的运行环境为Windows XP及以上操作系统，需要安装.NetFramework 4.0。

配置采集驱动、数据服务、IoT、数据存储、事件报警、计划任务和JavaScript脚本编程。

查看网关实时数据、通讯报文、运行日志。

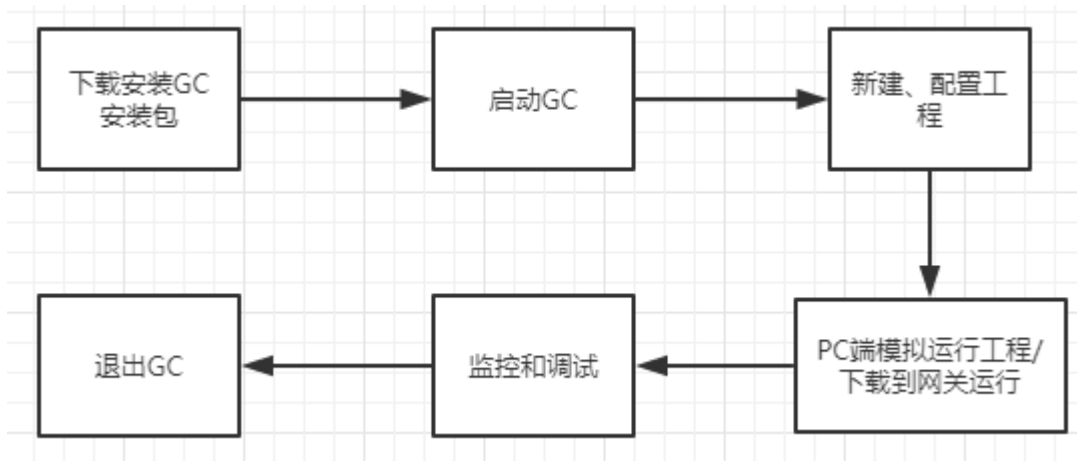
GC可在PC端脱离网关模拟运行工程，没有授权可模拟运行2小时。

本书主要内容

- 第一章：主要介绍GC的工程概念与使用。
- 第二章：主要介绍GC对LMGateway的管理。
- 第三章：主要介绍GC采集协议的驱动配置与各种内部点说明。
- 第四章：介绍GC的数据存储功能。
- 第五章：介绍GC的转发协议配置等操作说明。
- 第六章：介绍GC基于MQTT协议的云服务配置等操作说明。
- 第七章：介绍GC任务计划的运行方式与操作说明。
- 第八章：介绍GC报警的设置以及报警事件与报警恢复事件的存储与查询操作说明。
- 第九章：介绍GC的JavaScript自定义脚本的运行方式与API说明以及demo示例。
- 第十章：介绍工程在GC和在LMGateway运行的操作说明。
- 第十一章：介绍GC的模板操作说明。
- 第十二章：介绍GC从无到有配置工程的过程。

概述

GC使用步骤



GC运行流程

GC 安装/卸载

GC安装包可在罗米官网<http://www.lmgateway.com/> 下载，可稳定运行在Windows XP/ Windows 7/Windows 10系统下。

安装

右键“以管理员身份运行”安装包中的LM_GateWay_Setup.exe，选择安装的路径。

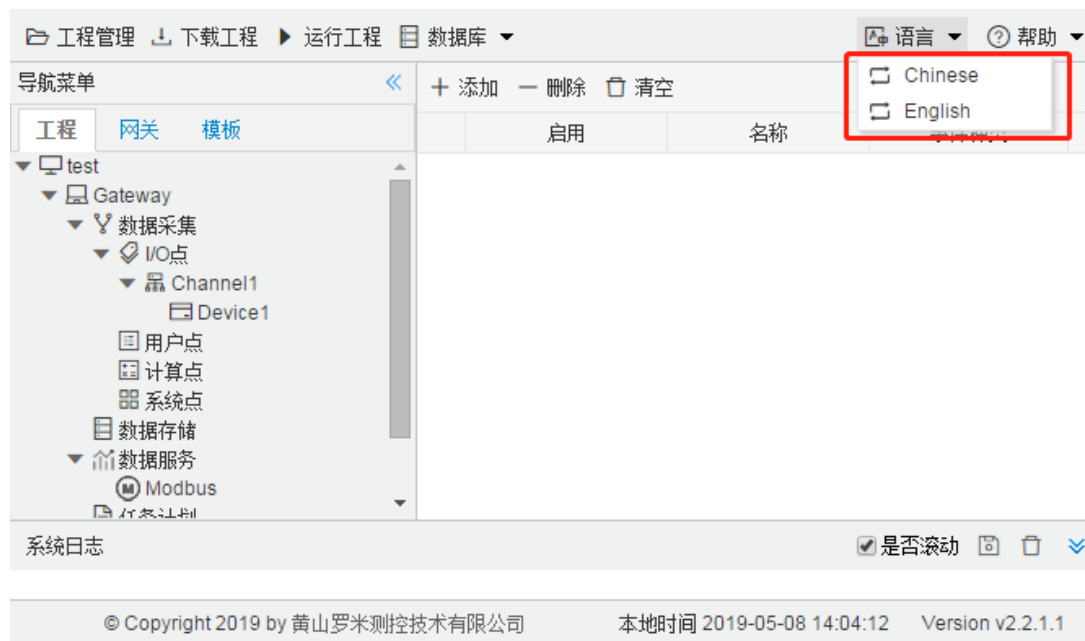
注意：window10由于安全策略的问题，不能把软件安装在系统盘下。

卸载

双击安装目录下的uninst.exe，按照卸载步骤即可完成GC的卸载。

语言切换

GC集成了中文、英文两种语言环境，用户可以根据需要在GC语言工具栏中选择相应的语言环境。



第一章 工程管理

工程是GC对网关的数据采集、数据服务及其他各项功能生成的配置文件，LMGateway根据该工程文件进行相应的采集和服务。

工程文件保存在GC安装目录的Project文件夹下，LMGateway在线时可将GC中打开的工程下载到网关，网关自动加载并重启运行。

工程管理主要分为新建工程、编辑工程、删除工程。

运行GC，点击工具栏中的“工程管理”按钮，弹框列举GC中所有工程。

1.1 新建工程

在“工程管理”对话框中，单击“新建工程”；在弹出的“新建工程”对话框中，输入工程名称，单击“确定”，操作步骤如图 1-1 所示。



图1-1 新建工程

工程新建完成以后，会在左侧工程树中显示新建工程的名称作为根节点，之后的工程配置都在此处进行，如图1-2所示。



图1-2 新建工程完成

添加网关

单击选中工程树中工程名称节点，右键单击选择“添加网关”，自定义网关名称和备注，选择网关类型，具体操作如图1-3所示。

提示：网关的侧标签有网关型号、网关版本和默认IP等信息。

注意：下载工程到网关时会进行网关类型和版本的验证。

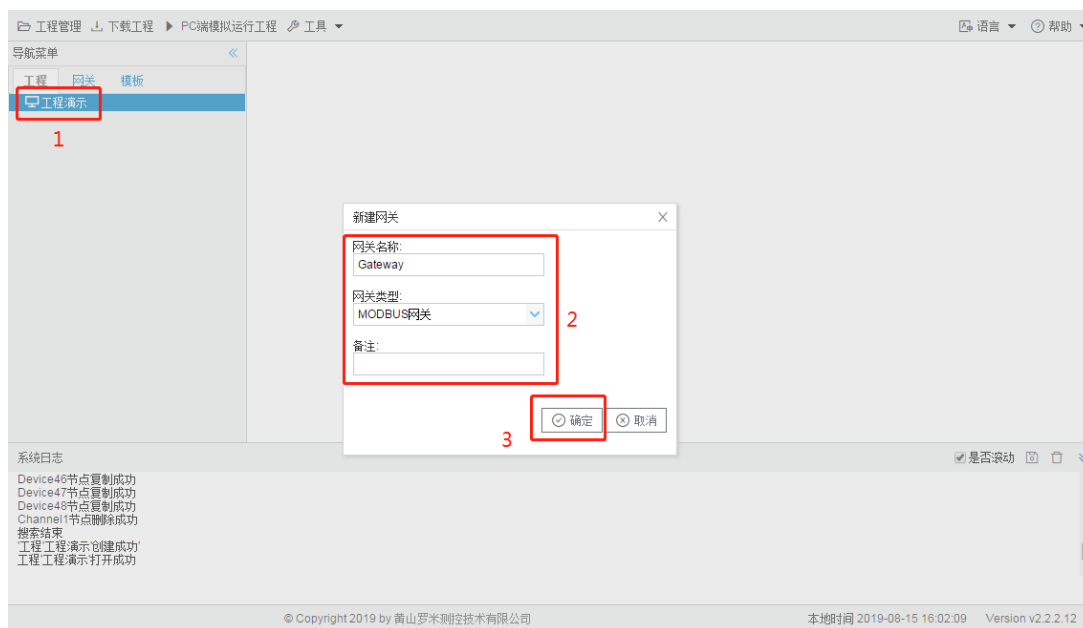


图1-3 添加网关

网关添加完成以后，在工程名称节点下会根据网关型号的不同，显示相应的工程树，如图1-4所示。

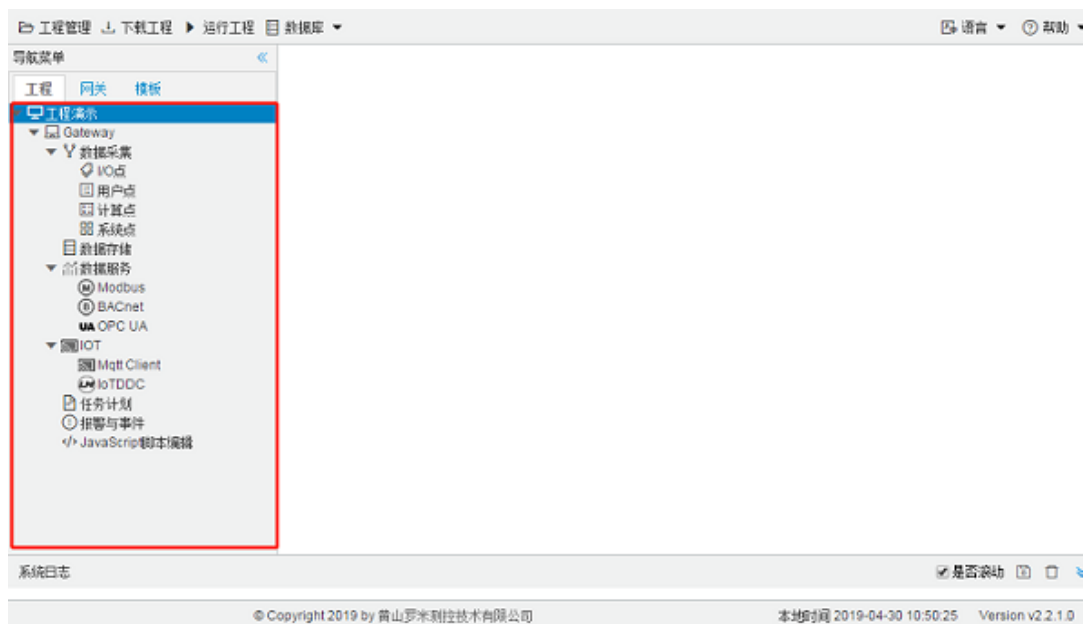


图1-4 添加网关完成

1.2 打开工程

在“工程管理”对话框中，单击选中需要打开的工程，单击“打开工程”按钮。操作步骤如图 1-3所示。



图1-3 打开工程

在打开工程完成之后，会在左侧工程树中显示选中工程的配置，如图1-4所示。



图1-4 打开工程完成

1.3 删除工程

在“工程管理”对话框中，单击选中需要删除的工程，单击“删除工程”按钮，在弹出的系统提示框中单击“确定”，完成删除工程的操作。操作步骤如图 1-5 所示。



图1-5 删除工程

第二章 网关管理

GC可以对网络中所有LMGateway进行管理，将装有GC的PC与网关联网，保证PC可以ping通网关，具体配置如下：

- 添加、搜索、查看网关；
- 查看实时数据，对Tag点值进行设置；
- 设置网关IP、密码，进行校时、重启等操作；
- 调试模式。

2.1 添加、搜索、查看网关

添加网关

已知LMGateway的IP地址，点击“添加设备”将网关添加到在线网关列表中。



图2-3 添加网关

搜索网关

将网络中的LMGateway全部搜索并列到在线网关列表中。

在“系统设置”的“网关备注”窗口中填写了备注，“搜索网关”时会将备注显示在IP之前。

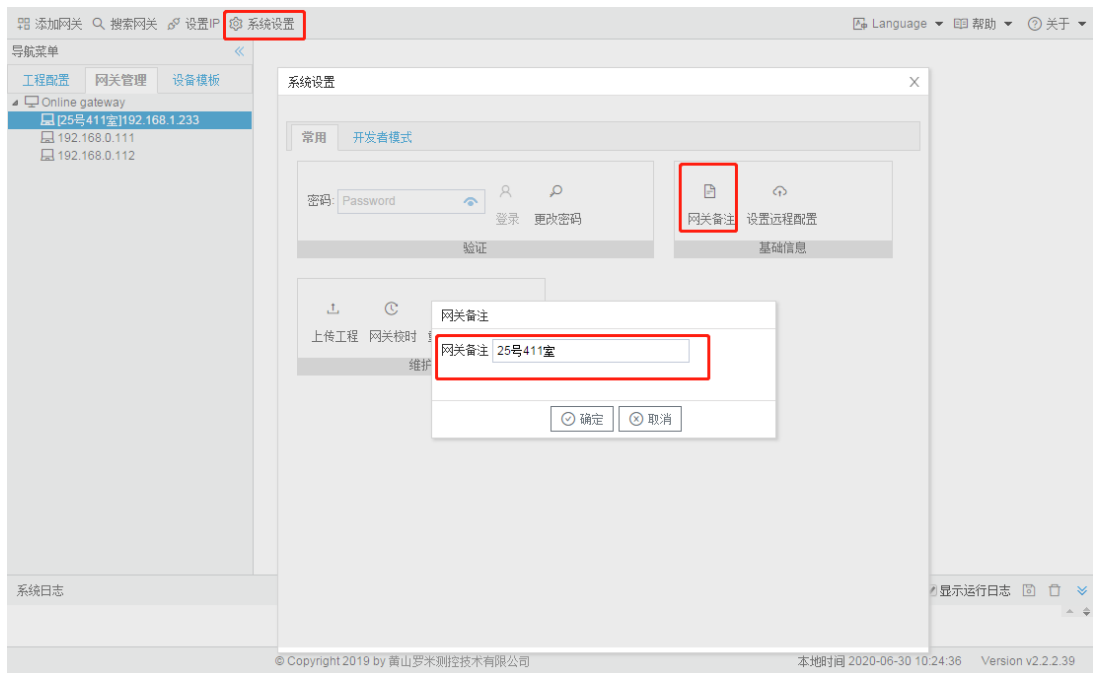


图2-4 设置网关备注



图2-4 搜索网关

查看网关信息

双击网关IP，弹出窗口中包含网关类型、软件版本、支持点数、网关备注、网关ID和SN号，其中支持点数是指网关中的数据点低于支持点数时可以保证网关的运行效率。

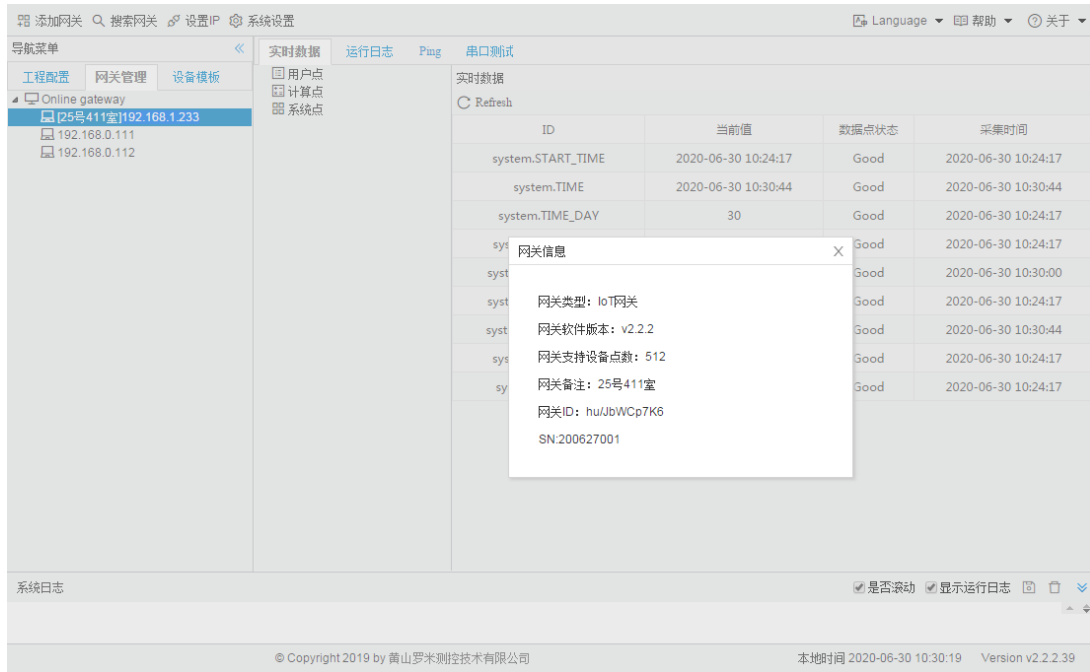


图2-5 查看网关信息

2.2 实时数据、设置Tag点值

查看实时数据

GC具有监控LMGateway数据的功能，单击网关IP，右侧显示通过 HTTP接口查询到的所有实时数据，可以通过单击左侧节点查看该节点下的数据。

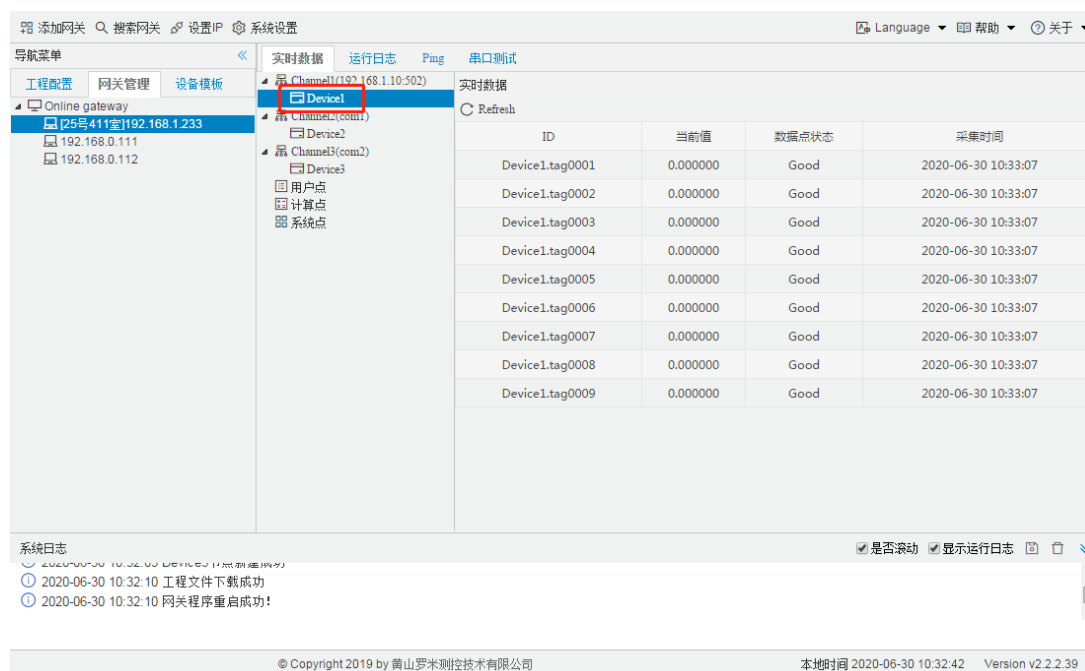


图2-6 实时数据

实时数据页面有以下字段：

- 名称：Tag点在当前LMGateway中的唯一标识。
- value：Tag点的实时值。
- quality：Tag点的质量戳，Good表示采集成功，此时value显示采集到的值；Error为采集失败，此时value显示为空。

设置Tag点值

GC可以对网关的数据点进行写操作。

在实时数据页面，单击Tag点的value字段，在弹出的窗口中输入需要写入的值，点击“写入”按钮。

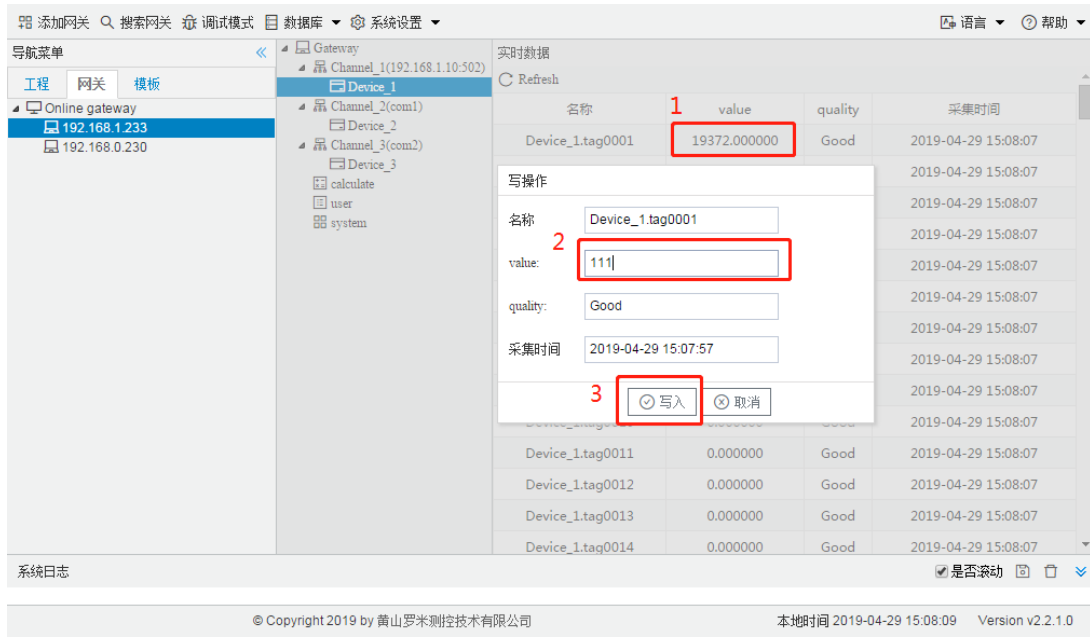
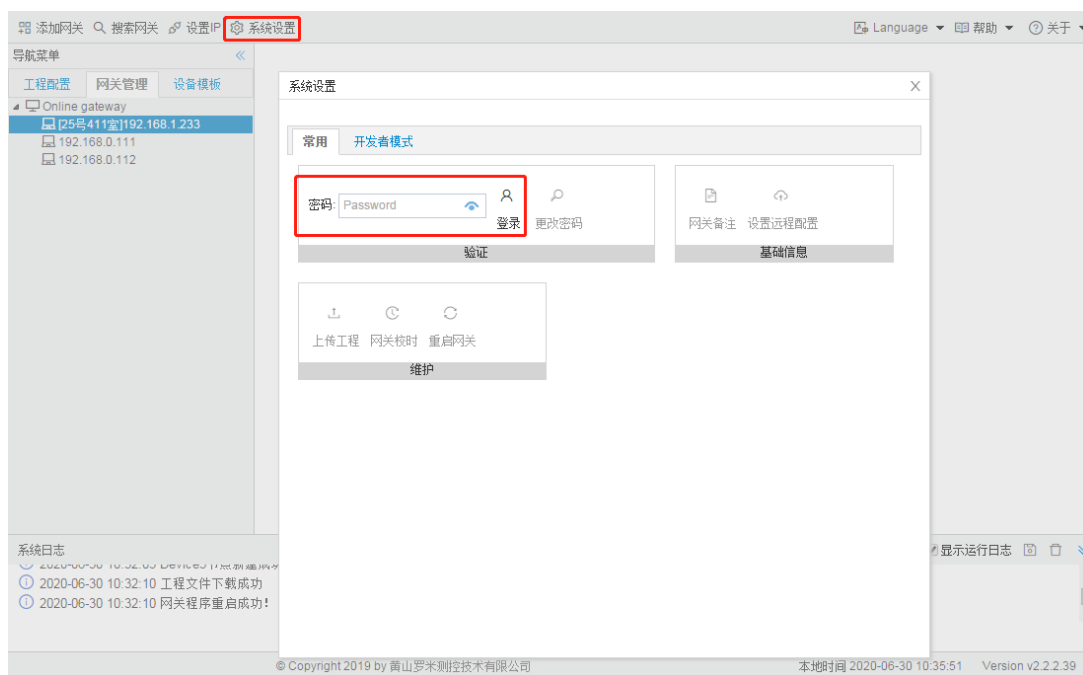


图2-7 设置Tag点值

2.3 系统设置

点击“系统设置”按钮，在弹出的“系统设置”框中输入登录密码后点击“登录”按钮(出厂默认无密码，可直接点击“登录”按钮)。



2.3.1 设置IP

用户可以通过GC修改在线LMGateway网口的网络参数。

点击“设置IP”按钮，弹出"IP地址配置"窗口。

设置有线IP:

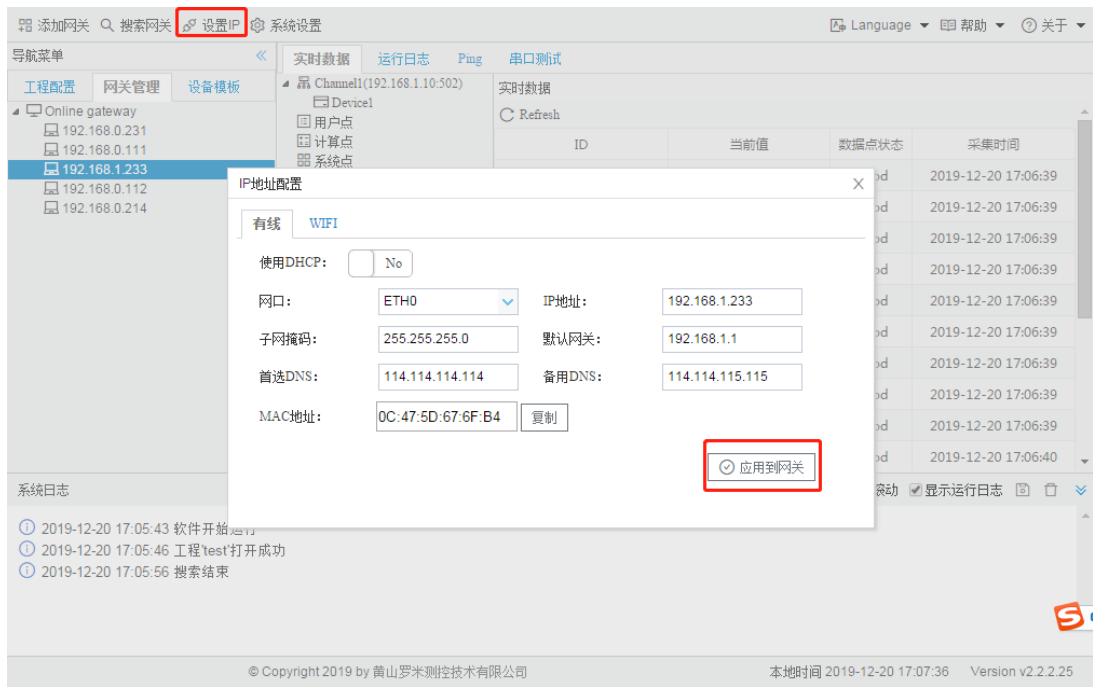
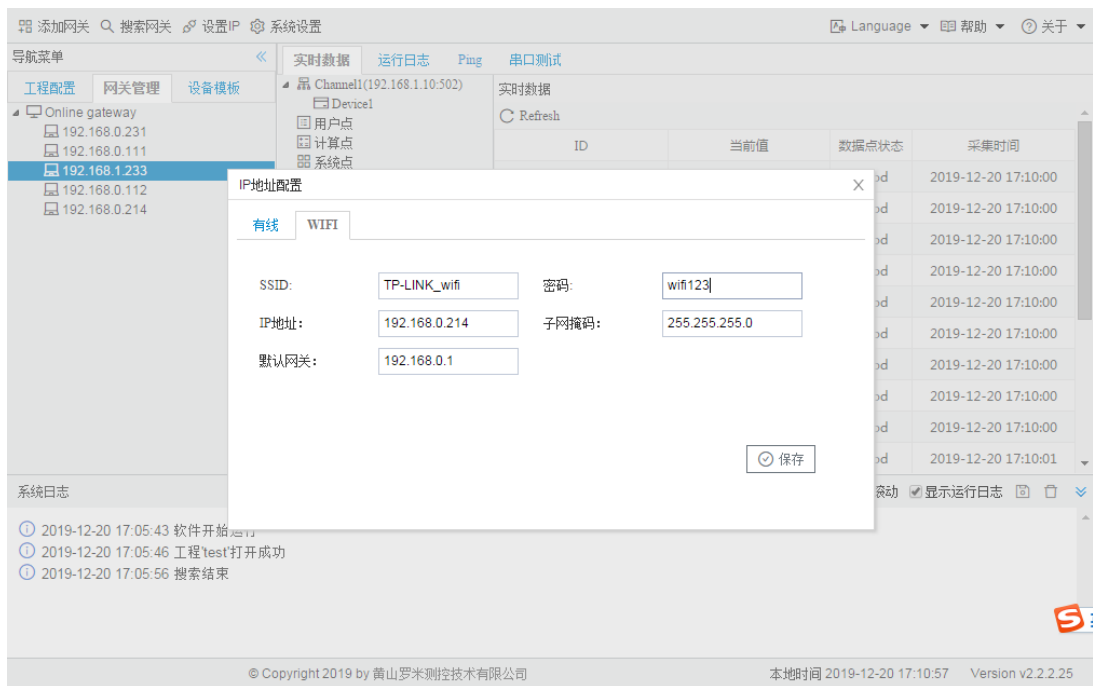


图2-8 设置IP

点击“应用到网关”按钮，网关立即生效。（网口支持DHCP）

设置WIFI:



将SSID、密码、IP信息正确填写完成后，点击“保存”，将usb无线网卡插入到网关的usb端口，重新启动网关即可使用usb无线网卡。

2.3.2 网关备注

GC可以给LMGateway设置备注，在搜索网关时显示该备注，便于客户管理。

点击"系统设置"框中“网关备注”按钮进行备注设置。



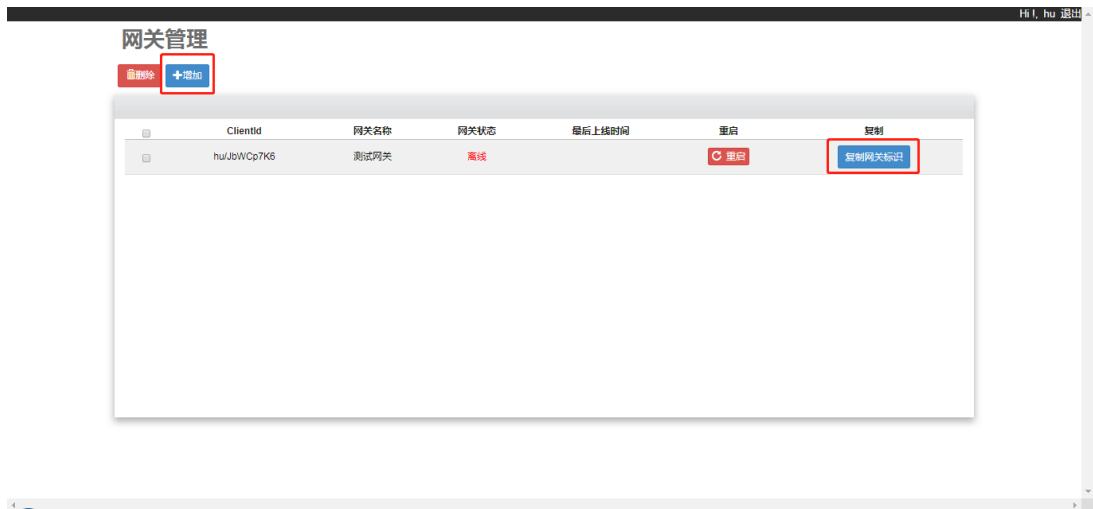
网关备注

2.3.3 设置远程配置

GC可以通过远程部署系统，将工程远程下载到网关当中。

1. 通过浏览器登录www.iotddc.com:8081，登录注册一个用户，之后添加网关，“复制网关标识”





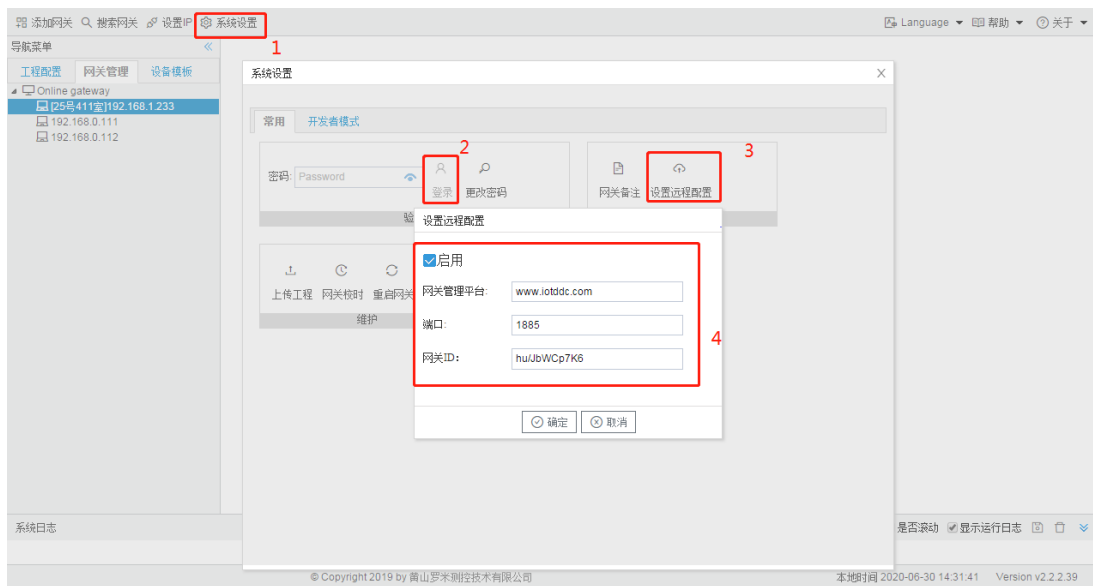
“网关标识”为远程部署系统中网关的唯一标识，每个网关的标识不能重复。

1. 点击GC的"系统设置"框中“设置远程配置”按钮，分别填写

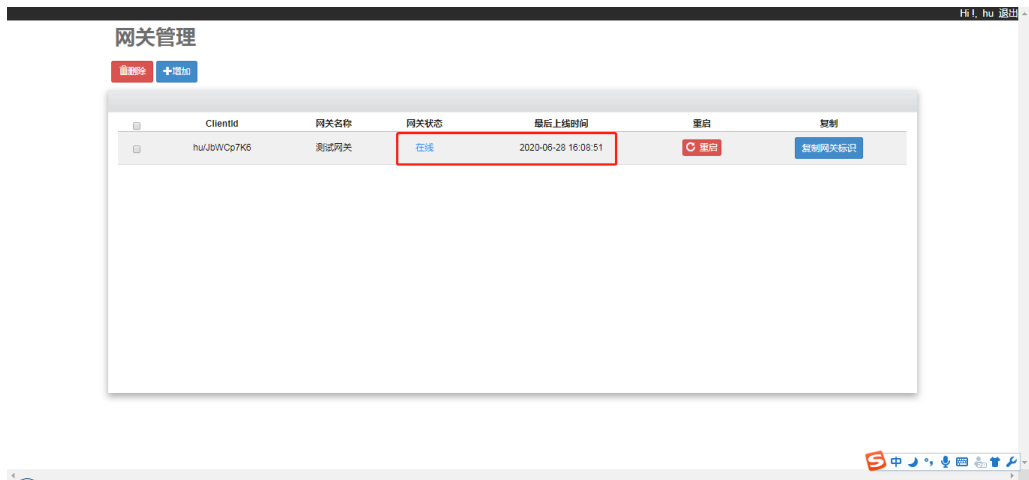
网关管理平台：www.iotddc.com

端口：1885

网关ID：复制的网关标识

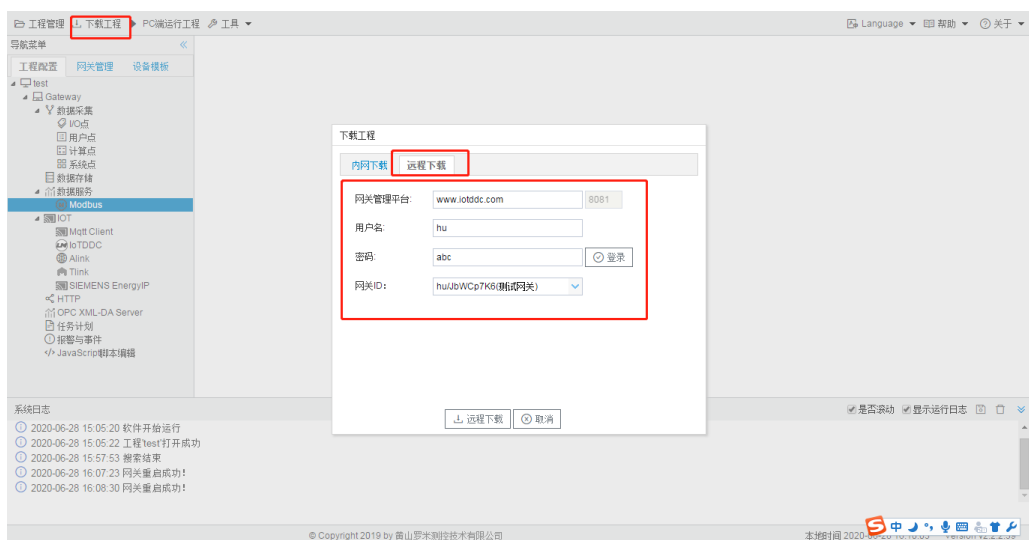


1. 重启网关，网络正常时平台页面会显示“在线”



用户也可通过此页面管理所有LMGateway，查看最后上线时间、进行网关重启。

1. 显示在线时，说明网关与部署系统连接正常，此时可通过配置工具进行远程配置



输入用户名和密码，点击“登录”按钮，在网关ID的下拉框中会显示该用户下所有的网关标识，选择需要将该工程下载到的网关标识，点击“远程下载”。

2.3.4 上传工程

GC可以将当前网关的配置文件上传到本地。

点击"系统设置"框中“上传工程”按钮，工程文件会上传至GC安装目录的Project文件夹下。

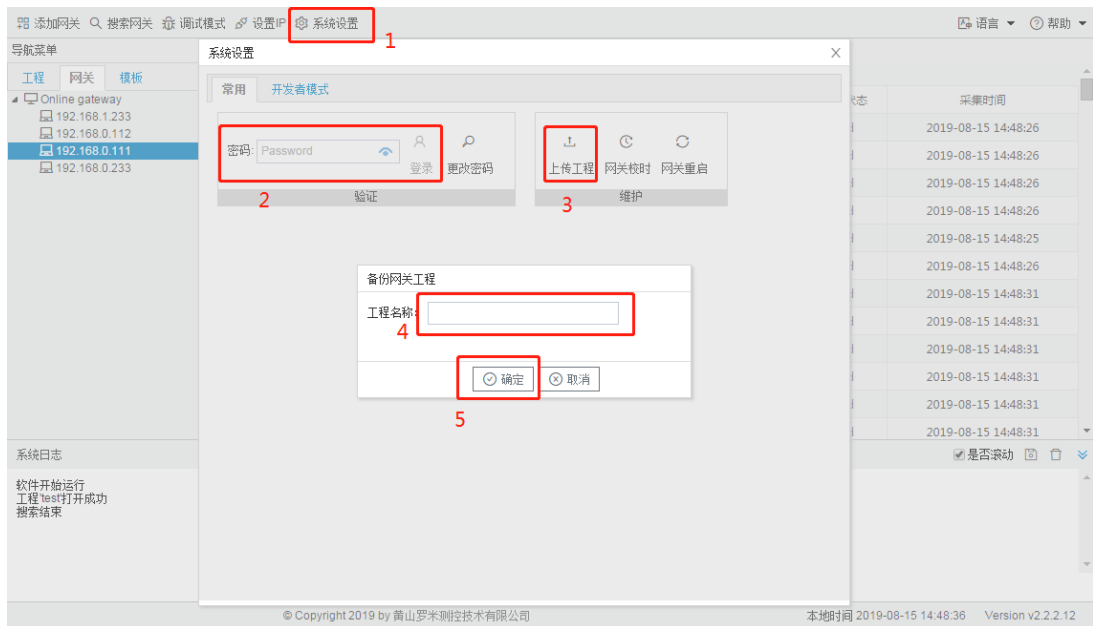


图2-9 上传工程

上传工程成功后GC会自动打开上传的工程。

2.3.5 密码设定

点击"系统设置"框中"密码设定"按钮，弹出"密码设定"窗口。

LMGateway出厂默认密码为空，即不使用密码。用户出于安全考虑可以给网关设置密码，用于工程的上传与下载。

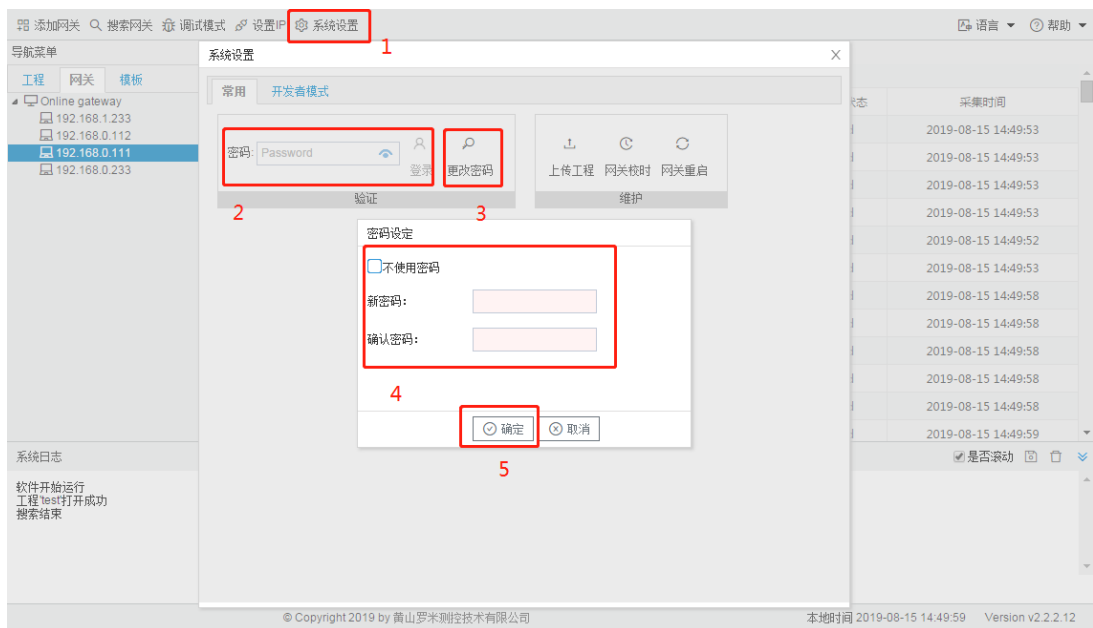


图2-10 密码设定

2.3.6 网关校时

点击"系统设置"框中"网关校时"按钮，弹出"网关校时"窗口。

用户可将LMGateway与时间源同步时间，也可以将LMGateway作为一个NTP服务器，同步给其他设备。

- 网关根据同步周期与NTP服务器同步时间

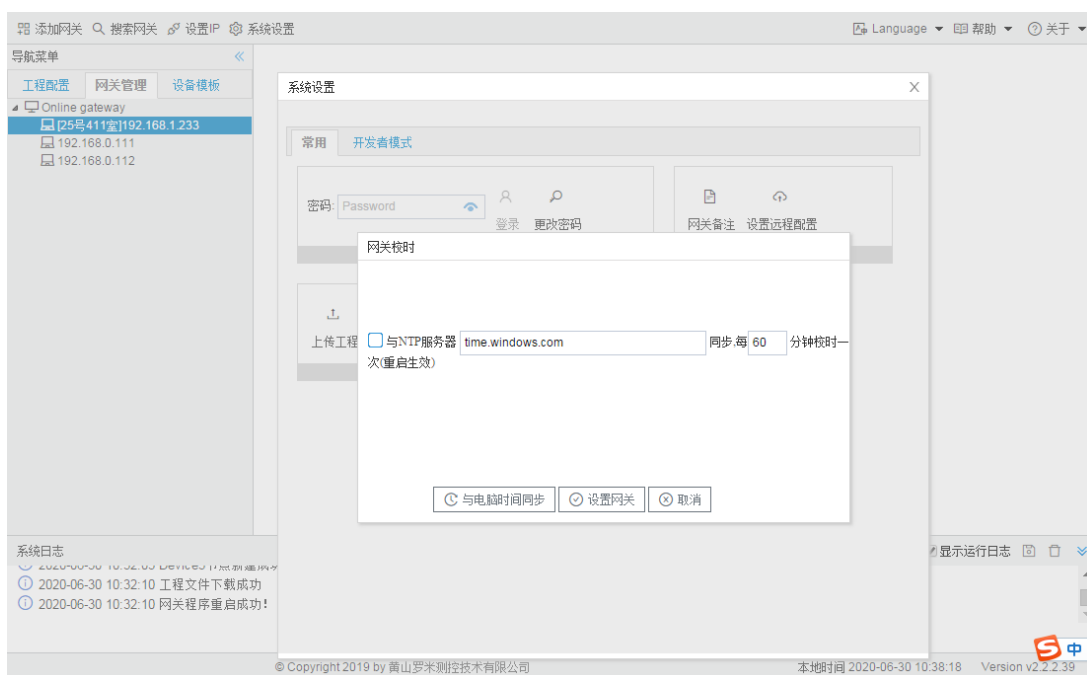


图2-11 网关校时

如果只需要将网关时间与电脑时间同步，只需要点击“与电脑时间同步”按钮；如果需要设置上图中的4，设置好之后需要点击“设置网关”按钮。

2.3.7 重启网关

点击"系统设置"框中"重启网关"按钮，弹出"重启网关"窗口。

GC对网关进行重启。

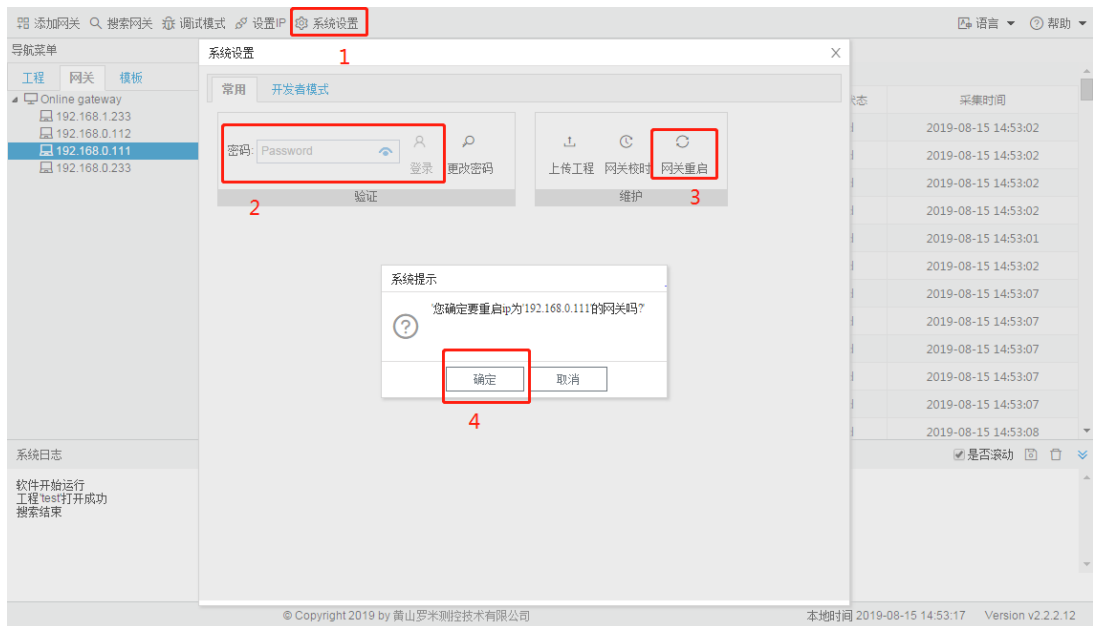


图2-12 重启网关

2.3.8 开发者模式

点击"系统设置"框中"开发者模式"选项卡，输入厂家密码。

建议用户不要随意点击，需要进行网关升级时请先联系厂家。

在升级过程中请不要操作GC和网关，在升级结束时会在系统日志中输出相应的信息。

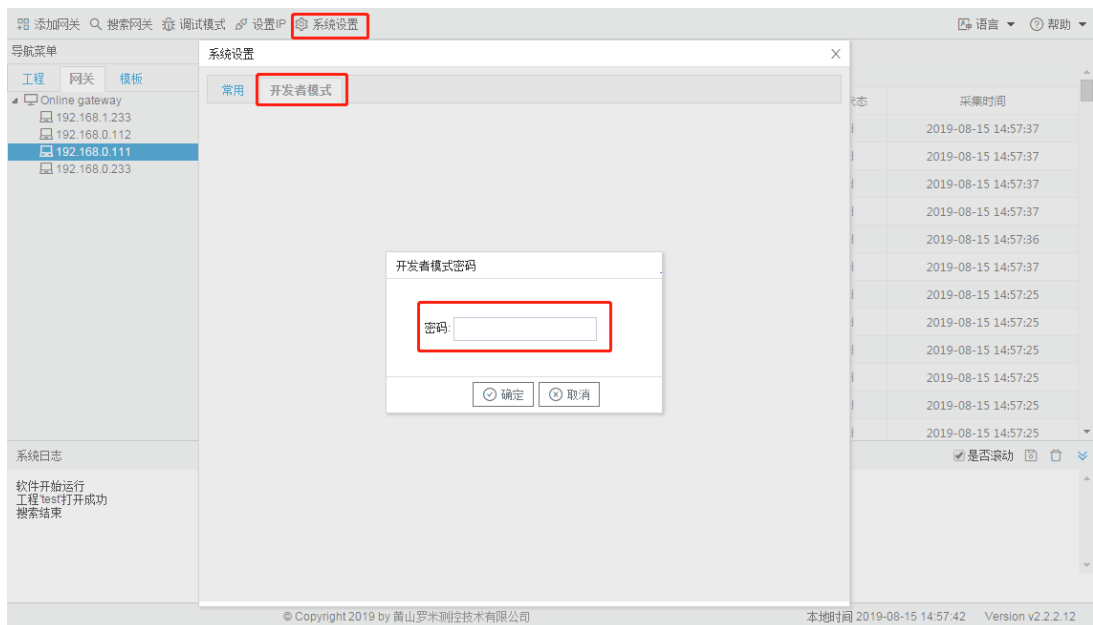


图2-13 网关升级

2.4 调试模式

用户通过点击工具栏中的“调试模式”按钮进入网关调试。

GC的调试模式分为三部分：网关数据采集日志和报文输出、网关Ping测试、网关串口测试。

2.4.1 网关数据采集日志和报文输出

用户可以查看网关运行日志及网关与设备交互的报文。

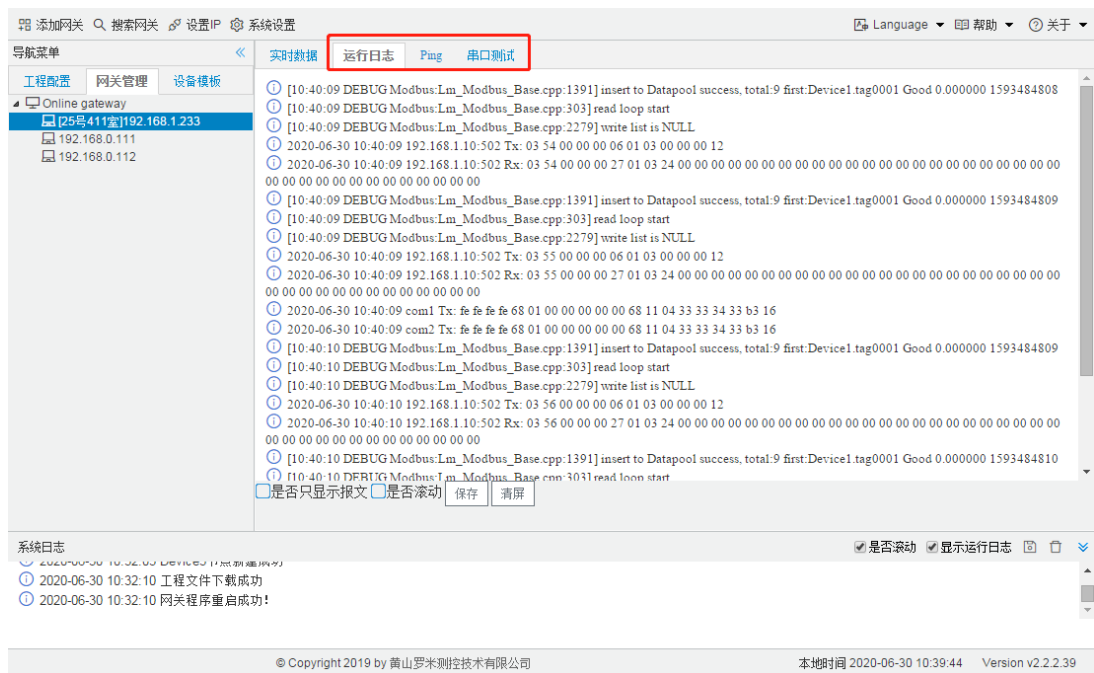


图2-14 网关数据采集日志和报文输出

2.4.2 Ping测试

用户输入目标IP，点击“确定”按钮，网关对目标IP执行ping操作。（默认进行5次ping操作）。

点击“实时数据”选项卡、“运行日志”选项卡或者重启网关恢复正常采集、服务。

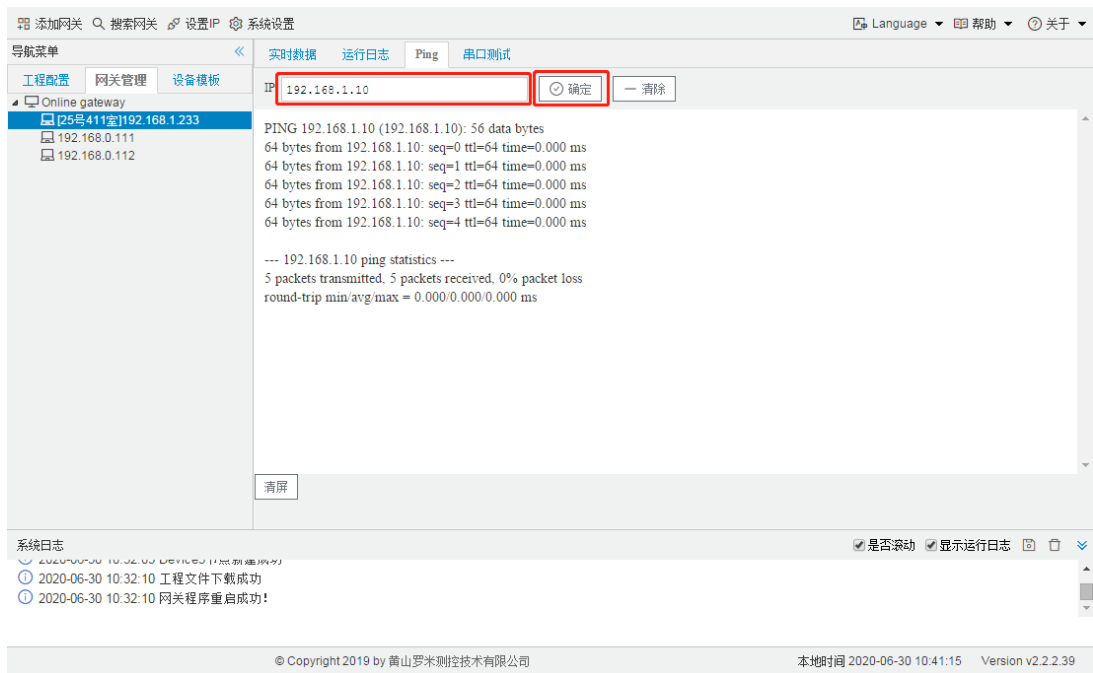


图2-15 Ping测试

2.4.3 串口测试

用户可以通过串口测试，设置网关串口，发送下方文本框中编辑的16进制报文，验证串口参数设置，查看报文收发。

点击“实时数据”选项卡、“运行日志”选项卡或者重启网关恢复正常采集、服务。

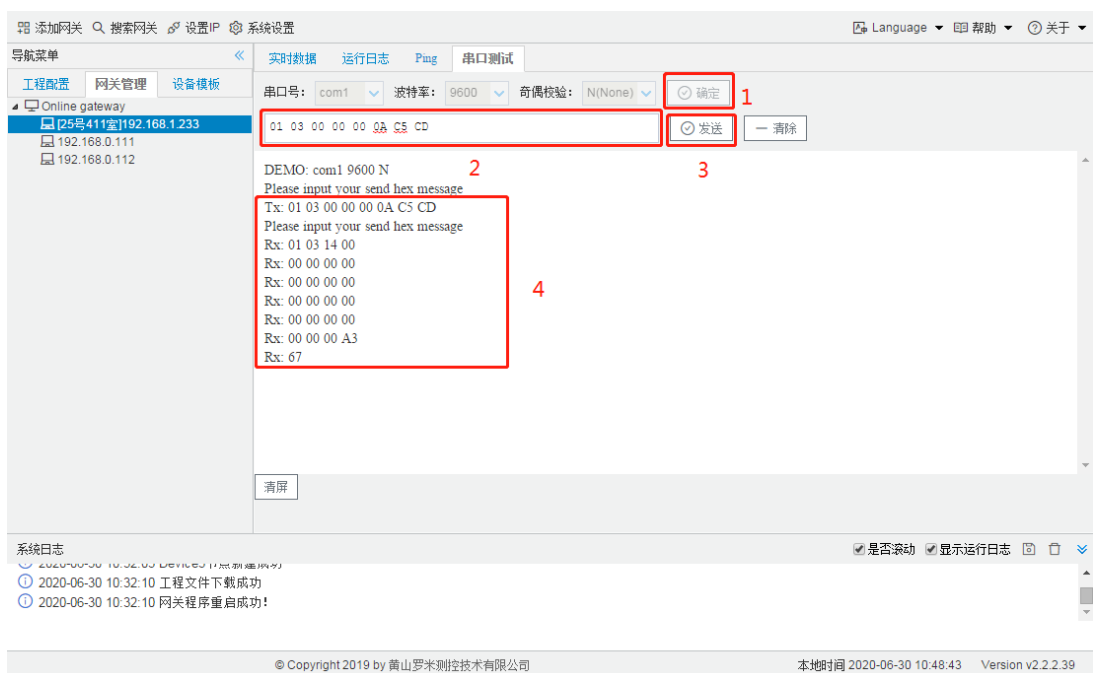


图2-16 串口测试

2.5 HTTP接口

网关开放以下HTTP接口，并且给出范例，具体范例在GC安装目录的Web Demo文件夹中。

2.5.1 实时数据

获取所有点的实时数据

接口功能：

获取此时网关中所有数据点的实时数据

URL（获取实时数据地址）：

<http://192.168.1.233/API/V2/real> (192.168.1.233为需要查询的网关的ip地址)

HTTP请求方式：

HTTP get

请求参数：

无参数

返回结果格式：

JSON

返回结果字段：

返回字段	字段类型	字段返回值	说明
deviceCode	string	ALL	指明是所有tag点的实时数据
val	string		所有tag点的实时数据
id	string		tag点的唯一标识
status	string	Good/Error	tag点的数据质量
timestamp	timestamp	1537007573	tag点采集的时间戳
val	string		tag点的实时值

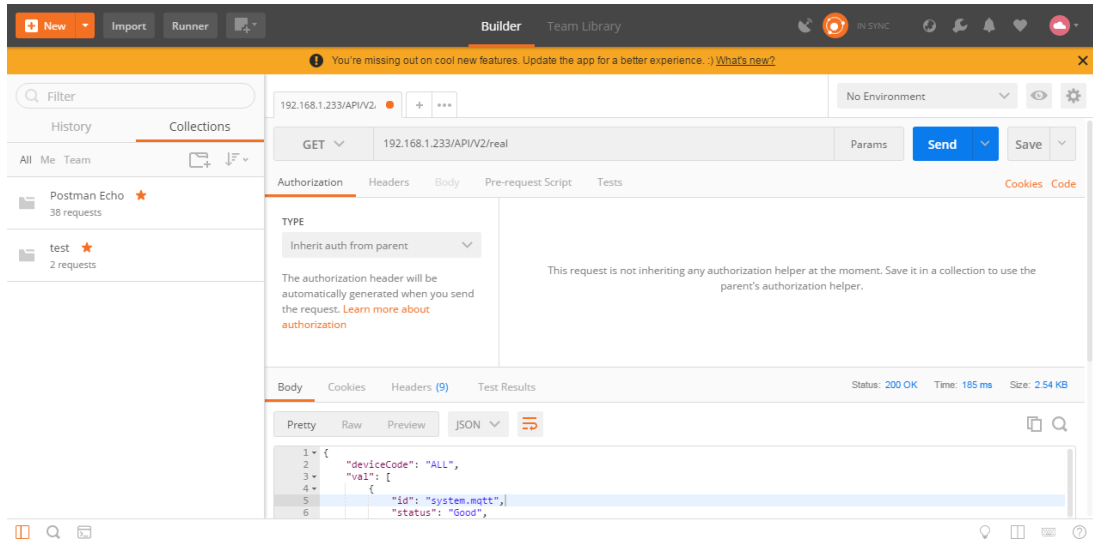
返回结果示例:

```

{
  "deviceCode": "ALL",
  "val": [
    {
      "id": "system.mqtt",
      "status": "Good",
      "timestamp": 1558312746,
      "val": "1"
    },
    {
      "id": "system.TIME_MINUTE",
      "status": "Good",
      "timestamp": 1558312800,
      "val": "40"
    },
    {
      "id": "system.TIME_WDAY",
      "status": "Good",
      "timestamp": 1558312748,
      "val": "1"
    },
    {
      "id": "Device1.tag0001",
      "status": "Error",
      "timestamp": 1558312858,
      "val": "0"
    }
  ]
}

```

Postman截图:



获取单个设备的实时数据

接口功能:

获取此时网关中单个设备的实时数据

URL（获取实时数据地址）:

<http://192.168.1.233/API/V2/real?deviceCode=Device1> (192.168.1.233
为需要查询的网关的ip地址, Device1为需要获取数据的设备名称)

HTTP请求方式:

HTTP get

请求参数:

deviceCode

返回结果格式:

JSON

返回结果字段:

返回字段	字段类型	字段返回值	说明
deviceCode	string		指明是此设备的所有数据
val	string		设备中所有tag点的实时数据
id	string		tag点的唯一标识
status	string	Good/Error	tag点的数据质量
timestamp	timestamp	1537007573	tag点采集的时间戳
val	string		tag点的实时值

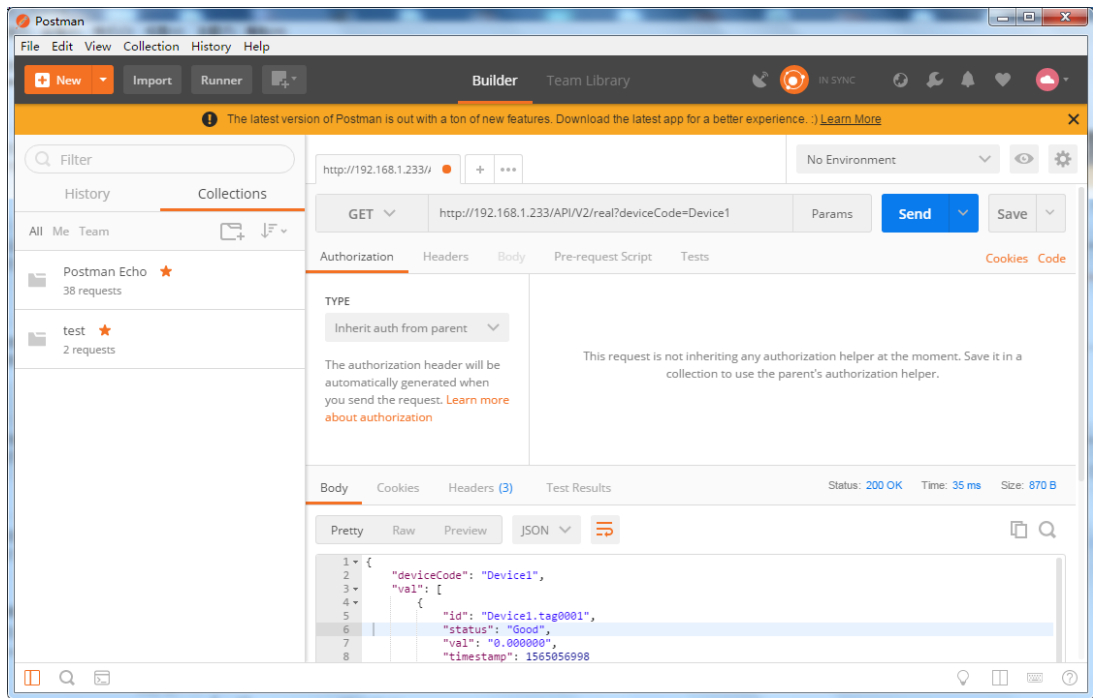
返回结果示例:

```

{
  "deviceCode": "Device1",
  "val": [
    {
      "id": "Device1.tag0001",
      "status": "Error",
      "timestamp": 1558312858,
      "val": "0"
    },
    {
      "id": "Device1.tag0002",
      "status": "Error",
      "timestamp": 1558312858,
      "val": "0"
    },
    {
      "id": "Device1.tag0003",
      "status": "Error",
      "timestamp": 1558312858,
      "val": "0"
    }
  ]
}

```

Postman截图:



2.5.2 数据设定

接口功能:

进行单个采集点的写操作

URL（进行写操作地址）:

<http://192.168.1.233/ctrlRequest> (192.168.1.233为需要进行写操作的网
关的ip地址)

HTTP请求方式:

HTTP post, form-data

请求参数:

字段名称	字段类型	必填	说明
id	string	Device1.tag0001	指明进行写操作的采集点
val	string	10	需要写入的值

返回结果格式:

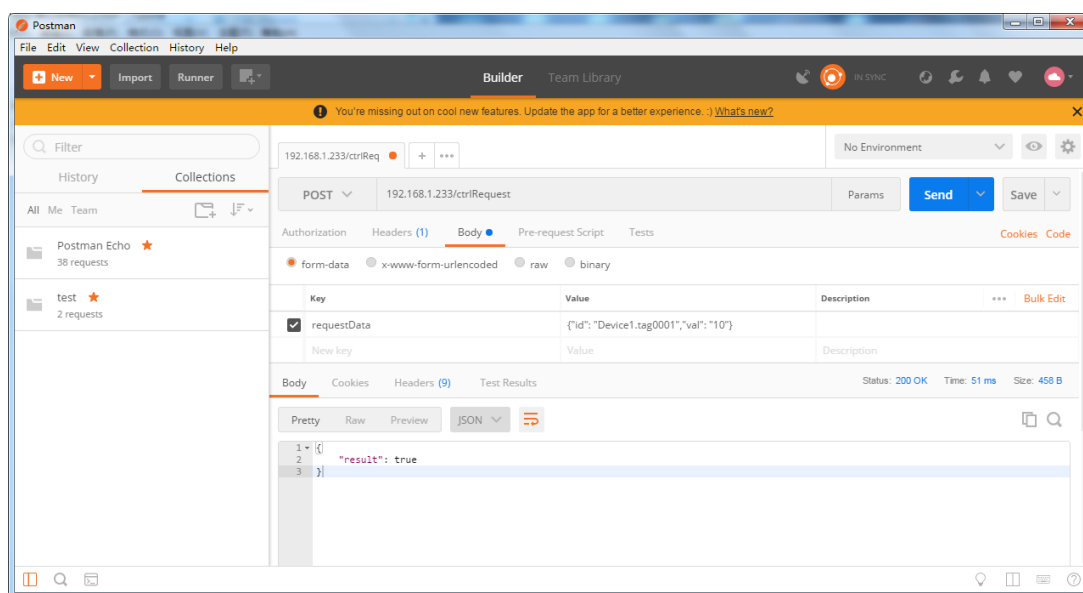
json

返回结果:

```
{
  "result": true
}
```

HTTP通讯成功返回结果都为true，指明网关接收到发送的写操作的数据。

Postman截图:



2.5.3 历史数据

接口功能:

获取指定日期、指定Tag点的历史数据

URL（获取历史数据地址）:

<http://192.168.1.233/API/V1/history?startTime=2019-07-09&tagID=Device1.tag0001> (192.168.1.233为需要获取历史数据的网关的ip地址，startTime为需要获取历史数据的指定日期，tagID为需要获取历史数据的指定Tag点)

HTTP请求方式:

HTTP get

请求参数:

startTime

tagID

返回结果格式:

json

返回结果:

```
[
  {
    "tagid": "Device1.tag0001",
    "val": "1.000000",
    "status": "Good",
    "timestamp": 1562638196
  },
  {
    "tagid": "Device1.tag0001",
    "val": "1.000000",
    "status": "Good",
    "timestamp": 1562638224
  },
  {
    "tagid": "Device1.tag0001",
    "val": "1.000000",
    "status": "Good",
    "timestamp": 1562638253
  }
]
```

Postman截图:

The screenshot displays the Postman Builder interface. At the top, there's a navigation bar with 'New', 'Import', and 'Runner' buttons, and a 'Builder' title. A notification banner at the top center reads: "You're missing out on cool new features. Update the app for a better experience. [What's new?](#)".

The main interface is divided into several sections:

- Left Panel (Collections):** Shows a search filter, 'History', and 'Collections' tabs. Under 'Collections', there are two collections: 'Postman Echo' (38 requests) and 'test' (4 requests). Below these, a list of requests is visible, including a GET request to 'http://192.168.1.10/get_real_db' and another GET request to 'http://192.168.1.10/API/V1/history'.
- Request Configuration:** The selected request is a GET request to 'http://192.168.1.233/API/V1/history?startTime=2019-07-09&tagID=Device1.tag0001'. It includes a 'Send' button and a 'Save' dropdown.
- Authorization:** The 'Authorization' tab is active, showing 'TYPE' set to 'Inherit auth from parent'. A note states: "The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)".
- Response:** The 'Body' tab is active, showing the response in 'JSON' format. The response is a JSON object:

```
{  "tagid": "Device1.tag0001",  "val": "1.000000",  "status": "Good",  "timestamp": 1562638196}
```

2.6 WEB服务器

网关自带 WEB 服务器，端口固定为 80。用户可以通过浏览器就登录到 WEB 页面，在网页里可以修改查看实时数据、历史存储、事件记录和重启网关。
(推荐使用谷歌浏览器)

2.6.1 网页登录

在浏览器中输入网关的 IP 地址，如图 2-17 所示。

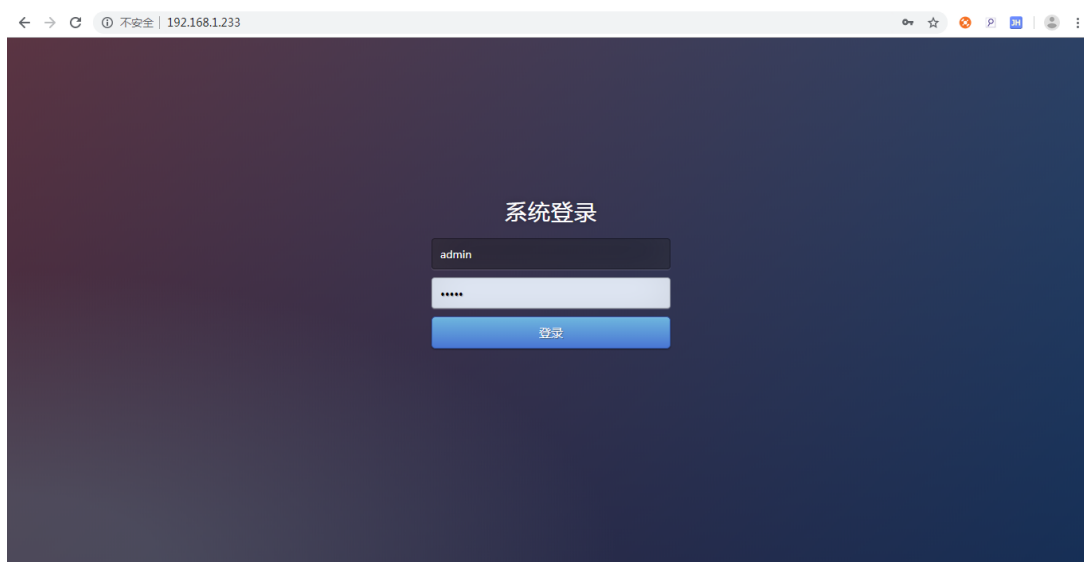


图2-17 网页登录

密码初始默认为luomi，可通过配置工具中“更改密码”按钮进行修改。

2.6.2 实时数据

在浏览器网页登录成功之后，会自动显示当前网关的实时数据。如图2-18所示。

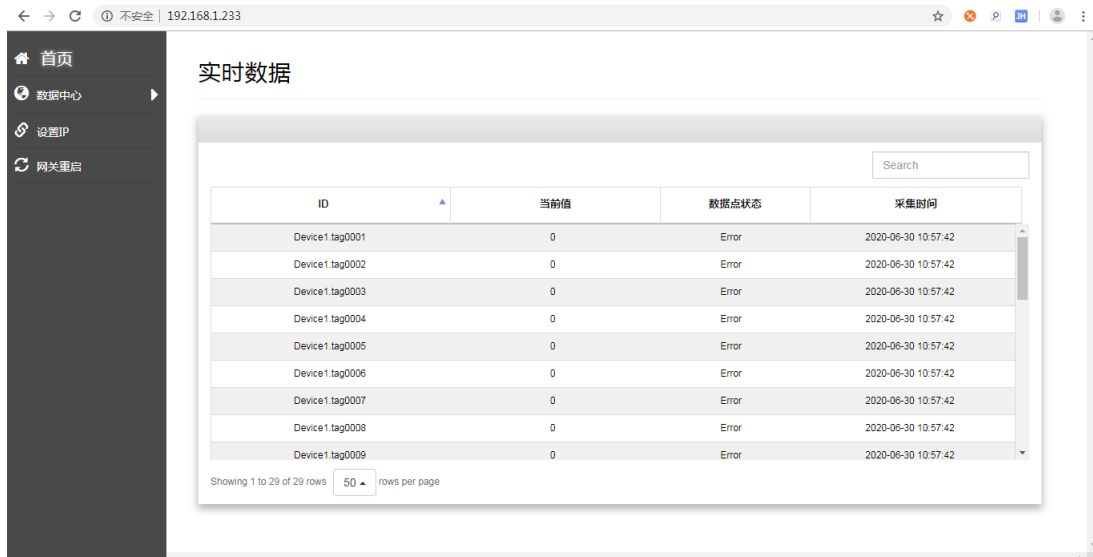


图2-18 实时数据

2.6.3 历史数据

用户可在历史数据页面每次查询一天的历史信息。在查询时需要当前网关开启了历史存储的功能。

用户先选择需要查询的日期和存储点，点击“查询”按钮。

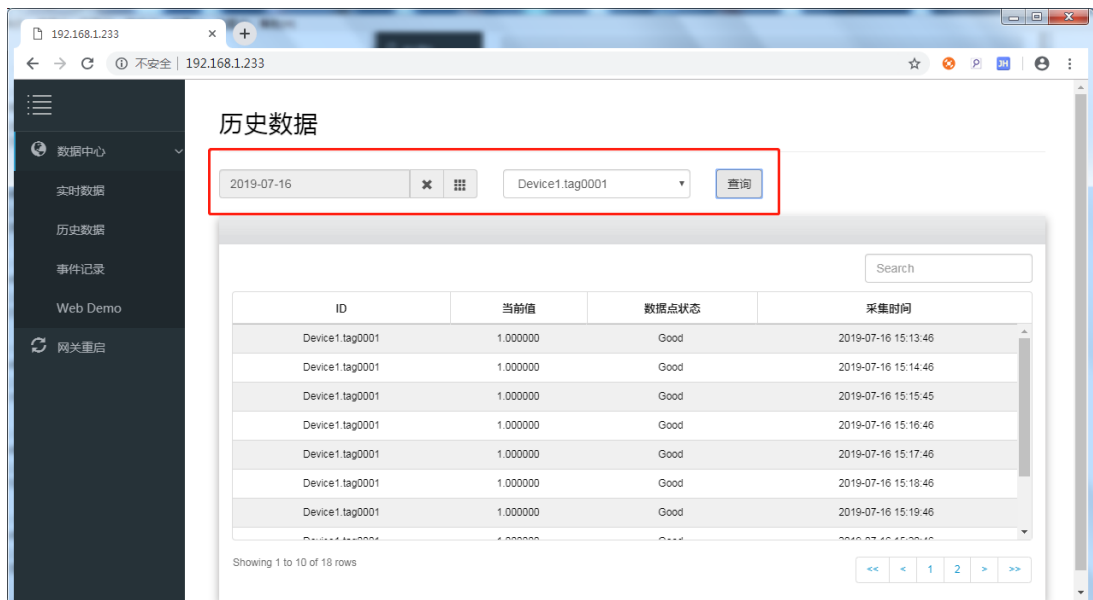


图2-19 历史数据

2.6.4 事件记录

用户可在事件记录页面查询事件信息。

用户选择需要查询的时间段，点击“查询”按钮。

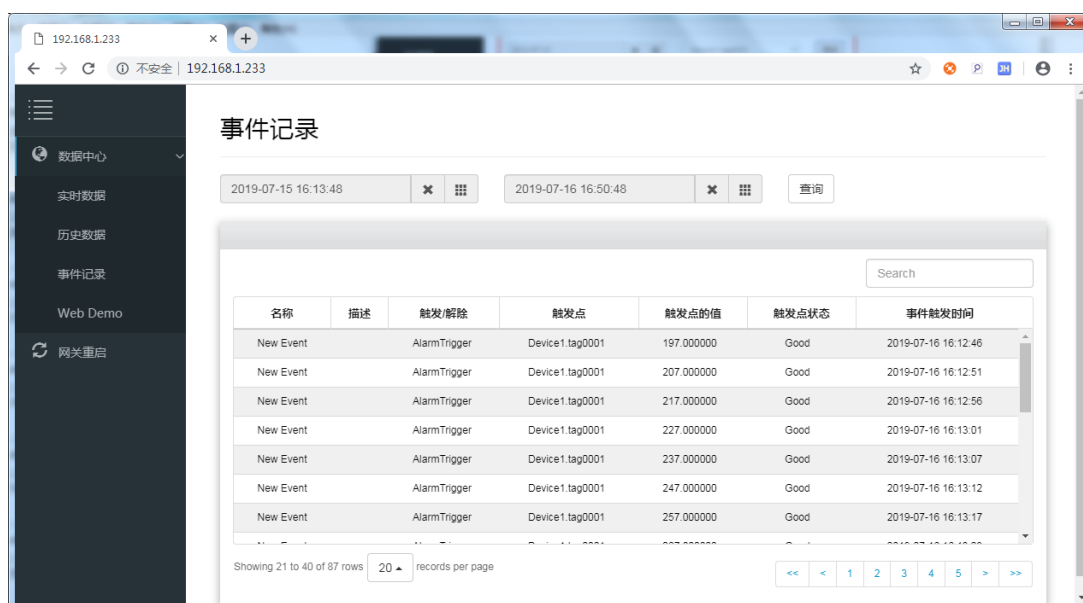


图2-20 事件记录

2.6.5 设置IP

用户可通过web页面设置网关IP。

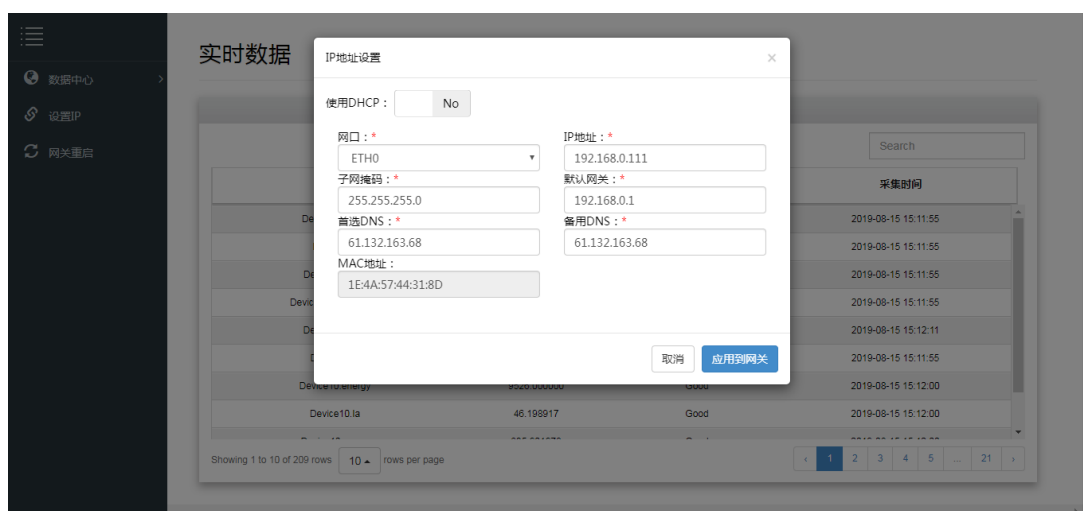


图2-21 设置IP

2.6.6 网关重启

点击左侧“网关重启”菜单，在弹出的对话框中点击“确定”按钮，即可对当前网关进行重启操作。

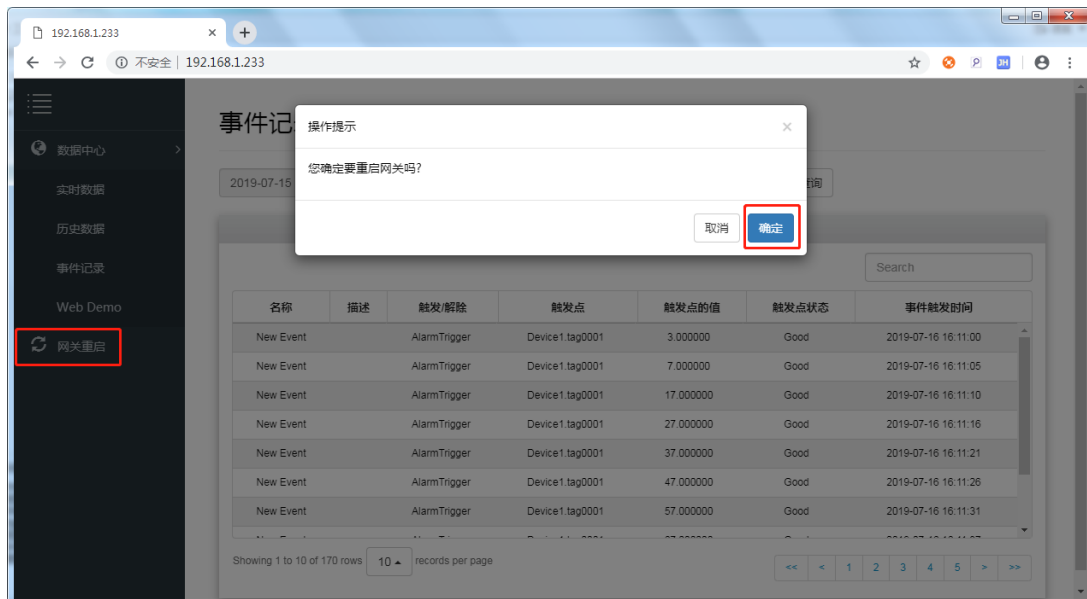


图2-22 网关重启

第三章 数据采集配置

数据采集分为I/O点和内部点，IO点为驱动采集点，内部点分为：系统点、计算点和用户点。I/O点需要针对每种采集需求完成通道、设备、Tag点的配置。内部点只具备工程意义。

各个驱动通道、设备、Tag点配置详见驱动通讯文档。

3.1 通道配置及协议选择

LMGateway网关支持串口、网口采集，可新增或修改通道参数。

3.1.1 新建通道

单击选中“I/O点”节点，右键单击选择“新建通道”，编辑通道参数，在此我们选择“ModbusTCPClient”协议作为范例进行演示，如图2-1所示。

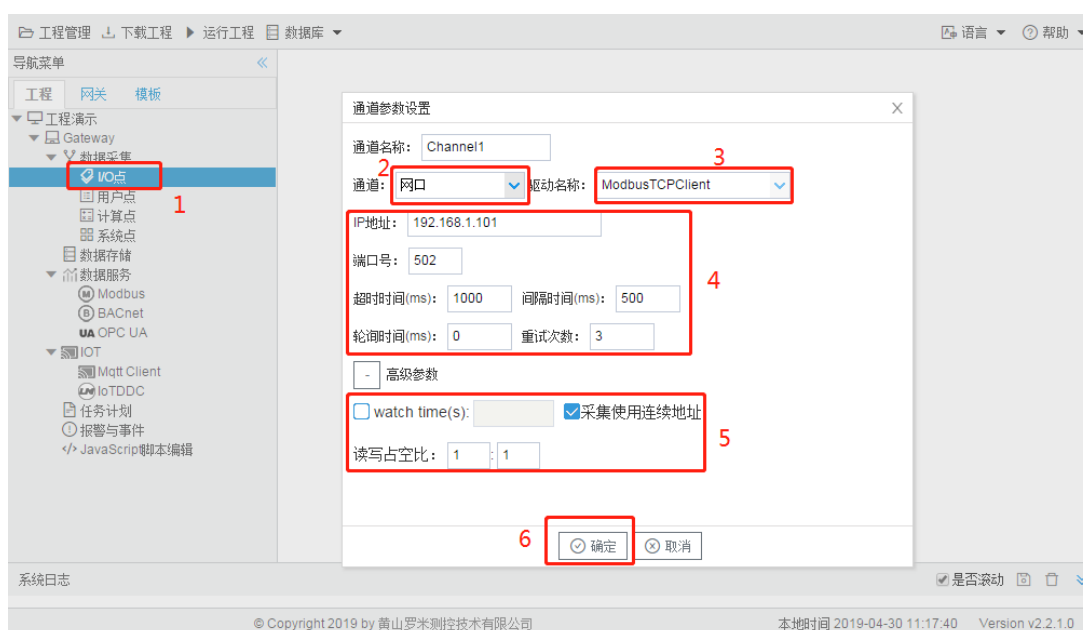


图3-1 新建通道

在通道参数设备的弹出框中，进行从上往下的顺序配置，根据上图的步骤进行配置：

1. 编辑通道名称；
2. 选择驱动通道是串口、网口；
3. 选择驱动名称；
4. 配置驱动的基本参数；
5. 配置驱动的高级参数；
6. 点击“确定”完成配置。

在每个协议的基本参数中都会包含超时时间、间隔时间、轮询时间和重试次数四项，含义如下：

- 超时时间：定义每一串报文发送后，等待被采集设备返回的时间。当通信正常的情况下，设置长一点不影响通信速度；假如设备响应速度比较慢，为了避免通信失败，建议设置长一点。
- 间隔时间：可自定义，网关接收到被采集设备返回的报文后等待间隔时间之后再发送下一组报文。
- 轮询时间：当前通道中完成所有采集任务后，等待轮询时间，再进入下一次采集周期。
- 重试次数：通讯失败后重新发送当前报文的次数。

通道配置界面根据协议的不同会判断是否显示高级参数的按键。每种协议的高级参数也有所不同。

3.1.2 修改通道

双击需要修改的通道，进入编辑窗口，如图2-2所示。

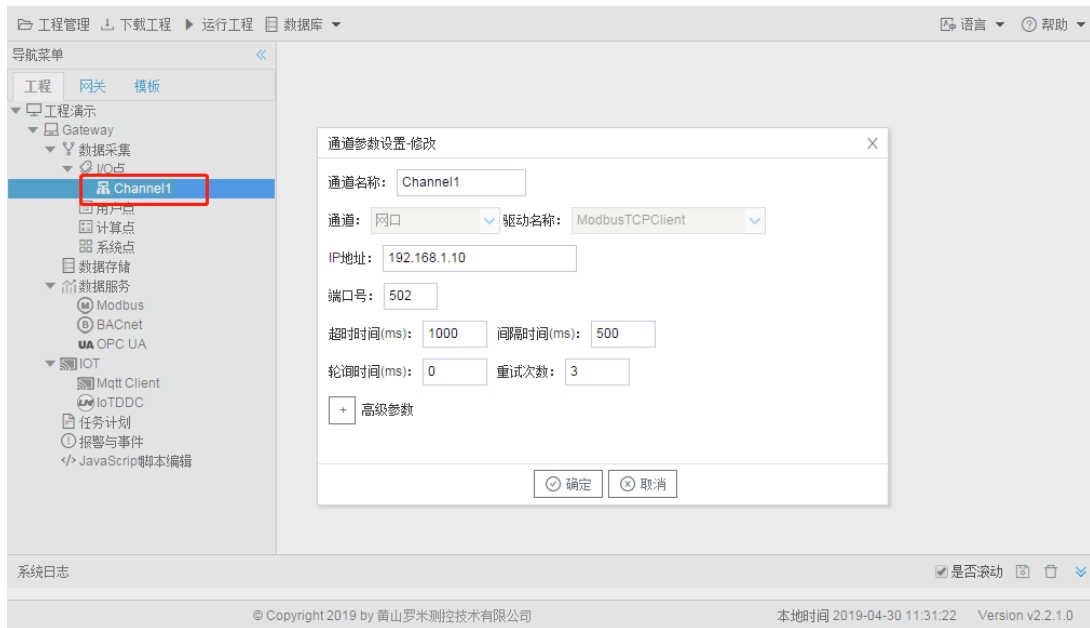


图3-2 修改通道

3.2 设备及点表编辑

3.2.1 新建设备

单击选中需要添加设备的通道，右键选择“新建设备”，如图2-3所示。

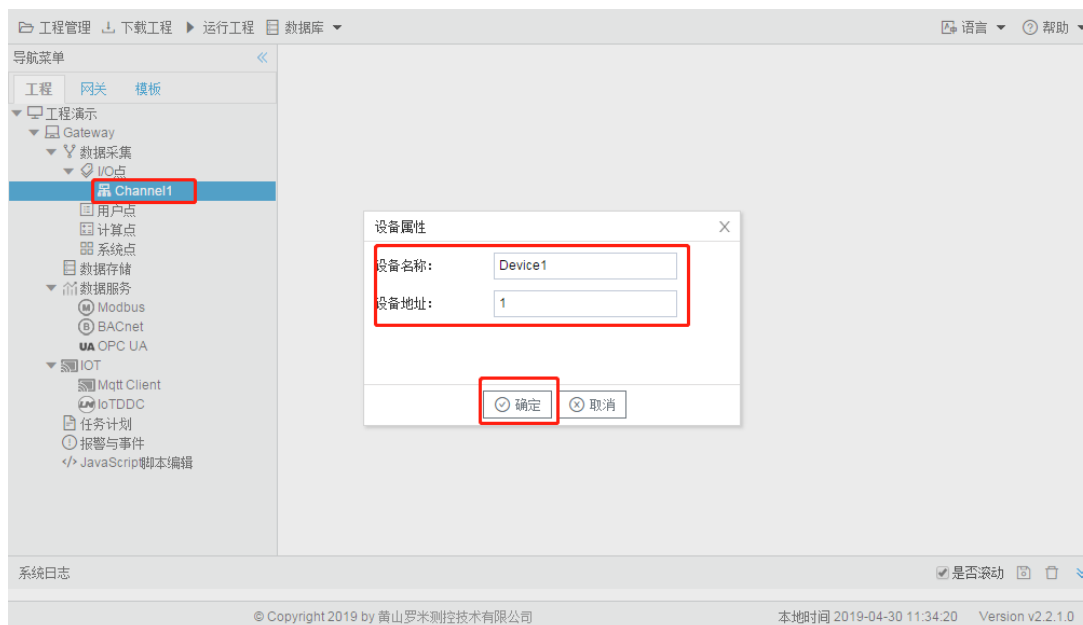


图3-3 新建设备

编辑设备属性，点击确定生效。

- 设备名称：自定义，设备在该通道下的唯一标记，不可重复；
- 设备地址：设备的通讯地址，如Modbus RTU从站设备的站号（slave ID），645电能表的表号，BACnet的设备ID等；

3.2.2 批量复制新增

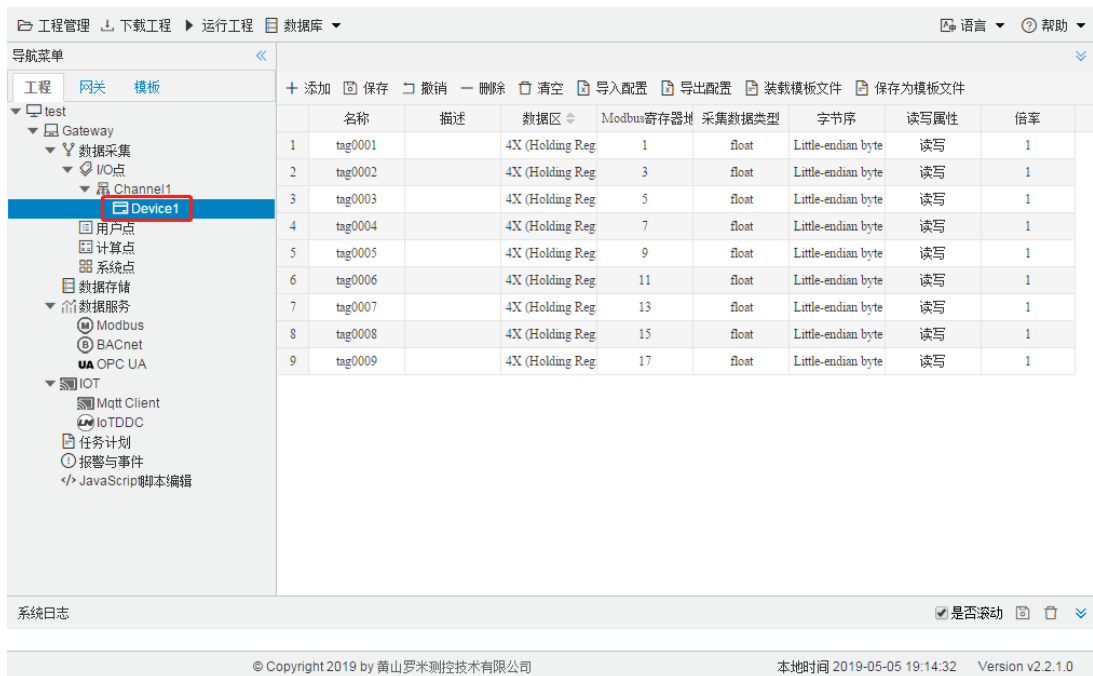
同一个通道下的设备具有相同的协议属性，可以选中已编辑好的设备，右键“批量复制新增”，在弹出框中点击“+”按钮，编辑新设备的设备名称，设备地址。这个操作可以快速编辑完成一个通道下的若干设备。如图3-4所示。



图3-4 批量复制新增

3.2.3 编辑点表

单击左侧工程树中的设备，进入设备采集点表编辑页面。如下图所示。



modbusTag点示例

3.3 用户点

用户点属于可读可写的内部点。

可用于JavaScript编程、作为一个控制信号等。

1. 单击工程树上的“用户点”节点。
2. 点击“添加”按钮新增一个用户点。
3. 输入配置页面中唯一的名称，选择点类型（模拟量和离散量），添加该点的默认值。
4. 点击“保存”完成添加点。

具体步骤如图3-5所示。

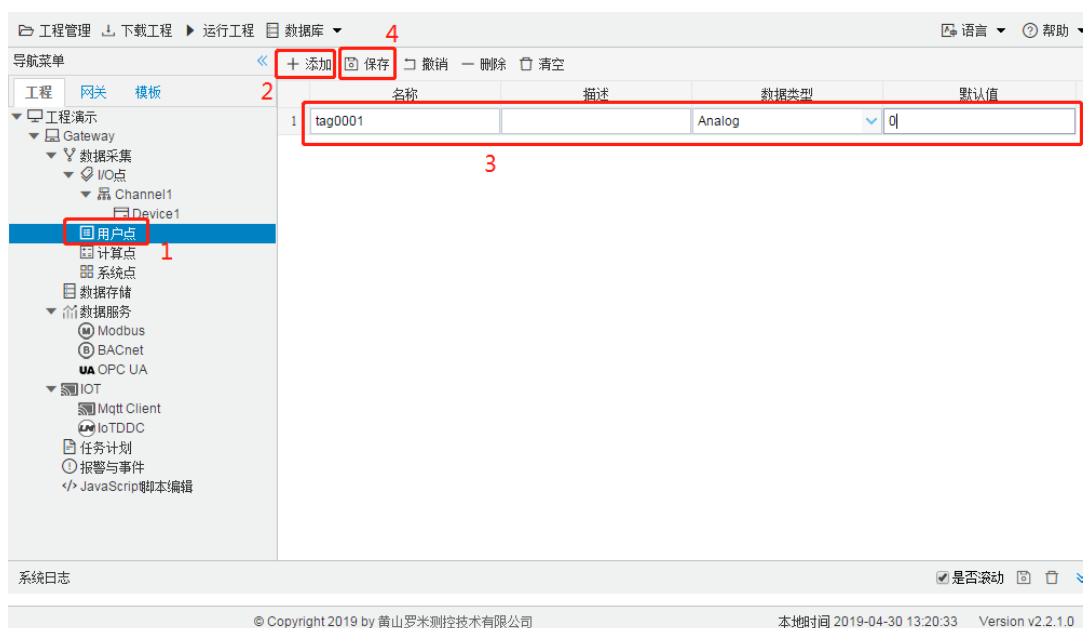


图3-5 添加用户点

表格中各字段说明如下：

- 名称：必填项，可修改。在此页面中不能重复，为该点的唯一标识，如上图名称为tag0001，在工程中该点的标识为user.tag0001
- 描述：选填项，该点的描述信息
- 数据类型：分为Analog(模拟量)，Discrete(离散量)和String(字符串)
- 默认值：工程运行时该点的默认值，当数据类型为Analog时可填任意数，当数据类型为Discrete时只能填0或1

3.4 计算点

计算点属于只读的内部点，它的值是某个表达式的计算结果。

该表达式的参数可以是Tag点或常数，在表达式中可以使用一些常用的计算方法，包括四则运算，逻辑运算，三角函数等。

通过使用计算点，可以做一些相对复杂的计算，例如将采集到的传感器数值通过换算得到实际的物理量，这样可以减少上位机的运算量，使得设备更加智能化。

每个计算点对应一个表达式，表达式可以有最多4个Tag点作为输入变量，为了方便起见，4个Tag点在表达式中分别使用A, B, C, D来表示。操作步骤如下：

1. 单击工程树上的“计算点”节点。
2. 点击“添加”按钮新增一个计算点。
3. 输入配置页面中唯一的名称，如下图名称为tag0001，在工程中该点的标识为calculate.tag0001。
4. 输入计算表达式，表达式中可使用的预设函数或运算符可以从表达式输入框上方的几个下拉框中选择，也可以手动输入预设的函数或运算符。
5. 单击表达式变量的输入框左侧添加按钮，选择该变量对应的Tag点。
6. 点击“确定”完成添加点。

具体步骤如图3-6所示。

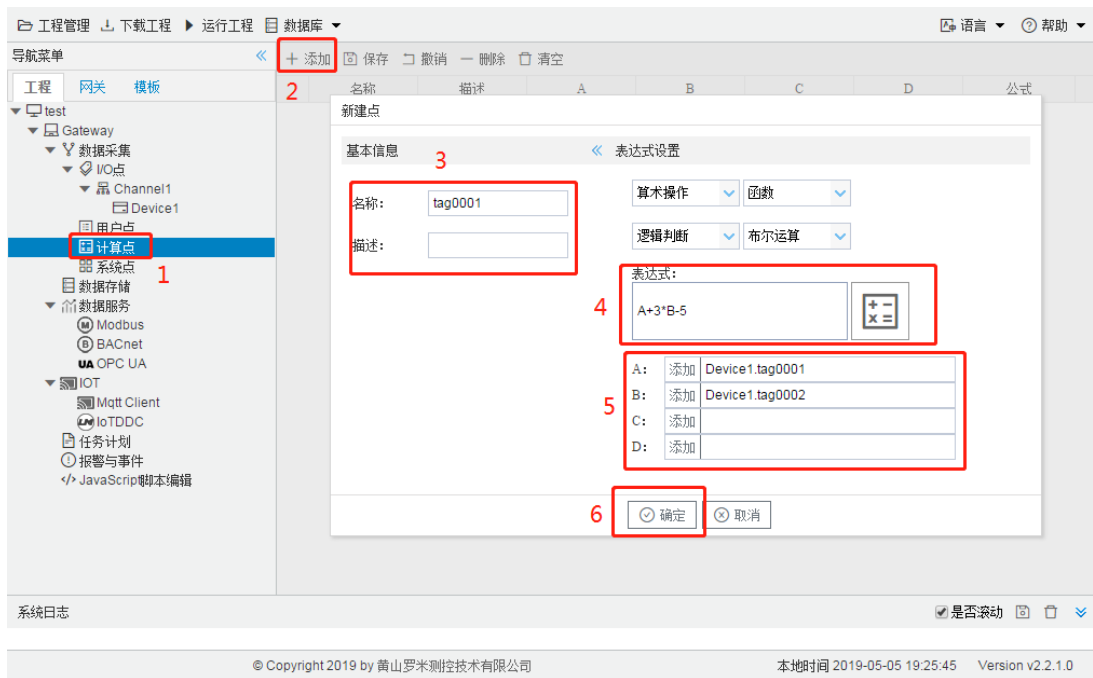




图3-6 添加计算点

表达式验证

界面中有一个计算器样式的按钮 ，点击它可以打开表达式检查器，如下图所示。表达式检查器中A、B、C、D均已赋值。点击计算按钮  得到计算结果，直观地验证表达式是否正确。

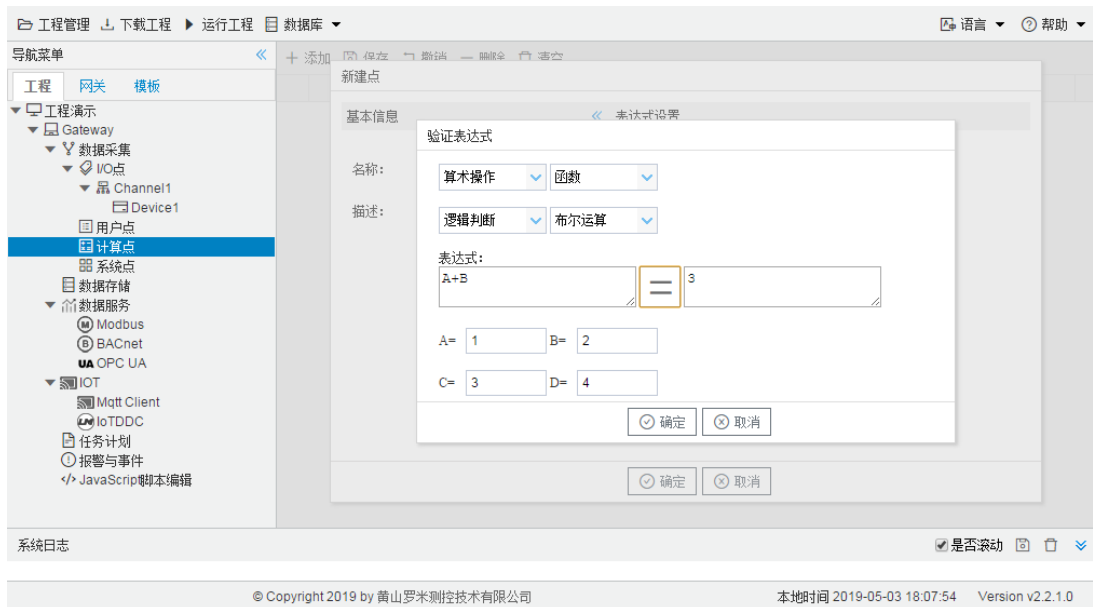


图3-7 表达式检查

3.5 系统点

系统点是只读的内部点，提供了网关的时间信息、网关与设备的通讯状态。

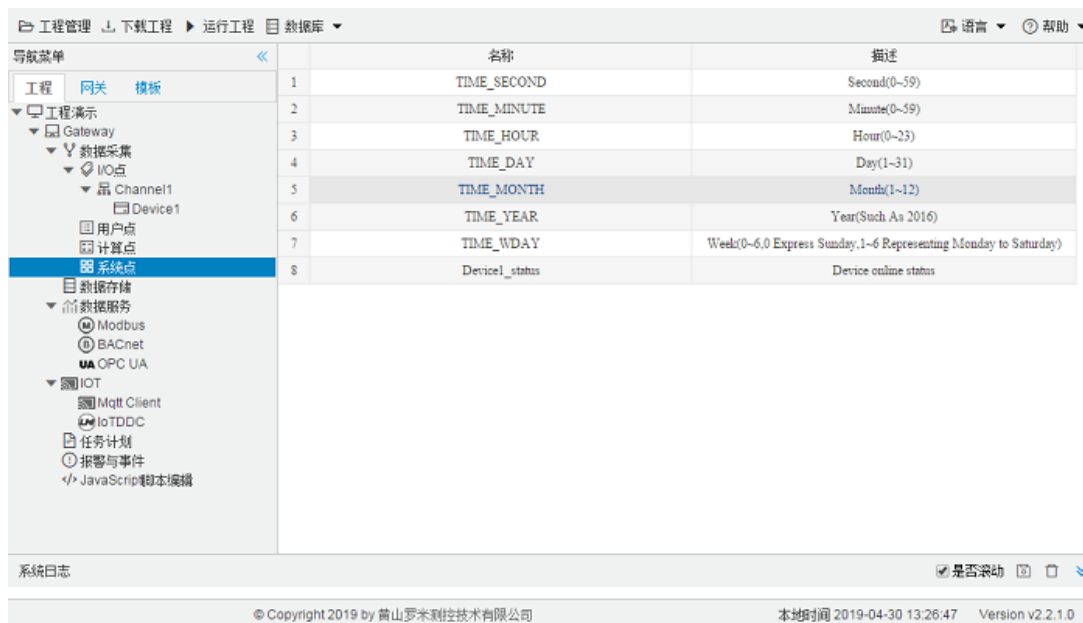


图3-8 系统点

默认拥有的系统点：

TIME_SECOND：系统时间中的秒值，在工程中该点的标识为 system.TIME_SECOND

TIME_MINUTE：系统时间中的分值，在工程中该点的标识为 system.TIME_MINUTE

TIME_HOUR：系统时间中的小时值，在工程中该点的标识为 system.TIME_HOUR

TIME_DAY：系统时间中几号，在工程中该点的标识为 system.TIME_DAY

TIME_MONTH：系统时间中几月，在工程中该点的标识为 system.TIME_MONTH

TIME_YEAR：系统时间中的年，在工程中该点的标识为 system.TIME_YEAR

TIME_WDAY：系统时间中星期几，在工程中该点的标识为 system.TIME_WDAY

设备状态点:

Device1_status: 在I/O点中新建设备后，系统点页面会自动添加该设备的状态点。其中“Device1”为设备名称，该设备下的所有数据点中有任意一个采集成功，则该设备的状态点值为1；该设备下的所有数据点均采集失败，则该设备的状态点值为0。

第四章 数据存储

LMGateway网关具备数据存储功能，这种功能可以实现I/O点、用户点、计算点和系统点的数据存储，网关的数据只能存储到TF卡当中。（TF卡仅支持FAT32格式）

数据存储会在TF卡中新建一个名为history的文件夹，每隔一天生成一个新的数据库文件，当存储空间不足时，会将最早的一个数据库文件删除。

数据库文件可通过GC上传至用户指定目录，也可直接取出TF卡获取数据库文件（建议：网关断电后进行插拔TF卡操作）。

该数据库文件可以直接通过sqlite工具打开。

4.1 数据存储配置

开启数据存储功能，需要在数据存储页面进行相关的配置，操作步骤如下：

1. 单击“数据存储”节点，打开数据存储的配置页面；
2. 勾选“启用”复选框；
3. 选择存储模式：周期存储（需要设置存储周期）和准点存储（需要添加每天存储的整点时间）；
4. 点击“保存”完成配置。

如图4-1所示。

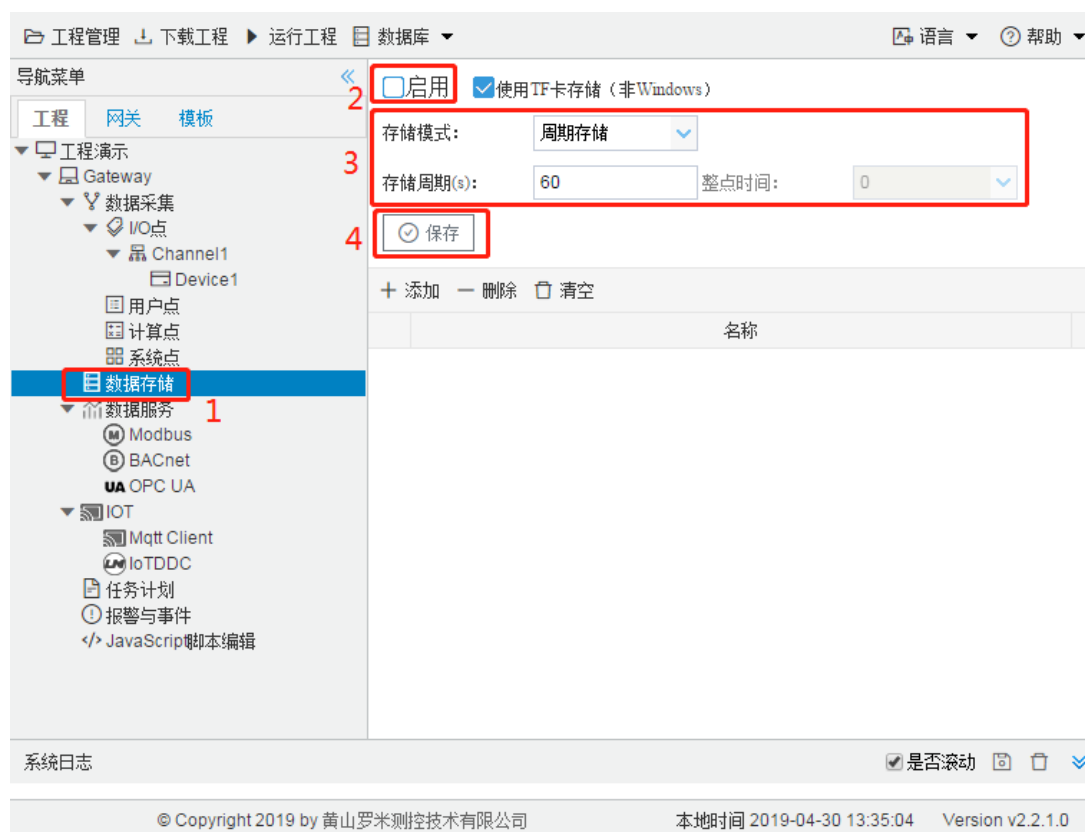


图4-1 数据存储配置

LMGateway网关的数据存储只能存储到TF卡当中。

GC运行工程时数据库文件存储在安装目录的history文件夹。

4.2 添加存储点

在数据存储功能中，用户可以自行添加需要存储的点，具体步骤如下：

1. 单击“添加”按钮；
2. 在弹出窗口中勾选需要存储的点；
3. 点击“确定”按钮完成存储点的添加。

如图4-2所示。



图4-2 添加存储点

第五章 数据服务

LMGateway为SCADA、BA等自控系统提供Modbus、BACnet、OPC UA、OPC DA数据服务。

5.1 Modbus

Modbus服务器实现了Tag点到Modbus寄存器的映射，允许支持Modbus Client的上位机通过Modbus TCP或者Modbus RTU的协议读写Tag点。

Modbus TCP和Modbus RTU通用参数如下：

- slaveID：网关Modbus服务的从站ID；
- 网关通讯异常处理：Tag点出错时，映射的Modbus寄存器点值会发生相应的改变。

“网关通讯异常处理”下拉框中有：

使用默认值：相应的Modbus地址的点值置成表格中“默认值”列的值

使用最后正常值：相应的Modbus地址的点值最后一次采集正常的值

- 服务延迟启动时间：Modbus服务延时多少秒启动。

Modbus TCP配置

使能Modbus TCP服务，允许上位机通过ModbusTCP协议经由网口连接访问网关。

端口号：设置Modbus TCP侦听端口号，默认值是502。

slaveID: 1

Modbus TCP 端口号: 502

Modbus RTU

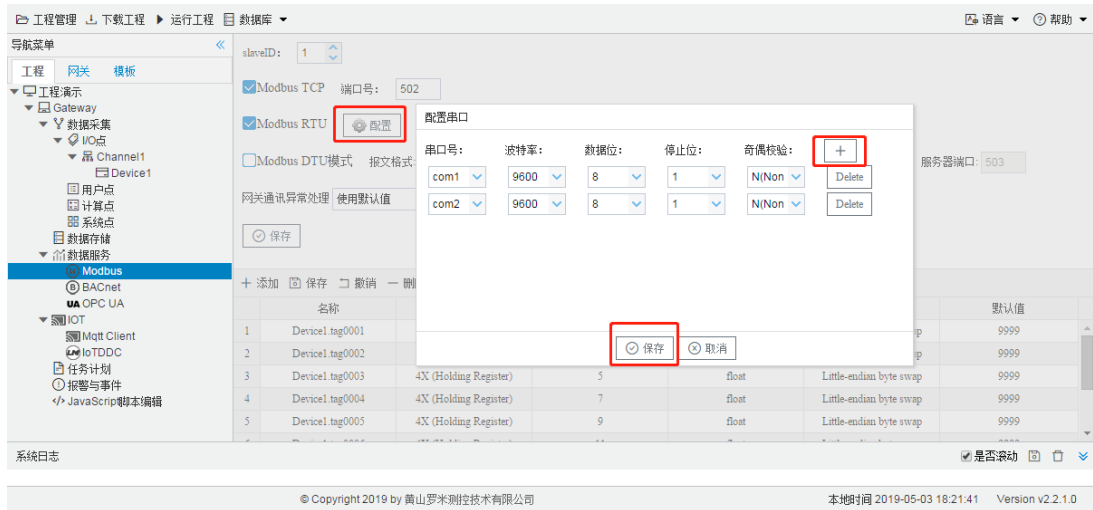
Modbus DTU模式 报文格式: RTU 序号: 123 心跳周期(s): 10 服务器地址: 192.168.1.10

服务器端口: 503

网关通讯异常处理: 使用最后正常值 服务延迟启动时间(s): 0

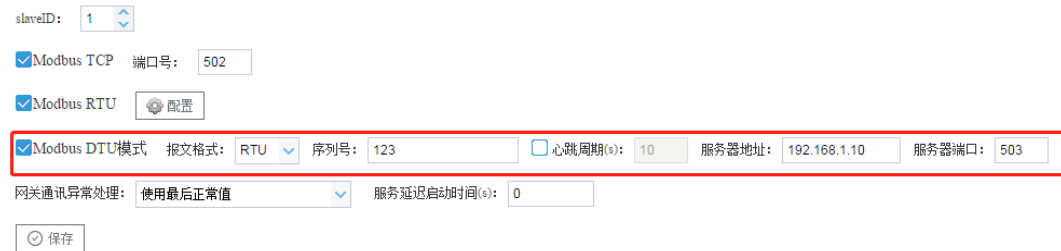
Modbus RTU配置

使能Modbus RTU服务，点击“配置”按钮，在弹出对话框中添加指定串口，提供Modbus RTU服务。允许上位机通过Modbus RTU协议经由串口连接（RS-232/485）来访问网关。



Modbus DTU模式配置

使能Modbus DTU模式，网关上电时，通过填写的服务器地址和端口连接服务器，服务器通过心跳包辨识网关，获取网关公网IP。服务器在该TCP链路上通过modbus TCP或modbus RTU格式报文读写网关Tag点。



报文格式：指定以modbus TCP或modbus RTU格式的报文进行交互。

心跳包格式：序列号文本框中填写的文本。

心跳周期：勾选则网关周期上传心跳包，不勾选则网关仅在上电时发送一次心跳包。

Modbus地址映射

将Tag点映射到Modbus寄存器上，配置步骤如下：

1. 单击“添加”按钮；
2. 在弹出窗口中勾选需要映射的点；
3. 选择modbus数据区、数据类型和字节序；
4. 点击“确定”按钮完成映射点的添加。

重复上述操作可添加更多的点到地址列表。

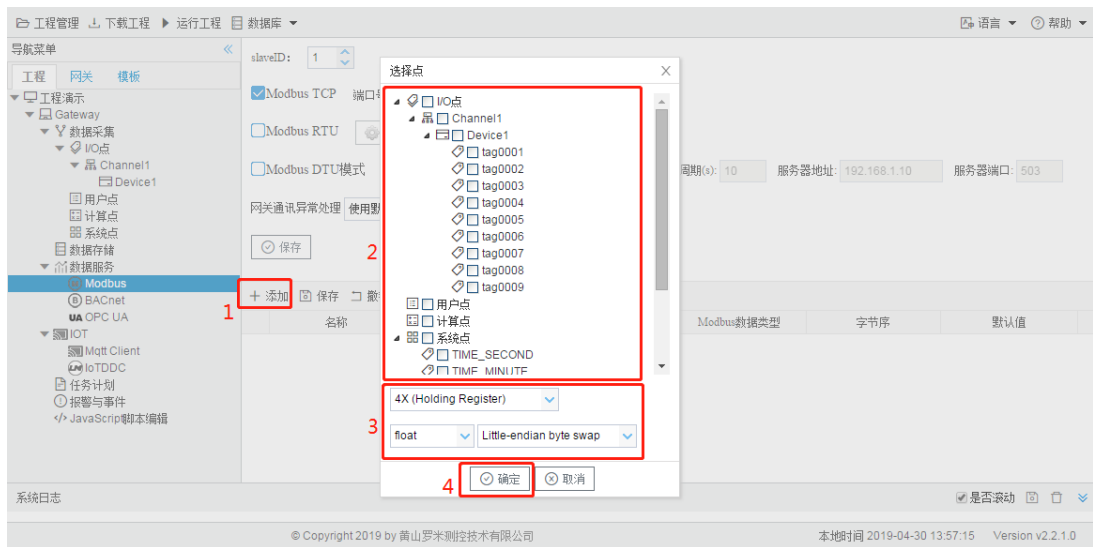
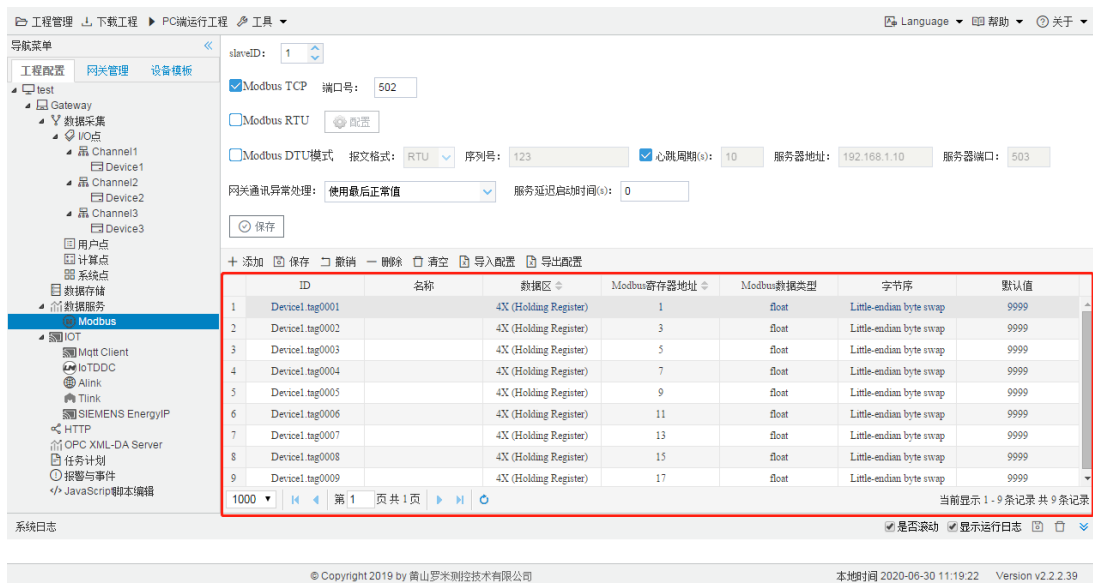


图5-1 Modbus地址映射



Modbus服务点表

双击Tag点可修改映射到Modbus寄存器的详细设定，可配置项有：

- 数据区：有0X (Coil Status), 1X (Input Status), 3X (Input Registers), 4X (Holding Register)四个数据区。
- Modbus寄存器地址：指定该Tag点在Modbus地址空间的起始地址，最小的地址为1。Modbus寄存器地址和数据区配合使用，如数据区选择4X (Holding Register)，Modbus寄存器地址填写1，这该点的Modbus地址为40001。
- Modbus数据类型：有bool、bit、uint16、int16、uint32、int32、float、double、uint64、int64共10种。

- 字节序：有Null、Big-endian、Little-endian、Big-endian byte swap、Little-endian byte swap共5种。
 - Null：用于bool、bit、uint16、int16四种数据类型，表示无字节序；
- **Big-Endian**：大端模式，是指数据的低位（就是权值较小的后面那几位）保存在内存的高地址中，而数据的高位，保存在内存的低地址中，这样的存储模式有点儿类似于把数据当作字符串顺序处理：地址由小向大增加，而数据从高位往低位放；有些软件中描述成 **4 3 2 1**
 - **Little-Endian**：小端模式，是指数据的低位保存在内存的低地址中，而数据的高位保存在内存的高地址中，这种存储模式将地址的高低和数据位权有效地结合起来，高地址部分权值高，低地址部分权值低，和我们的逻辑方法一致。有些软件中描述成 **1 2 3 4**
 - **Big-Endian byte swap**：大端反转，有些软件中描述成 **2 1 4 3**
 - **Little-Endian byte swap**：小端反转，有些软件中描述成 **3 4 1 2**
- 默认值：默认设为9999，可修改，在“网关通讯异常处理”中选择了“使用默认值”，Tag点出错时，相应的Modbus寄存器的点值置成该默认值。

示例

用Modbus poll软件读取ModbusTCP服务示例，步骤如下：

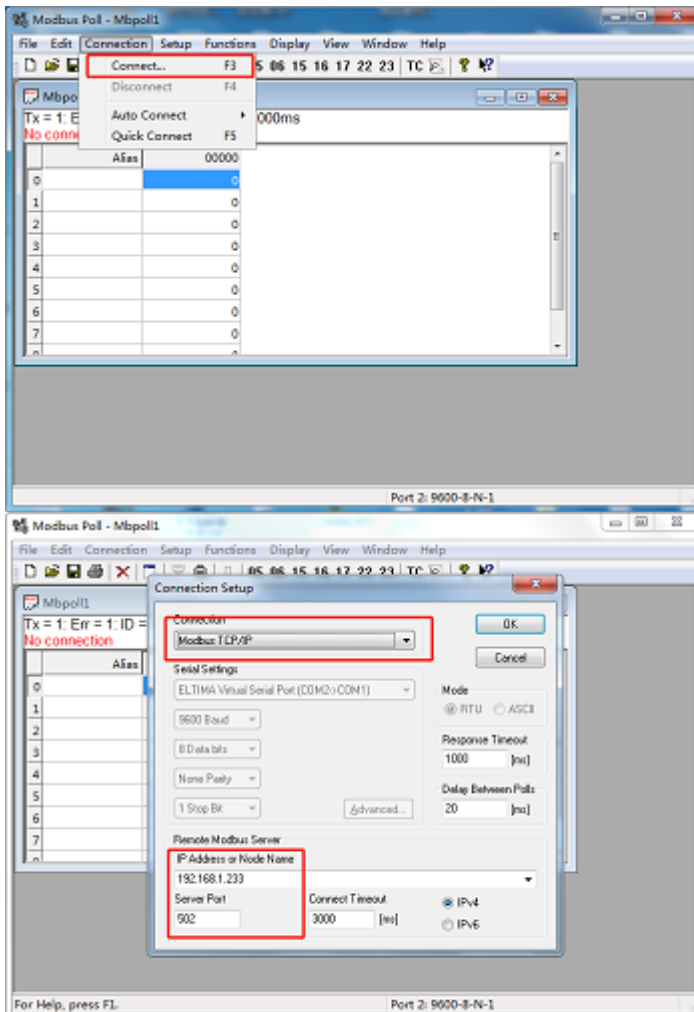
1. 使能Modbus TCP服务，地址映射如下：

The screenshot shows the LM Gateway software interface. The 'Modbus TCP' configuration is enabled with port 502. Below the configuration, a table lists the mapping of Modbus tags to registers, data types, and byte orders. The table is highlighted with a red border.

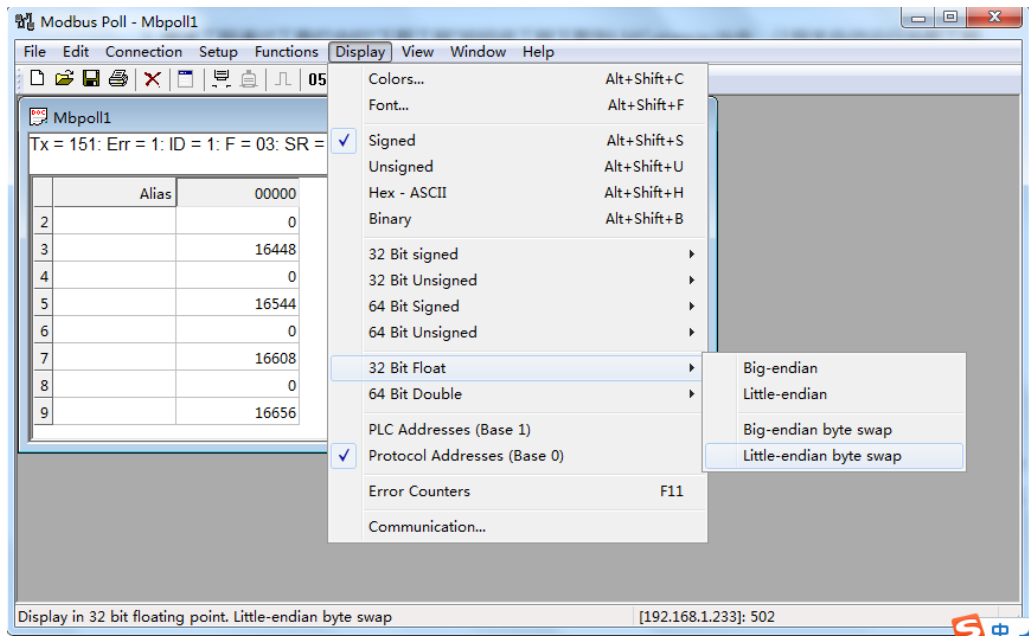
ID	名称	数据区	Modbus寄存器地址	Modbus数据类型	字节序	默认值
1	Device1.tag0001	4X (Holding Register)	1	float	Little-endian byte swap	9999
2	Device1.tag0002	4X (Holding Register)	3	float	Little-endian byte swap	9999
3	Device1.tag0003	4X (Holding Register)	5	float	Little-endian byte swap	9999
4	Device1.tag0004	4X (Holding Register)	7	float	Little-endian byte swap	9999
5	Device1.tag0005	4X (Holding Register)	9	float	Little-endian byte swap	9999
6	Device1.tag0006	4X (Holding Register)	11	float	Little-endian byte swap	9999
7	Device1.tag0007	4X (Holding Register)	13	float	Little-endian byte swap	9999
8	Device1.tag0008	4X (Holding Register)	15	float	Little-endian byte swap	9999
9	Device1.tag0009	4X (Holding Register)	17	float	Little-endian byte swap	9999

1. 将该工程通过工具栏中的“下载工程”按钮下载到LMGateway当中。

2. 打开Modbus Poll软件，点击上方工具栏中“Connection”下的“Connect...”，在弹出框中选择“Modbus TCP/IP”，输入LMGateway的IP地址和端口号，点击“OK”完成连接配置。



3. 根据GC中Modbus服务页面的映射地址、数据类型和字节序修改Modbus Poll软件工具栏中“Display”，查看寄存器数据。



如果此时页面上显示“**No connection**”，说明没有连接上LMGateway的ModbusTCP服务，请检查通讯配置。

5.2 BACnet

BACnet服务器实现了Tag点到BACnet对象的映射，允许支持BACnet的上位机通过BACnet IP或者BACnet MSTP协议读写Tag点。

设备ID:	<input type="text" value="123"/>	设备名称:	<input type="text" value="LM Gateway"/>	对象描述:	<input type="text" value="BACnet slave"/>	生产商名称:	<input type="text" value="LM"/>
生产商ID:	<input type="text" value="123"/>	位置:	<input type="text" value="CN"/>	BACnet字符编码:	<input type="text" value="UTF-8"/>		

上图为BACnet服务的设备属性，包含设备ID、设备名称、设备描述、生产商名称、生产商ID、位置和BACnet字符编码。

转成 BACnet 协议时，注意 BACnet 字符编码选择，默认为 UTF-8，若上位机需要中文显示，（例如江森 Metasys 系统）需要勾选为 Unicode。

UTF-8: Insight、Delta 、etc.

Unicode: Metasys、Niagara 、etc.

BACnet IP配置

使能BACnet IP服务，允许上位机通过BACnet IP协议经由网口连接访问设备。

端口号：设置BACnet IP侦听端口号，默认值是47808。

绑定端口：指定BACnet IP服务绑定的网卡(windows版本需设置成eth0)。

BBMD：勾选BBMD，设置为 BACnet 跨网段时使用，注册为外部设备，在“配置”对话框中填写注册为外部设备的上位机的 IP 地址。在对应的上位机处需要设置对应 BBMD 设备的 IP 地址为当前网关的 IP 地址。

设备ID:	<input type="text" value="123"/>	设备名称:	<input type="text" value="LM Gateway"/>	对象描述:	<input type="text" value="BACnet slave"/>	生产商名称:	<input type="text" value="LM"/>
生产商ID:	<input type="text" value="123"/>	位置:	<input type="text" value="CN"/>	BACnet字符编码:	<input type="text" value="UTF-8"/>		
<input type="checkbox"/> 启用BACnetIP服务	端口号:	<input type="text" value="47808"/>	绑定网口:	<input type="text" value="eth0"/>	<input type="checkbox"/> BBMD	<input type="button" value="配置"/>	
<input type="checkbox"/> 启用BACnetMSTP服务	串口号:	<input type="text" value="com1"/>	波特率:	<input type="text" value="38400"/>	MAC地址:	<input type="text" value="12"/>	超时时间(ms): <input type="text" value="60000"/>
<input type="button" value="保存"/>							

BACnet MSTP配置

使能BACnet MSTP服务，允许上位机通过BACnet MSTP协议经由串口连接访问设备。

MAC地址：BACnet MSTP服务的MAC地址。

超时时间：APDU的超时时间。

BACnet地址映射

将Tag点映射到BACnet的九种对象当中，配置步骤如下：

1. 选择需要映射的BaCnet上传表中的对象类型，支持的对象类型有 AI,AO,AV,BI,BO,BV,MSI,MSO,MSV；
2. 单击“添加”按钮；
3. 在弹出的选择点窗口中勾选需要映射的点；
4. 点击“确定”按钮完成映射点的添加。

重复上述操作可添加更多的点到地址列表。

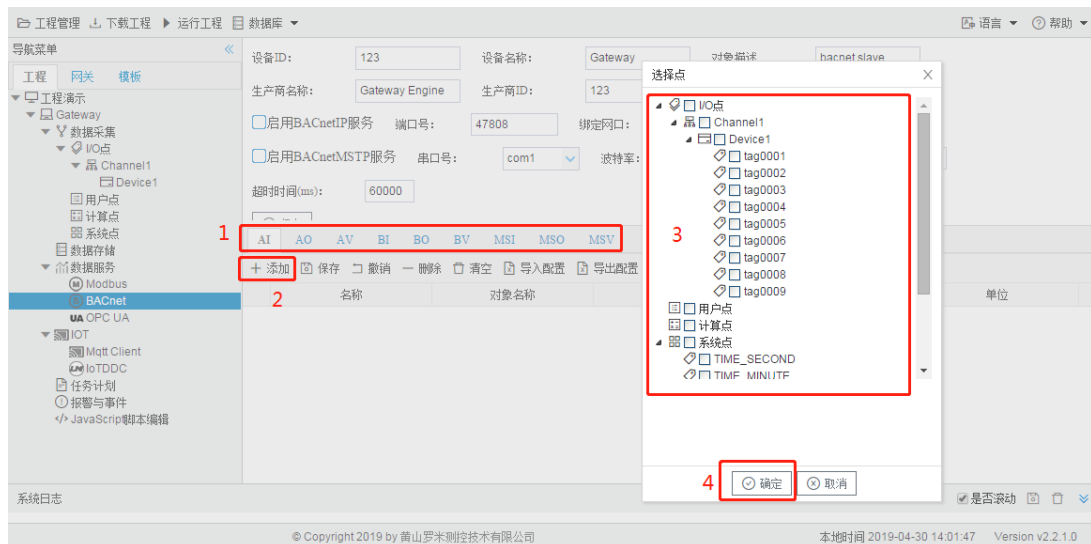


图5-2 BACnet对象类型与点的映射

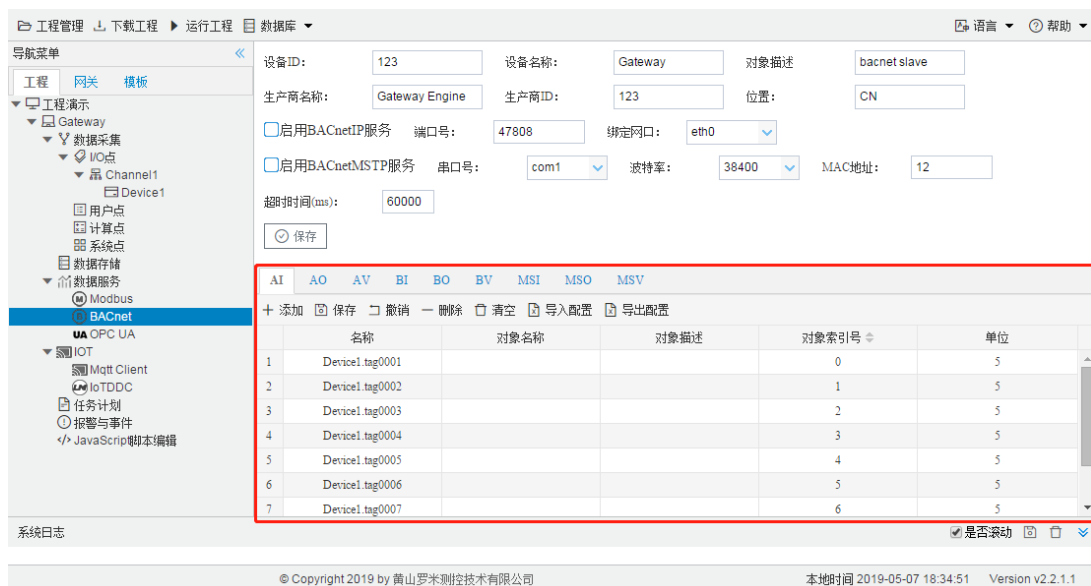


图5-2 BACnet服务点表

双击Tag点可修改Tag点映射到BACnet地址的详细设定，可配置项有：

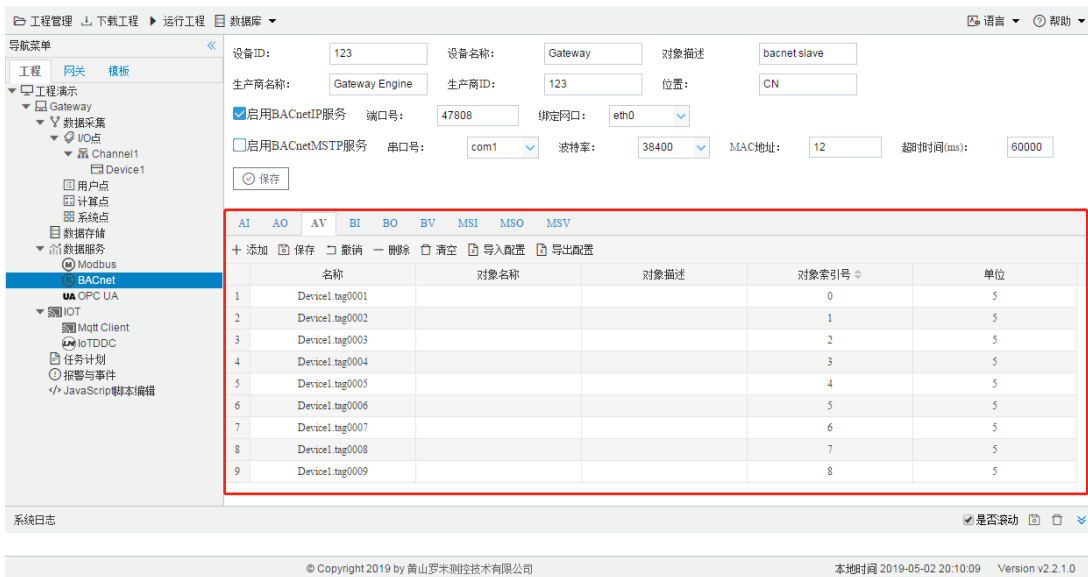
- 对象名称：可编辑，BACnet数据点的对象名称。
- 对象描述：可编辑，BACnet数据点的对象描述。
- 对象索引号：必填项，与AI,AO,AV,BI,BO,BV,MSI,MSO,MSV组合为AI0, AI1, AI2等。
- 单位：可编辑，通过下拉框选择。

MSI、MSO、MSV为多态，如果需要映射到BACnet的多态上，MSI、MSO、MSV的页面中“多态”列必须要有最少一个状态。

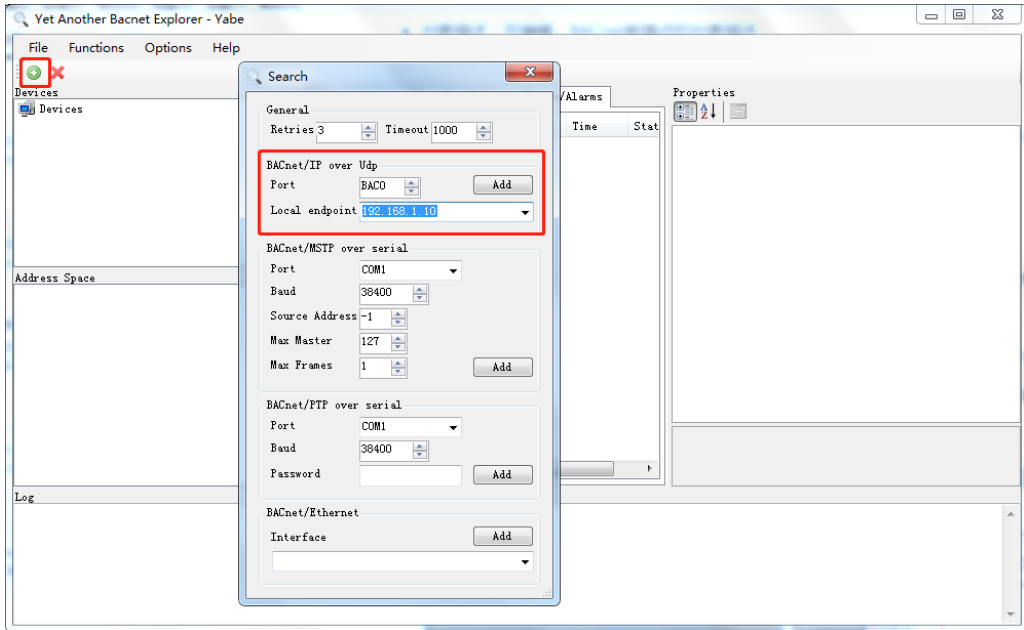
示例

用Yabe软件读取BACnetIP服务示例，步骤如下：

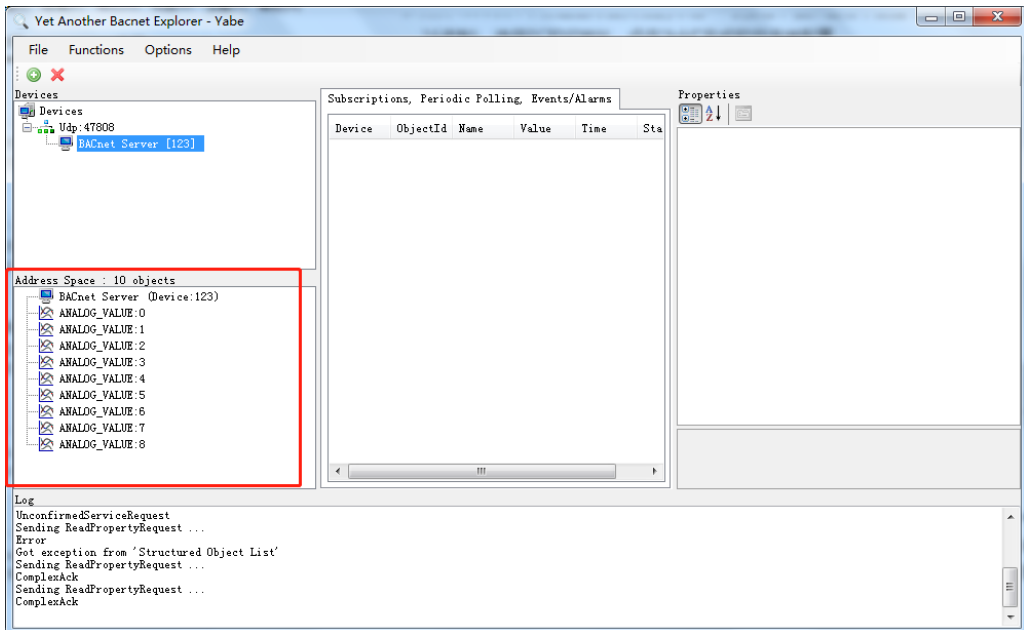
1. 使能BACnet IP服务，具体映射如下：



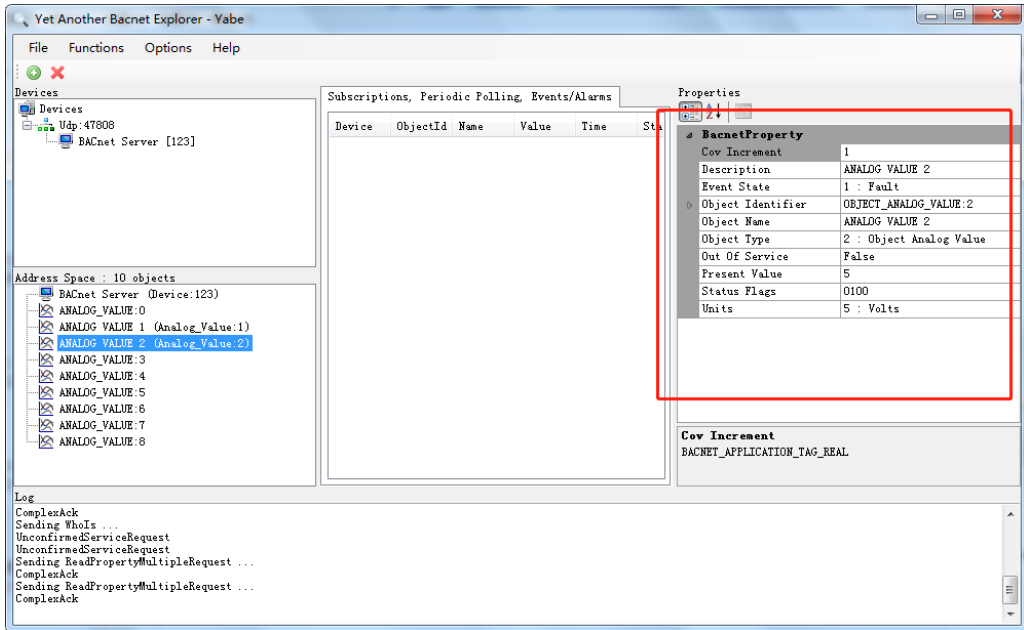
1. 将该工程通过工具栏中的“下载工程”按钮下载到LMGateway当中。
2. 打开Yabe软件，点击上方工具栏的绿色“+”号按钮，在弹出框中上图中的端口号(BAC0为上图中47808的16进制)、选择PC的IP地址，点击“Add”完成软件连接配置。



3. 在Udp:47808节点下有GC中BACnet页面设备ID的设备，说明已经连接上LMGateway的BACnet IP服务，单击此设备节点，就会在左侧中部显示搜索到的所有此设备映射出的BACnet设备和对象。



4. 点击每一个对象，就会显示该对象的所有属性。



5.3 OPC UA

OPC UA服务器实现了Tag点到对象属性的映射，允许上位机通过 OPC UA协议读写Tag点。

The screenshot shows a configuration window for OPC UA. At the top, there is a checked checkbox labeled '启用' (Enable). To its right are two input fields: '端口号:' (Port) with the value '4840' and '绑定网口:' (Bind Network Interface) with a dropdown menu showing 'eth0'. Below this, there are three radio button options for authentication: 'Anonymous' (selected), 'Username:' (with an empty text box), and 'Password:' (with an empty text box). To the right of these is a dropdown menu for '安全策略:' (Security Policy) with 'None' selected. Below the password field is a 'Certificate:' dropdown menu and a 'Load' button. At the bottom left, there is a '保存' (Save) button.

端口号：设置OPC UA侦听端口号，默认值是4840。

Anonymous：选中之后，启用匿名的方式连接 OPCUAServer。

Username、Password：选中之后，在“Username”填入用户名，“Password”填入密码。启用户名密码的

方式连接 OPCUAServer。

Certificate：选中之后，点击“Load”，导入证书的路径。启用交换证书的方式连接OPCUAServer。

安全策略：None，Basic256，Basic128Rsa15，Basic256Sha256

1. 当选中为 None时，OPCClient 和OPCServer 之间的通信将不会有加密和签名；
2. 当“安全策略”选中为 Basic256，Basic128Rsa15，Basic256Sha256中的任意一个时，OPCClient和OPCServer 之间的通信将会有签名或者签名和加密，提高客户端和服务器的通信安全。

OPC UA对象属性映射

将Tag点映射到OPC UA对象属性上，配置步骤如下：

1. 单击“添加”按钮；
2. 在弹出的选择点窗口中勾选需要映射的点；
3. 单击“确定”按钮完成映射点的添加。

重复上述操作可添加更多的点到地址列表。

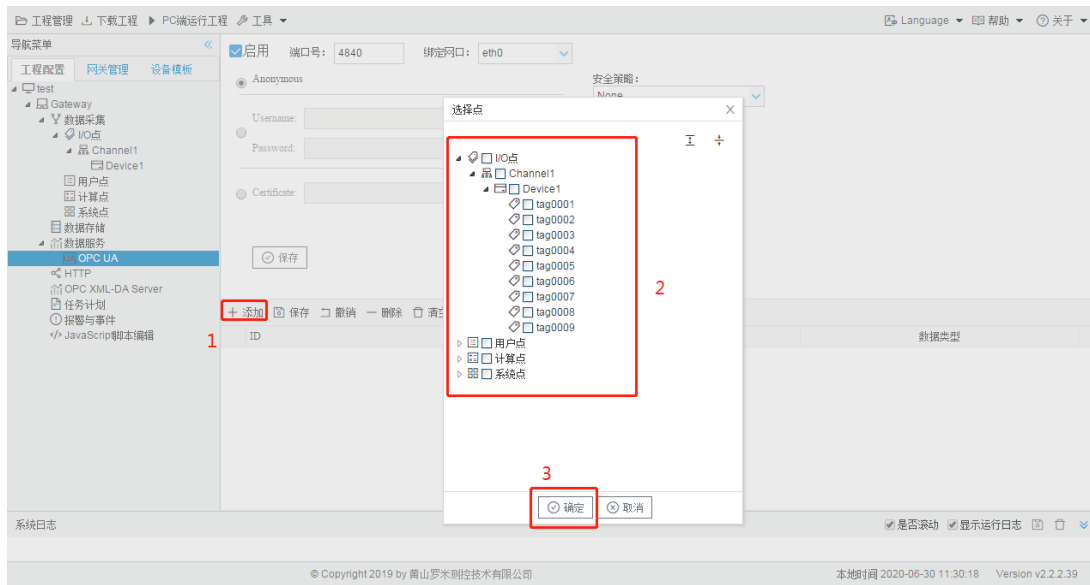
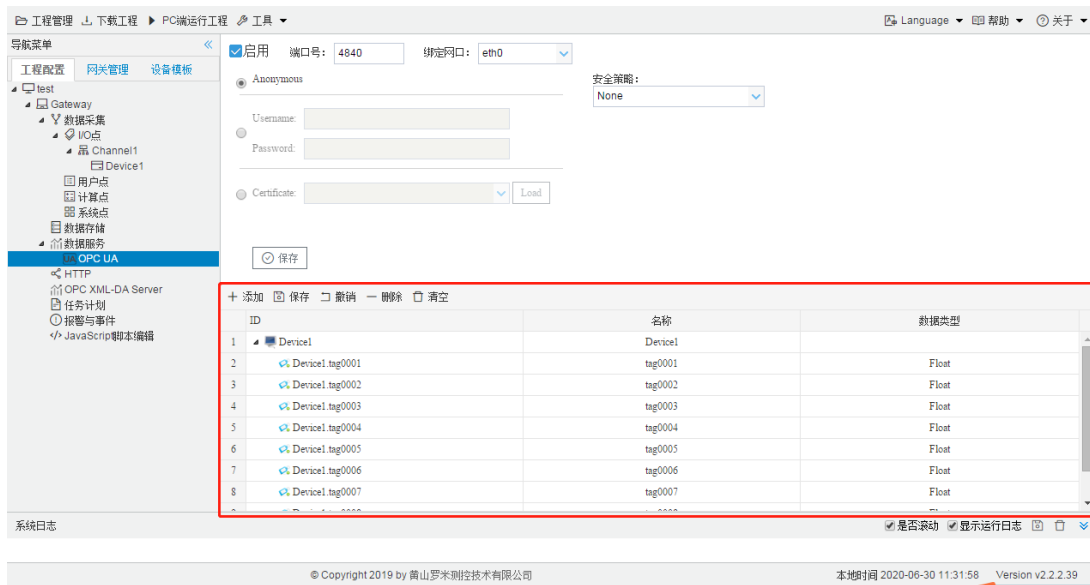


图5-3 OPC UA地址映射



将工程文件中的I/O、用户、计算、系统映射成为OPC UA中DeviceFolder文件夹下的对象，网关中的Tag点会映射成为对象中的属性。

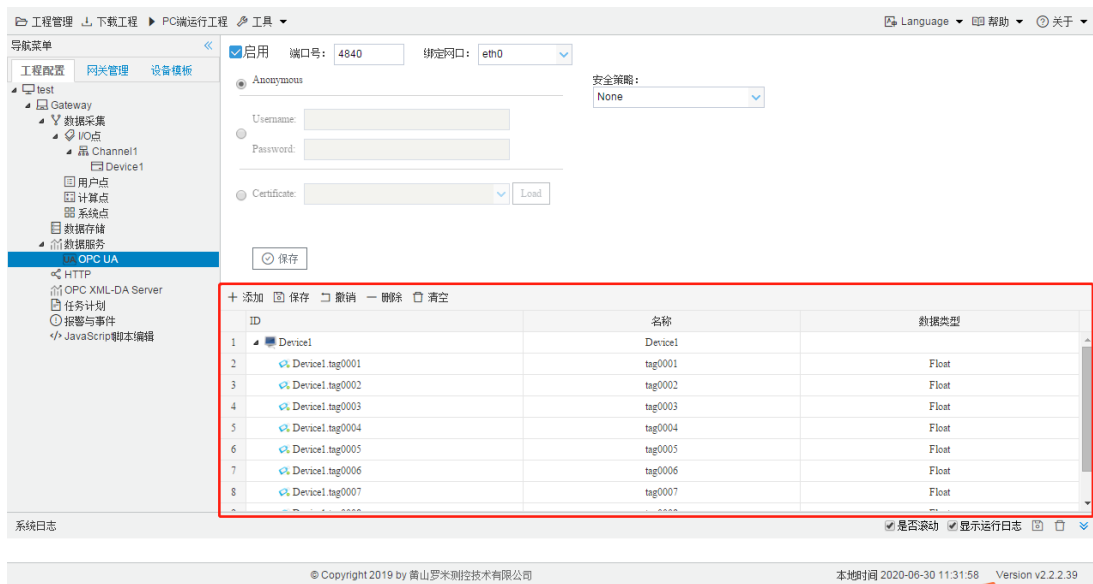
描述：对象或属性的描述信息。

数据类型：支持Boolean,UInt16,Int16,UInt32,Int32,Float六种。

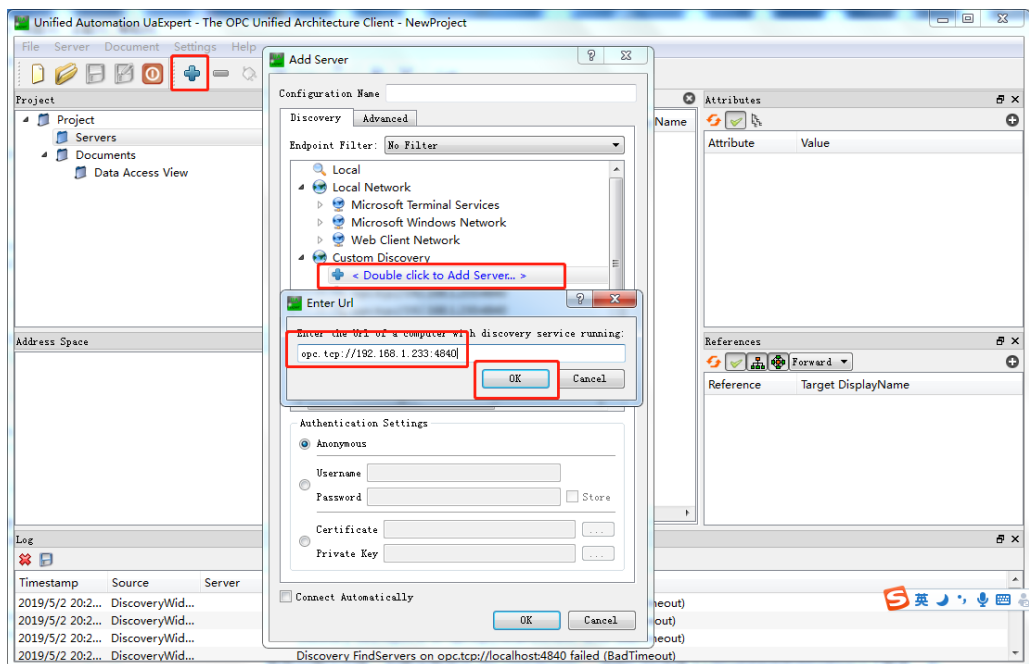
示例

用UaExpert软件读取OPC UA服务示例，步骤如下：

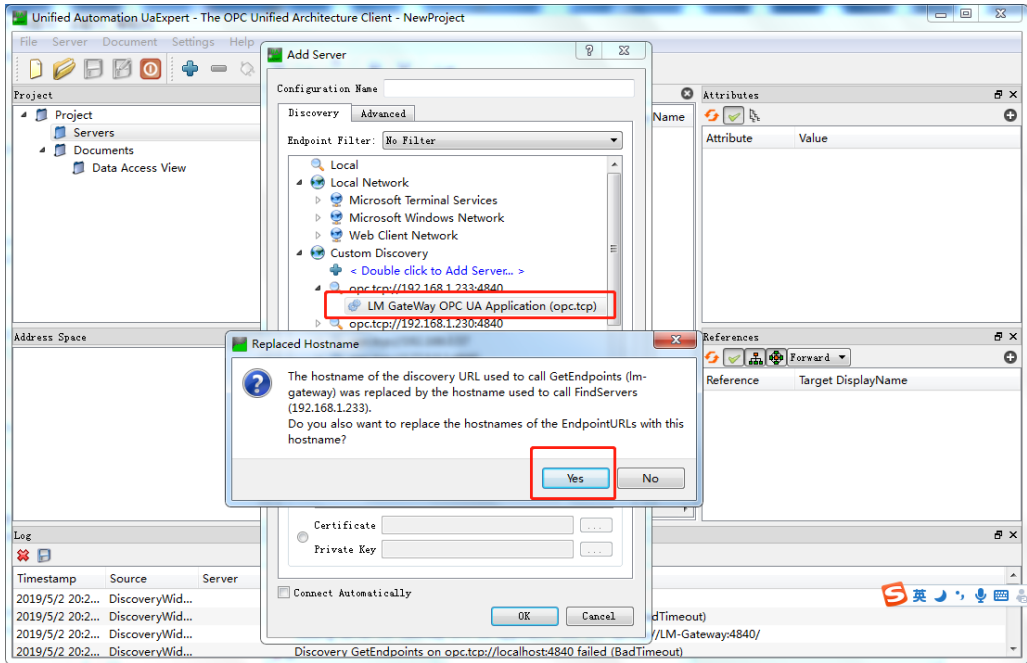
1. 使能OPC UA服务，对象属性映射如下：



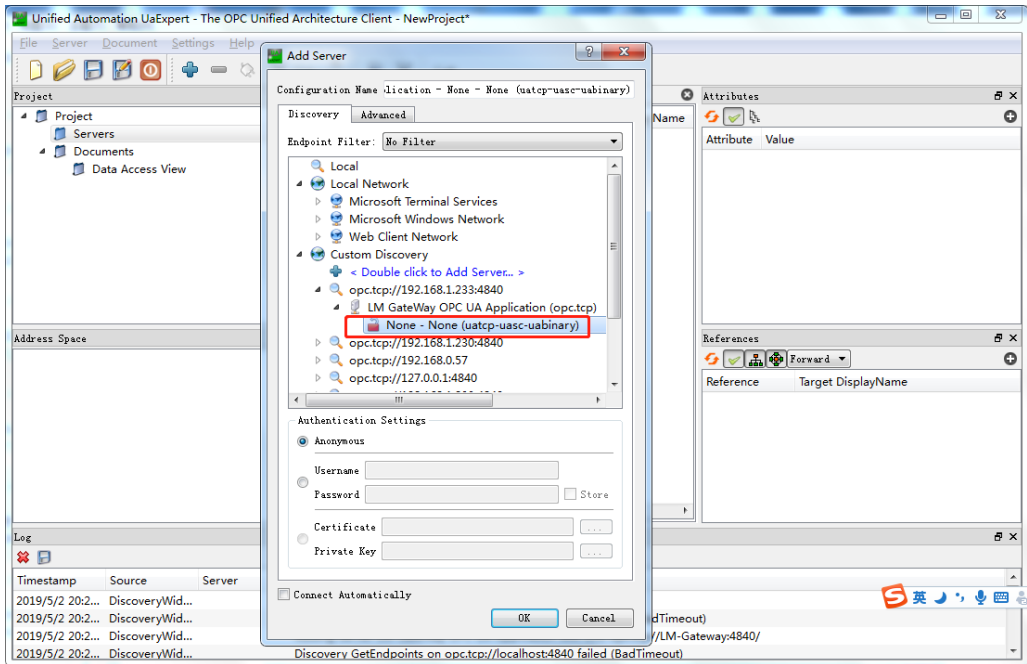
1. 将该工程通过工具栏中的“下载工程”按钮下载到LMGateway当中。
2. 打开UaExpert软件，点击上方工具栏中蓝色“+”号按钮，在标题为"Add Server"弹出框中双击“Double click to Add Server...”，输入LMGateway的IP地址和上图中的端口号，点击“OK”。



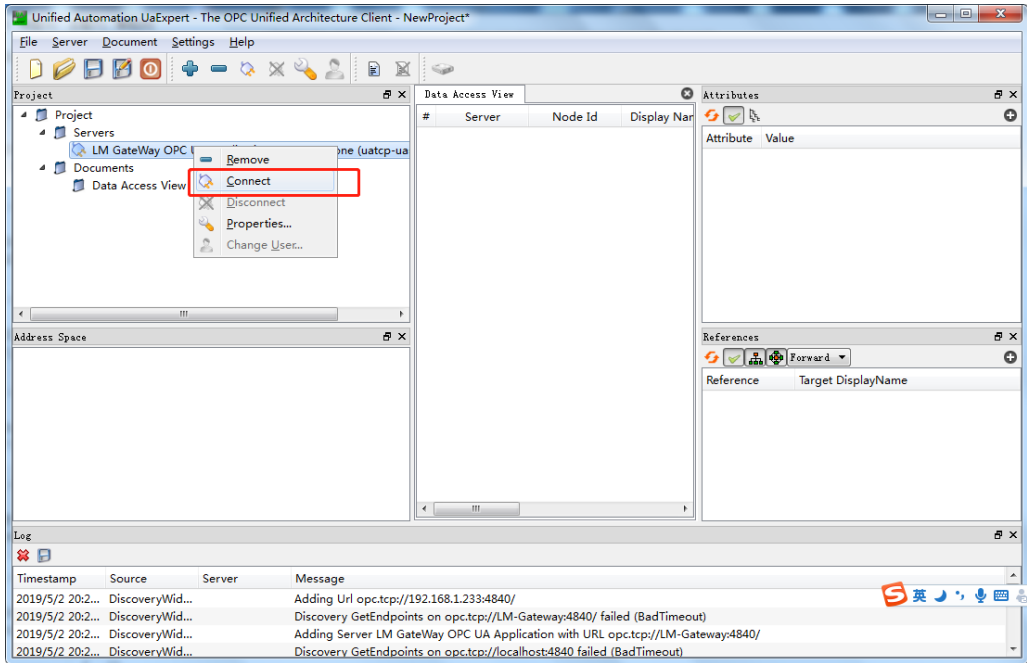
双击上图添加的opc://192.168.1.233:4840节点下的子节点，在弹出框中点击"yes"。



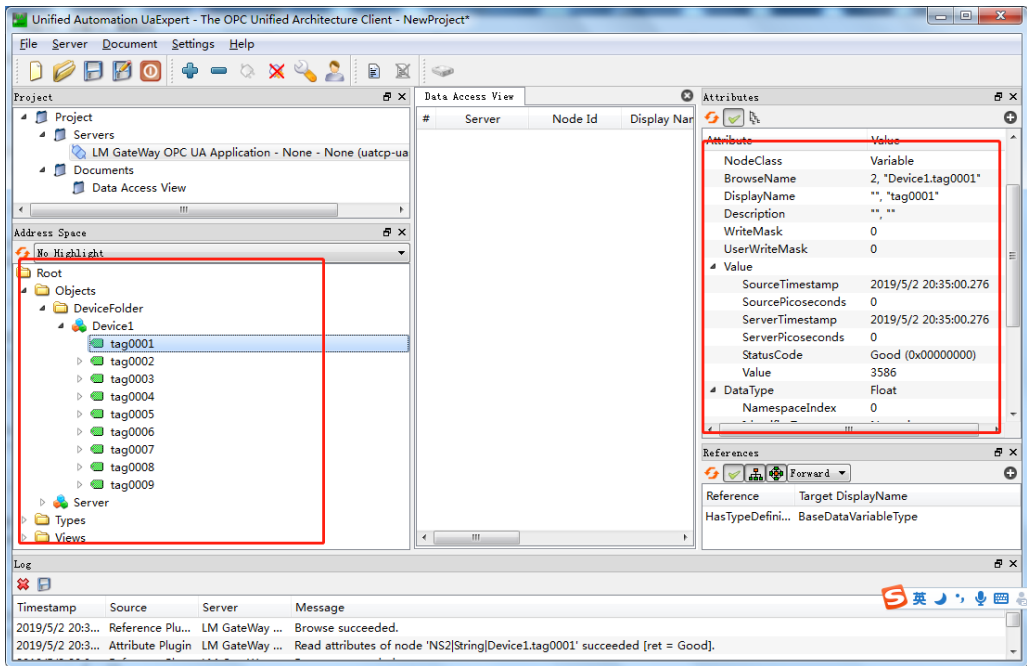
双击下图中的不加密的连接方式。



3. 此时在Servers下就会添加一个LM Gateway OPC UA节点，右键选择"Connect"。



1. 连接成功之后，会在左侧显示所有的映射点，单击每个点会在右侧显示该点的所有属性。



5.4 HTTP

网关作为HTTP客户端，将添加的需要数据的数据点，转换成相应的json格式的数据上传。

HTTP参数如下：

- 实时数据推送方法：POST/GET；实时数据推送的URL；
- 历史数据推送方法：POST/GET；历史数据推送的URL；
- HTTP的消息头（Header）；
- 超时时间：定义每一条数据发送后，等待服务端响应的的时间；
- 上传周期：网关每隔周期时间进行数据上传（上传周期需要大于超时时间）。

启用

实时数据接口：

历史数据接口：

Header		Content-Type ▼
Key	Value	
<input type="text" value="Content-Type"/>	<input type="text" value="application/json"/>	<input type="button" value="+"/> <input type="button" value="delete"/>

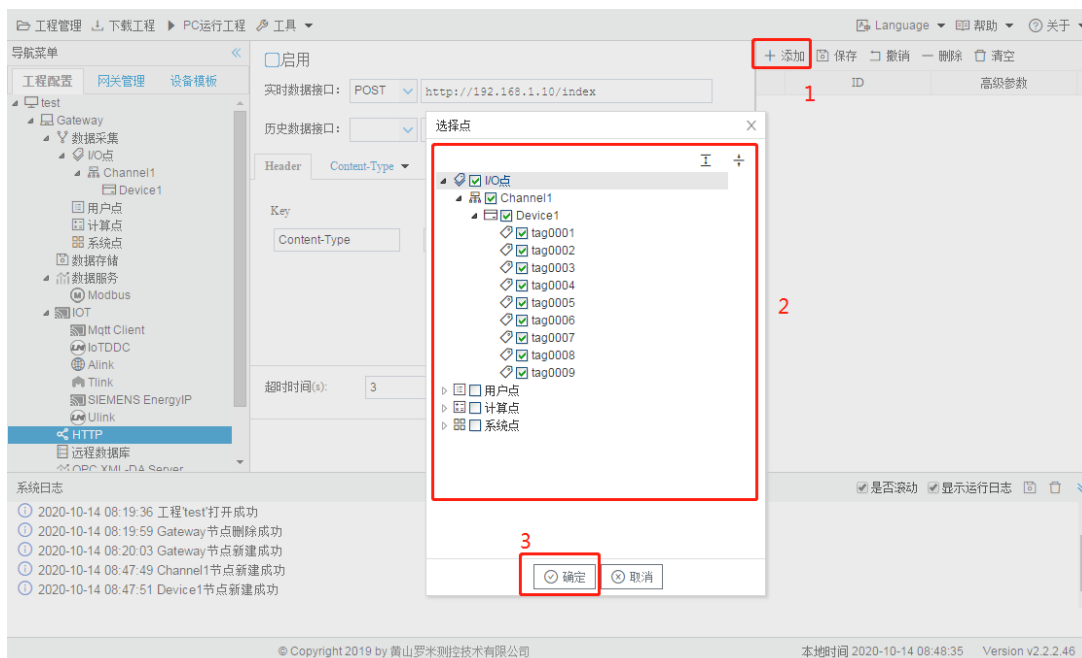
超时时间(s): 上传周期(s):

上传点添加

配置步骤如下：

1. 单击“添加”按钮；
2. 在弹出窗口中勾选需要上传的数据点；
3. 单击“确定”按钮完成映射点的添加。

重复上述操作可添加更多的点到上传列表。



高级参数

如果需要在mqtt上传的数据点添加额外的属性字段，可以添加“高级参数”（JSON对象格式）。

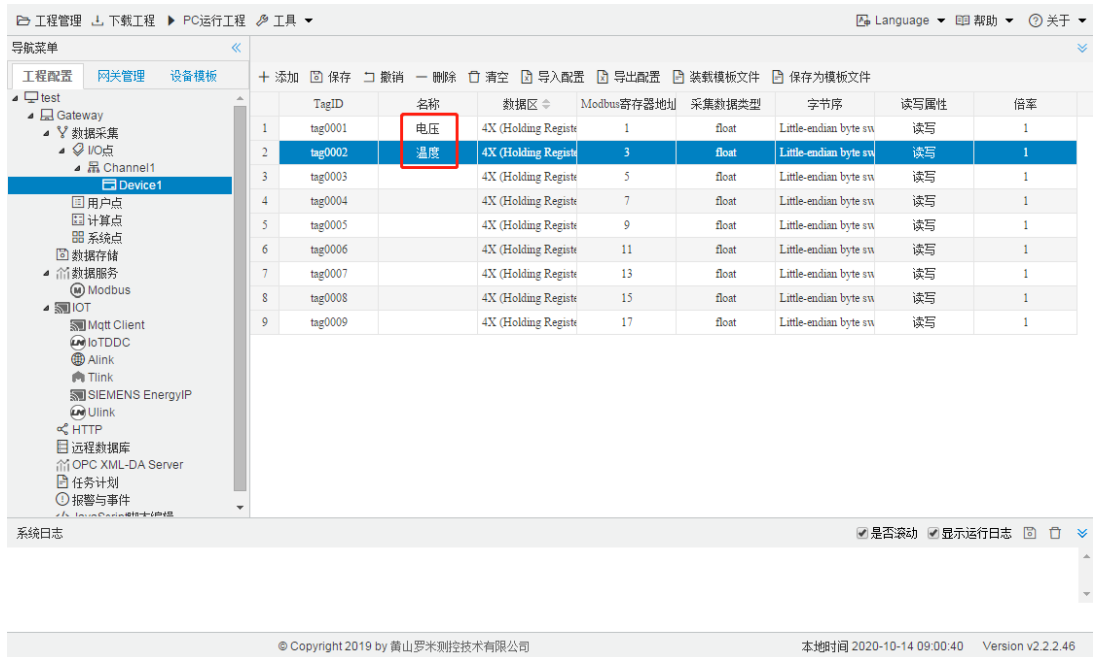
在Mqtt Client页面点击“添加”按钮，选择需要上传的Tag点，双击Tag点，在“高级参数”字段中添加所需要的json对象，如{"unit": "摄氏度"}。

该操作需要配合模板使用。

HTTP上传的json格式默认如下：

```
{
  "温度": 35.4,
  "电压": 212,
  "time": 1602637080
}
```

其中“温度”和“电压”分别为Device1.tag0001和Device1.tag0002的名称



如果客户需要修改上传的json格式，可以有如下方式：

1. 自行修改GC安装目录下mqtt/http/default.js，并下载到网关当中；

i. 参照default.js:

```
(function() {
    var timestamp = Math.round(new Date().getTime() / 1000);
    result = { "time": timestamp };
    data.forEach(function(tag) {
        if (result[tagExt[tag.Id].description] != "") {
            if (isNaN(tag["Val"]))
                result[tagExt[tag.Id].description] = tag["Val"];
            else {
                if (tag["Status"] == "Good") {
                    result[tagExt[tag.Id].description] = parseFloat(tag["Val"]);
                }
            }
        }
    });
    return JSON.stringify(result);
})();
```

其中 (function() {

.....

})();

为主体，js需要写在其中；

data为经过过滤后的数据，格式为

```
[
  {
    "Id":"Device1.tag0001",
    "Status":"Good",
    "Timestamp":1574926563,
    "Val":"1.000000"
  },
  {
    "Id":"Device1.tag0002",
    "Status":"Good",
    "Timestamp":1574926563,
    "Val":"3.000000"
  }
]
```

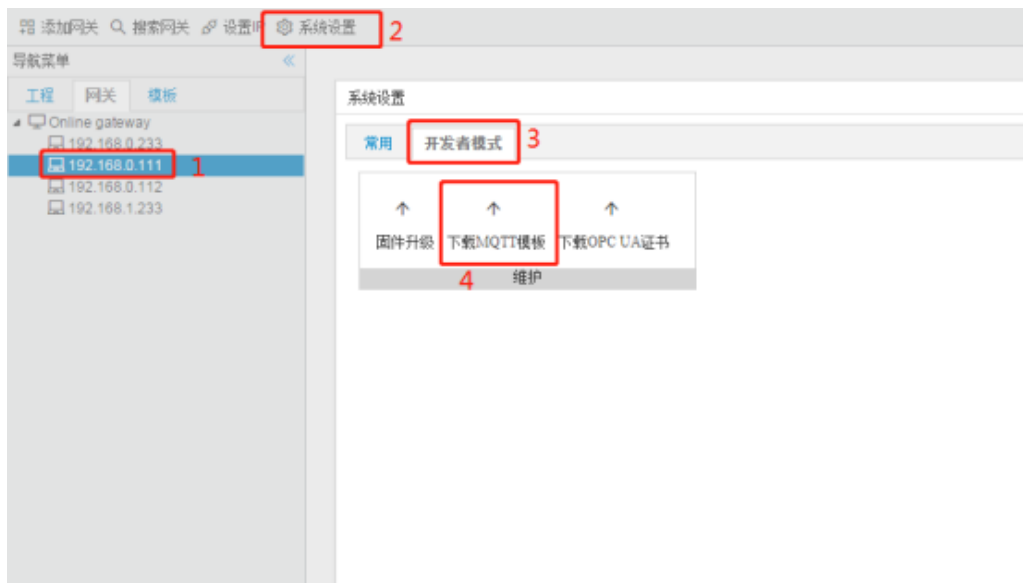
tagExt为tag点的额外属性，包括采集页面中的名称（**description**）、tag点所在的设备名称、i

```
```json
{
 "Device1.tag0001":{
 "description":"",
 "deviceCode":"Device1",
 "objectType":"AV",
 "tagCode":"tag0001"
 },
 "Device1.tag0002":{
 "description":"",
 "deviceCode":"Device1",
 "objectType":"AV",
 "tagCode":"tag0002"
 }
}
```
```

模板最后返回主题需要上传的字符串：

1. 模板编写完成之后，需要将模板名称改为**default.js**；
2. 可以先用配置工具“PC端运行工程”测试模板能否正确运行；

3. 需要下载模板到网关：在“系统设置”中选择“开发者模式”，密码为“luomi”，选择“下载MQTT模板”，之后下载工程，网关就会根据新建的模板上传数据。



1. 联系我们并提供需要的json格式

5.5 远程数据库

目前仅支持mysql远程数据库。

mysql远程数据库参数如下：

- mysql远程数据库的IP地址和端口号；
- mysql远程数据库的用户名、密码和数据库名称；
- SQL语句：需要编写具体插入数据的sql语句；
- 存储模式：周期存储（需要设置存储周期）和准点存储（需要添加每天存储的整点时间）；

数据库类型： 启用

IP地址： 端口号：

用户名： 密码： 数据库名称：

SQL语句：

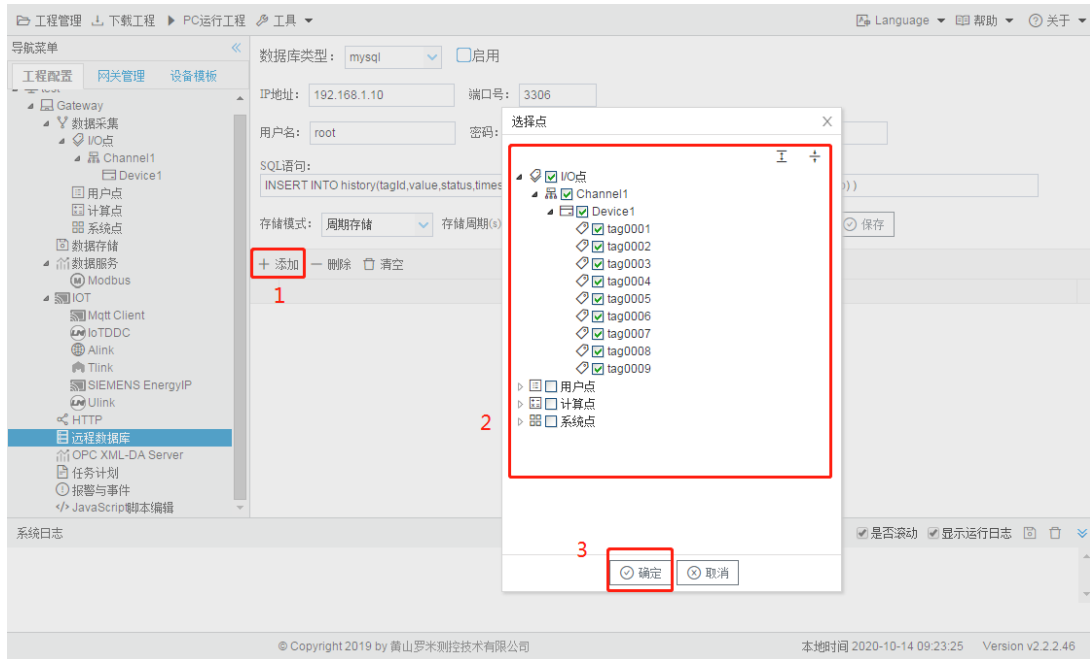
存储模式： 存储周期(s)： 整点时间：

上传点添加

配置步骤如下：

1. 单击“添加”按钮；
2. 在弹出窗口中勾选需要上传的数据点；
3. 点击“确定”按钮完成映射点的添加。

重复上述操作可添加更多的点到上传列表。



5.6 OPC XML-DA Server

OPC XML-DA Server默认启用;

端口号: 8080;

URL: <http://X.X.X.X/soap> (例如<http://192.168.1.233:8080/soap>)



The image shows a dialog box titled "OPC XML-DA服务器设置" (OPC XML-DA Server Settings) with a close button (X) in the top right corner. The dialog contains the following elements:

- A checked checkbox labeled "启用OPC XML-DA服务器" (Enable OPC XML-DA Server).
- A text input field labeled "端口号:" (Port Number) with the value "8080".
- A text input field labeled "URL:" with the value "http://X.X.X.X/soap".
- A "确定" (OK) button at the bottom center.

第六章 IoT

LMGateway可以通过MQTT协议与云服务器通信。支持阿里云、百度云及其他私有云服务器。

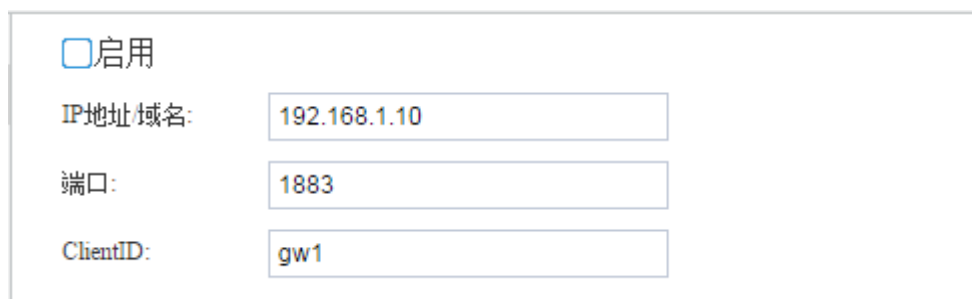
罗米测控推出IoT数据中心，可方便用户验证网关、云服务、微信公众号等物联网应用。

6.1 Mqtt Client

用于配置Mqtt连接属性、发布订阅的主题、需要上传的tag点等。

6.1.1 基础连接属性

- 勾选“启用”框，网关开启MQTT客户端；
- IP地址/域名、端口：MQTT broker的IP/域名、端口号（默认1883）；
- ClientID：客户端唯一标识，不可重复。broker和Mqtt客户端通过ClientID 保持唯一的 TCP 连接，如出现重复ClientID，则broker会踢掉前一个。



The image shows a configuration form for an MQTT client. It includes a checkbox labeled '启用' (Enable) which is checked. Below it are three input fields: 'IP地址/域名' (IP address/domain) with the value '192.168.1.10', '端口' (Port) with the value '1883', and 'ClientID' with the value 'gw1'.

图6-1 mqtt基础连接属性

6.1.2 进阶连接属性

1. 基本配置（General）

- Keep Alive(s): 心跳，客户端在Connect的时候设置 Keep Alive 时长。如果服务端在 $1.5 * \text{KeepAlive}$ 时间内没有收到客户端的报文，它必须断开客户端的网络连接，默认为60秒。
- Timeout(s): 定义客户端发送信息到云端响应的最大时间间隔，客户端超过时间没有得到相应后会主动中断连接，默认为30秒。
- Clean Session:

NO——开启会话重用机制。网络断开重连后，恢复之前的Session 信息。需要客户端和服务端有 相关Session持久化机制。

YES——关闭会话重用机制。每次Connect都是一个新Session，会话仅持续和网络连接同样长的时间。

The screenshot shows the 'General' configuration page for MQTT. It has four tabs: 'General', 'User Credentials', 'SSL/TLS', and 'Last Will and Testament'. The 'General' tab is active. Below the tabs, there are three settings: 'Keep Alive(s)' with a text input field containing '60', 'Timeout(s)' with a text input field containing '30', and 'Clean Session' with a radio button set to 'No'.

图6-2 mqtt General

1. 登录凭证（User Credentials）

用户名、密码：根据broker的配置，确定mqtt连接时是否需要用户名、密码的验证。

The screenshot shows the 'User Credentials' configuration page for MQTT. It has four tabs: 'General', 'User Credentials', 'SSL/TLS', and 'Last Will and Testament'. The 'User Credentials' tab is active. Below the tabs, there are two input fields: '用户名:' and '密码:'.

图6-3 mqtt User Credentials

2. 通信加密（SST/TLS）

SSL有两种连接方式，默认不启用SSL安全连接：

- 单向验证——需要上传CA文件到网关，当网关与broker连接时，网关作为客户端提供证书供broker验证。CA文件由云服务商提供。
- 双向验证——需要上传CA文件、Cert文件、KEY文件到网关，当网关与broker连接时，网关与broker均向对方提供证书。CA文件、Cert文件、KEY文件均由云服务商提供。

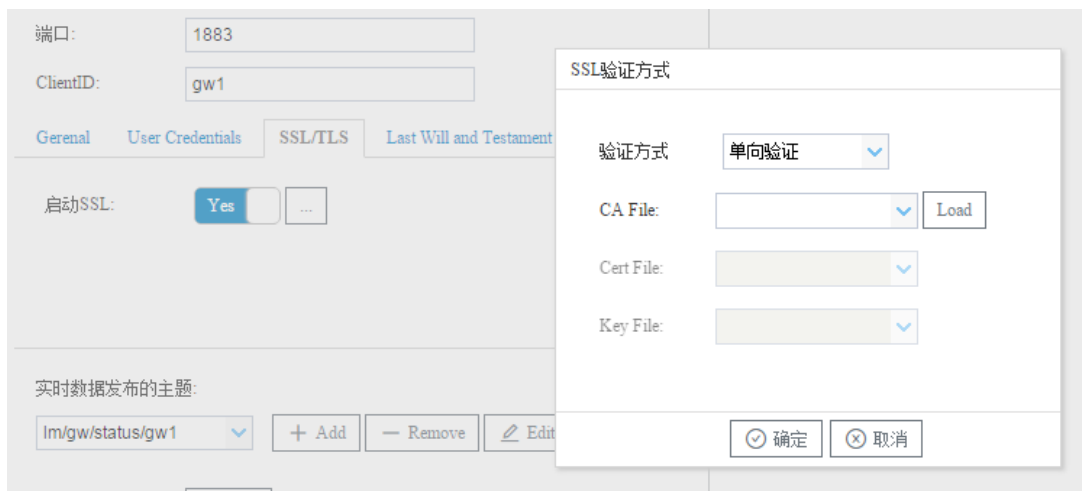


图6-4 mqtt User SSL/TLS

1. 遗嘱配置（Last Will and Testament）

遗嘱是当网络连接断开时，客户端根据主题发布消息。启用遗嘱时需要设置遗嘱主题和遗嘱消息。

默认不启用遗嘱。



图6-5 mqtt User Last Will and Testament

6.1.3 发布、订阅主题配置

实时数据发布主题：多主题、多频率、不同格式上传

可通过增删改按钮对实时数据主题进行操作，每个主题可以使用不同的主题配置、不同的上传频率、不同的数据过滤方式、不同的数据格式模板等。

添加的主题会显示到实时主题下拉列表中。选择下拉列表中一个主题，点击右侧“添加”按钮，选择该主题上传的Tag点。

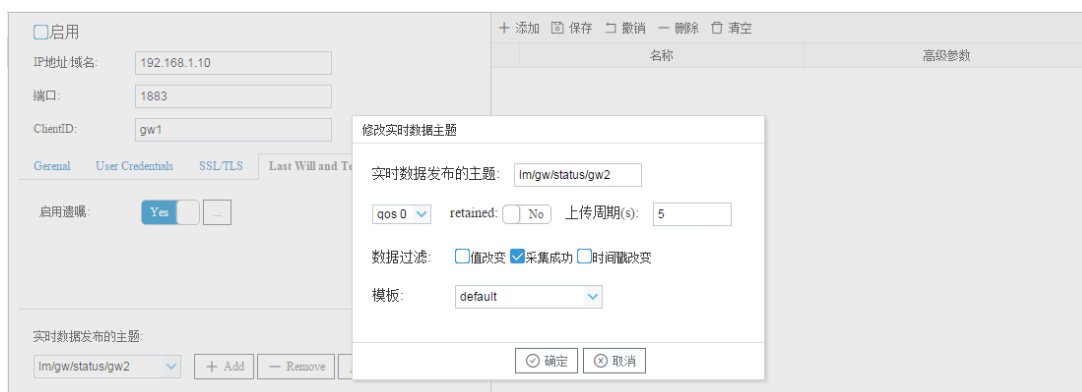
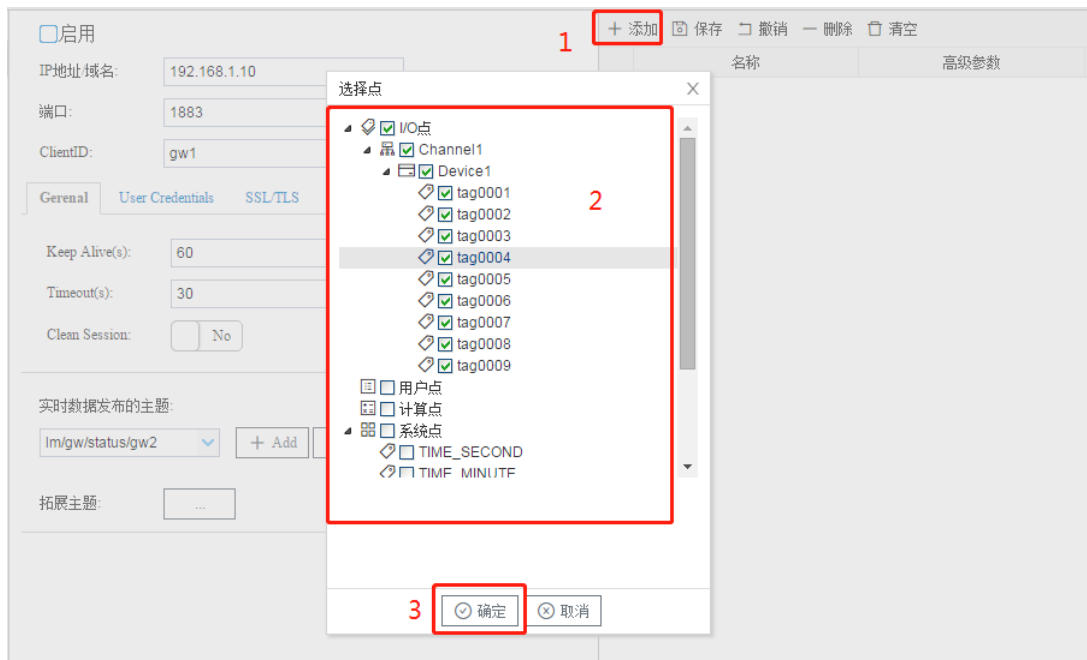


图6-6 mqtt实时数据发布主题

实时数据主题窗口包括以下配置项：

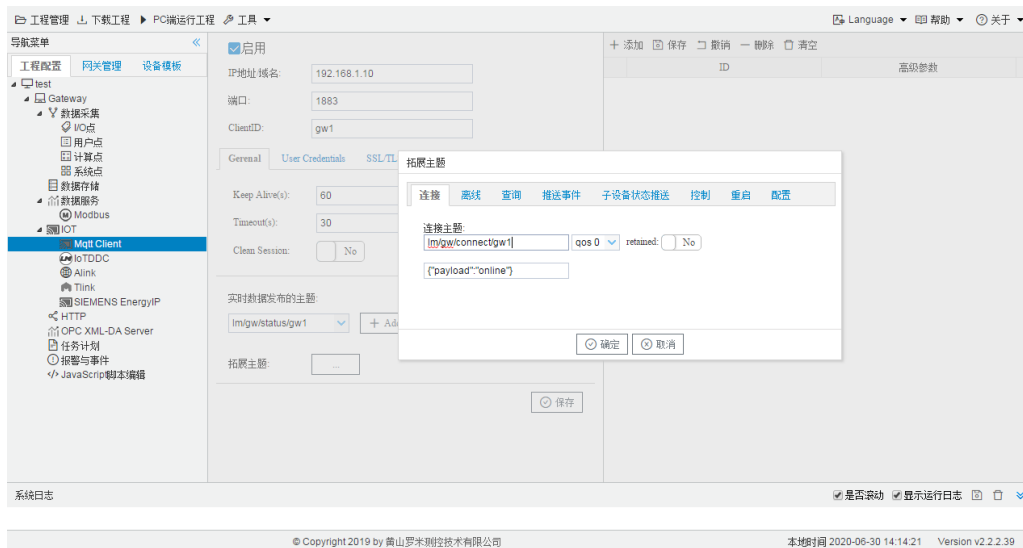
- QoS：传输消息等级，默认为qos0
 - qos0：最多一次的传输。也就是发出去就删掉。（速度快）
 - qos1：至少一次的传输。发出去之后必须等待ack，没有ack，就要找时机重发
 - qos2：只有一次的传输。消息id将拥有一个简单的生命周期。（可靠性高）
- retained：默认为No
 - Yes：表示发送的消息需要一直持久保存（不受服务器重启影响），不但要发送给当前的订阅者，并且以后新来的订阅了此Topic name的订阅者会马上得到推送。备注：当订阅该Topic的客户端上线后，只会接收到最新的一条消息。
 - No：仅仅为当前订阅者推送此消息。
- 上传周期：mqtt发布消息的频率。
- 数据过滤：以Tag点的值改变、是否采集成功（质量戳）和时间戳改变作为上传条件。
- 模板：数据上传的格式，用户可以通过下拉框选择已有的模板。可根据用户需求定制模板。



添加上传点

拓展主题

- 连接主题：mqtt连接成功时，将填写的字符串发布到相应的主题。



- 离线数据发布：网关在mqtt断线时，实时数据会根据上传周期将数据缓存在网关中。在mqtt恢复连接时，网关会在离线主题依次发布缓存数据。
 - 网关设备缓存离线数据的容量为20M，当离线数据大于20M时，网关就会缓存新的一条离线数据时，删除最早的一条离线数据。
- 查询数据主题：在配置界面中“查询数据订阅的主题”一项输入需要订阅的主题，当选中default.js时，网关订阅到如下格式的数据，就进行数据查

询的操作，网关会在“查询数据发布的主题”发布此次查询到的结果。

```
{ "operate": "read", "devices": [
  { "deviceCode": "Device_1", "tags": null },
  { "deviceCode": "Device_2",
    "tags": [
      { "tagCode": "tag0008" }
    ]
  }
]
}
```

- 推送事件主题：如果在“报警与事件”(8.1章节)当中配置了报警事件，当报警事件触发或者解除时，网关通过推送事件的主题将该事件发布到 **Broker**。
- 子设备状态推送主题：子设备为网关通讯的设备，在子设备上线和下线时会将状态推送到配置主题。
- 写操作主题：在配置界面中“写操作订阅的主题”一项输入需要订阅的主题，当选中 **default.js** 时，发布如下格式的数据，网关进行相应的写操作。

```
[
  {
    "operate": "write", //操作类型
    "deviceCode": "Device1", //采集协议页面中的设备名称
    "tagCode": "tag0001", //数据点的名称
    "val": "10" //需要写入的数值，字符串类型
  }
]
```

在网关写操作结束，网关会在“控制响应发布的主题”发布此次查询到的结果。以 **default.js** 为例。

```
{
  "deviceCode": "Device1", //采集协议页面中的设备名称
  "message": "写操作返回信息",
  "status": true, // true表示写操作成功, false表示写操作失败
  "tagCode": "tag0001", //数据点的名称
  "value": "10" //写入的数值，字符串类型
}
```

- 重启网关主题：在配置界面中“重启网关订阅的主题”一项输入需要订阅的主题，订阅到如下所示的格式的数据，重启网关。

```
{"operate": "reboot"}
```

- 配置网关主题：GC配置生成的数据库文件发布到配置网关的主题当中，远程配置网关。

拓展主题

连接
离线
查询
推送事件
子设备状态推送
控制
重启
配置

控制订阅的主题:

qos 0
▼
default.js
▼

控制响应发布的主题:

qos 0
▼
retained: No
default.js
▼

✔ 确定
✘ 取消

图6-7 mqtt拓展主题

高级参数

如果需要在mqtt上传的数据点添加额外的属性字段，可以添加“高级参数”（JSON对象格式）。

在Mqtt Client页面点击“添加”按钮，选择需要上传的Tag点，双击Tag点，在“高级参数”字段中添加所需要的json对象，如{"unit": "摄氏度"}。

该操作需要配合模板使用。

| + 添加 📁 保存 🗑 撤销 - 删除 🧼 清空 | | |
|--------------------------------------|-----------------|------|
| | 名称 | 高级参数 |
| 1 | Device1.tag0001 | |
| 2 | Device1.tag0002 | |
| 3 | Device1.tag0003 | |
| 4 | Device1.tag0004 | |

图6-8 mqtt高级参数

6.1.4 模板说明

Mqtt Client页面的主题中，实时数据、查询、控制、控制返回、重启的主题具有模板功能。

实时数据

目前支持8种模板。

1. **default.js**: 因为生成的json格式中不包含质量戳，所以需要在“数据过滤”中勾选“采集成功”，生成的json格式如下：

```
{
  "Device1":{
    "tag0001":2,
    "tag0002":3,
    "tag0003":1.23,
    "tag0004":0,
    "tag0005":0,
    "tag0006":0,
    "tag0007":0,
    "tag0008":0,
    "tag0009":0
  },
  "clientid":"gw1",
  "system":{
    "TIME_DAY":19,
    "TIME_HOUR":17,
    "TIME_MINUTE":38,
    "TIME_MONTH":6,
    "TIME_SECOND":55,
    "TIME_WDAY":3,
    "TIME_YEAR":2019
  },
  "time":"1560937137"
}
```

1. **DCC.js**: 生成的json格式如下：


```

{
  "Device1":[
    {
      "description":"",
      "deviceCode":"Device1",
      "id":"Device1.tag0001",
      "objectType":"AV",
      "status":"Good",
      "tagCode":"tag0001",
      "timestamp":1560937562,
      "val":"2.000000"
    },
    {
      "description":"",
      "deviceCode":"Device1",
      "id":"Device1.tag0002",
      "objectType":"AV",
      "status":"Good",
      "tagCode":"tag0002",
      "timestamp":1560937562,
      "val":"3.000000"
    },
    {
      "description":"",
      "deviceCode":"Device1",
      "id":"Device1.tag0003",
      "objectType":"AV",
      "status":"Good",
      "tagCode":"tag0003",
      "timestamp":1560937562,
      "val":"1.230000"
    },
    {
      "description":"",
      "deviceCode":"Device1",
      "id":"Device1.tag0004",
      "objectType":"AV",
      "status":"Good",
      "tagCode":"tag0004",
      "timestamp":1560937562,
      "val":"0.000000"
    },
    {
      "description":"",
      "deviceCode":"Device1",
      "id":"Device1.tag0005",
      "objectType":"AV",
      "status":"Good",

```

```

        "tagCode":"tag0005",
        "timestamp":1560937562,
        "val":"0.000000"
    },
    {
        "description":"",
        "deviceCode":"Device1",
        "id":"Device1.tag0006",
        "objectType":"AV",
        "status":"Good",
        "tagCode":"tag0006",
        "timestamp":1560937562,
        "val":"0.000000"
    },
    {
        "description":"",
        "deviceCode":"Device1",
        "id":"Device1.tag0007",
        "objectType":"AV",
        "status":"Good",
        "tagCode":"tag0007",
        "timestamp":1560937562,
        "val":"0.000000"
    },
    {
        "description":"",
        "deviceCode":"Device1",
        "id":"Device1.tag0008",
        "objectType":"AV",
        "status":"Good",
        "tagCode":"tag0008",
        "timestamp":1560937562,
        "val":"0.000000"
    },
    {
        "description":"",
        "deviceCode":"Device1",
        "id":"Device1.tag0009",
        "objectType":"AV",
        "status":"Good",
        "tagCode":"tag0009",
        "timestamp":1560937562,
        "val":"0.000000"
    }
],
"clientid":"gw1",
"time":"1560937563"
}

```

1. Alink.js: 因为生成的json格式中不包含质量戳, 所以需要在“数据过滤”中勾选“采集成功”, 生成的json格式如下:

```
{
  "id": "1",
  "method": "thing.event.property.post",
  "params": {
    "Device1_tag0001": 2,
    "Device1_tag0002": 3,
    "Device1_tag0003": 1.23,
    "Device1_tag0004": 0,
    "Device1_tag0005": 0,
    "Device1_tag0006": 0,
    "Device1_tag0007": 0,
    "Device1_tag0008": 0,
    "Device1_tag0009": 0,
    "system_TIME_DAY": 19,
    "system_TIME_HOUR": 17,
    "system_TIME_MINUTE": 43,
    "system_TIME_MONTH": 6,
    "system_TIME_SECOND": 43,
    "system_TIME_WDAY": 3,
    "system_TIME_YEAR": 2019
  },
  "version": "1.0"
}
```

1. HuaYuan.js: 生成的json格式如下:

```
{
  "ClientId": "gw1",
  "Devices": [
    {
      "DeviceName": "Device1",
      "Tags": [
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0001",
          "Val": "2.000000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0002",
          "Val": "3.000000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0003",
          "Val": "1.230000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0004",
          "Val": "0.000000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0005",
          "Val": "0.000000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0006",
          "Val": "0.000000"
        },
        {
          "PickTime": "2019-06-19 17:46:52",
          "Status": "Good",
          "TagCode": "tag0007",
          "Val": "0.000000"
        }
      ]
    }
  ]
}
```

```
    {
      "PickTime": "2019-06-19 17:46:52",
      "Status": "Good",
      "TagCode": "tag0008",
      "Val": "0.000000"
    },
    {
      "PickTime": "2019-06-19 17:46:52",
      "Status": "Good",
      "TagCode": "tag0009",
      "Val": "0.000000"
    }
  ]
},
"UploadTime": "2019-06-19 17:46:53"
}
```

1. InfluxDB.js: 生成的json格式如下:

```

[
  {
    "description": "",
    "id": "gw1.Device1.tag0001",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 2
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0002",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 3
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0003",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 1.23
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0004",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0005",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0006",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  }
]

```

```
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0007",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0008",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  },
  {
    "description": "",
    "id": "gw1.Device1.tag0009",
    "objectType": "AV",
    "status": "Good",
    "time": "1560966486",
    "val": 0
  }
]
```

1. JinHe.js: 因为生成的json格式中不包含质量戳，所以需要在“数据过滤”中勾选“采集成功”，生成的json格式如下：

```
{
  "Device1":[
    {
      "id":"Device1.tag0001",
      "tsp":"2019-06-19 17:48:50",
      "val":"2.000000"
    },
    {
      "id":"Device1.tag0002",
      "tsp":"2019-06-19 17:48:50",
      "val":"3.000000"
    },
    {
      "id":"Device1.tag0003",
      "tsp":"2019-06-19 17:48:50",
      "val":"1.230000"
    },
    {
      "id":"Device1.tag0004",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    },
    {
      "id":"Device1.tag0005",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    },
    {
      "id":"Device1.tag0006",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    },
    {
      "id":"Device1.tag0007",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    },
    {
      "id":"Device1.tag0008",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    },
    {
      "id":"Device1.tag0009",
      "tsp":"2019-06-19 17:48:50",
      "val":"0.000000"
    }
  ],
}
```



```
"clientid":"gw1",
"time":"2019-06-19 17:48:51"
}
```

1. LMIoT.js: 因为生成的json格式中不包含质量戳, 所以需要在“数据过滤”中勾选“采集成功”, 生成的json格式如下:

```
{
  "clientid":"gw1",
  "devices":[
    {
      "deviceid":"Device1",
      "status":"1",
      "tags":{
        "tag0001":"2.000000",
        "tag0002":"3.000000",
        "tag0003":"1.230000",
        "tag0004":"0.000000",
        "tag0005":"0.000000",
        "tag0006":"0.000000",
        "tag0007":"0.000000",
        "tag0008":"0.000000",
        "tag0009":"0.000000"
      }
    }
  ],
  "seq":"3",
  "system":{
    "deviceid":"system",
    "tags":{
      "TIME_DAY":"19",
      "TIME_HOUR":"17",
      "TIME_MINUTE":"50",
      "TIME_MONTH":"6",
      "TIME_SECOND":"3",
      "TIME_WDAY":"3",
      "TIME_YEAR":"2019"
    }
  },
  "time":"1560937804",
  "type":"data"
}
```

1. Tlink.js: 因为生成的json格式中不包含质量戳, 所以需要在“数据过滤”中勾选“采集成功”, 生成的json格式如下:

```
{
  "sensorDatas":[
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0001"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0002"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0003"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0004"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0005"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0006"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0007"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0008"
    },
    {
      "addTime":"2019-06-19 17:51:15",
      "flag":"Device1.tag0009"
    }
  ]
}
```

查询

1. **default.js**: 只解析向查询主题发送如下json格式的数据，可以进行仪表下所有数据点的查询和指定数据点的查询。

```

{"operate":"read","devices":[
  {"deviceCode":"Device1","tags":null},
  {"deviceCode":"Device2",
   "tags":[
     {"tagCode":"tag0008"}
   ]
  }
]}

```

1. LMIoT.js: 只解析向查询主题发送如下json格式的数据，可以进行仪表下所有数据点的查询。

```

{
  "type": "inquire",
  "time": "1557136546",
  "seq": "230",
  "clientid": "gw1",
  "devices": [
    "Device1",
    "Device2"
  ]
}

```

查询返回

1. default.js: 查询结束后通过“查询数据发布的主题”发布如下json格式的数据:

```

{
  "Device1":{
    "tag0001":2,
    "tag0002":3,
    "tag0003":1.23,
    "tag0004":0,
    "tag0005":0,
    "tag0006":0,
    "tag0007":0,
    "tag0008":0,
    "tag0009":0
  },
  "clientid":"gw1",
  "time":"1560938379"
}

```

1. LMIoT.js: 查询结束后通过“查询数据发布的主题”发布如下json格式的数据:

```
{
  "clientid":"gw1",
  "devices":[
    {
      "deviceid":"Device1",
      "status":"1",
      "tags":{
        "tag0001":"2.000000",
        "tag0002":"3.000000",
        "tag0003":"1.230000",
        "tag0004":"0.000000",
        "tag0005":"0.000000",
        "tag0006":"0.000000",
        "tag0007":"0.000000",
        "tag0008":"0.000000",
        "tag0009":"0.000000"
      }
    },
    {
      "deviceid":"Device2",
      "status":"0",
      "tags":{
      }
    }
  ],
  "seq":"0",
  "time":"1560938617",
  "type":"inquireResp"
}
```

控制

1. Alink.js: 只解析向控制主题发送如下json格式的数据, 可以同时下发多个数据点的控制。

```
{
  "method": "thing.service.property.set",
  "id": "763382906",
  "params": {
    "Device1_tag0001": 0
  },
  "version": "1.0.0"
}
```

1. **default.js**: 只解析向控制主题发送如下json格式的数据，可以同时下发多个数据点的控制。

```
[
  {
    "operate": "write",
    "deviceCode": "Device1",
    "tagCode": "tag0001",
    "val": "10"
  },
  {
    "operate": "write",
    "deviceCode": "Device2",
    "tagCode": "tag0002",
    "val": "20"
  }
]
```

1. **defaultEx.js**: 和**default.js**的区别在于解析的json中每个数据点都包含顺序号seq，有于控制返回相应的顺序号，如下所示。

```
[
  {
    "operate": "write",
    "seq": 1,
    "deviceCode": "Device1",
    "tagCode": "tag0001",
    "val": "10"
  },
  {
    "operate": "write",
    "seq": 1,
    "deviceCode": "Device2",
    "tagCode": "tag0002",
    "val": "20"
  }
]
```

1. **LMIoT.js**: 只解析向控制主题发送如下json格式的数据，包含顺序号，对 **clientid** 进行验证判断是否进行数据点的控制。可以同时下发多个数据点的控制。

```
{
  "type": "write",
  "time": "1557136546",
  "seq": "230",
  "clientid": "gw1",
  "devices": {
    "deviceid": "Device1",
    "tagcode": "tag0001",
    "value": "63"
  }
}
```

1. **Tlink.js**: 只解析向控制主题发送如下json格式的数据，可以同时下发多个数据点的控制。其中 **flag** 即为GC中的数据点ID。

```

{
  "sensorDatas":[
    {
      "sensorsId":200302860,
      "value":"1111",
      "flag":"D1.tag0002"
    }
  ],
  "down":"down"
}

```

控制返回

1. **default.js**: 和上述控制主题中**default.js**配合使用，每操作结束一个数据点，通过控制响应发布的主题发布如下如下json格式数据。

```

{
  "deviceCode":"Device1",
  "message":"写操作返回信息",
  "status":true/false,
  "tagCode":"tag0001",
  "value":"10"
}

```

1. **defaultEx.js**: 和上述控制主题中**defaultEx.js**配合使用，每操作结束一个数据点，通过控制响应发布的主题发布如下如下json格式数据。

```

{
  "deviceCode":"Device1",
  "message":"写操作返回信息",
  "seq":1,
  "status":true/false,
  "tagCode":"tag0001",
  "value":"10"
}

```

1. **LMIoT.js**: 和上述控制主题中**LMIoT.js**配合使用，每操作结束一个数据点，通过控制响应发布的主题发布如下如下json格式数据。

```
{
  "type": "write",
  "time": "1557136546",
  "seq": "230",
  "clientid": "gw1",
  "devices": {
    "deviceid": "Device1",
    "tagcode": "tag0001",
    "value": "63",
    "status": "true/false"
  }
}
```

重启

1. **default.js**: 向重启主题发布如下json格式是网关重启（GC运行工程时订阅到消息不会重启电脑）。

```
{
  "operate": "reboot"
}
```

1. **LMIoT.js**: 向重启主题发布如下json格式，网关对clientid进行验证判断是否重启网关（GC运行工程时订阅到消息不会重启电脑）。

```
{
  "type": "reboot",
  "time": "1557136546",
  "seq": "230",
  "clientid": "gw1"
}
```

6.1.5 自定义模板

GC支持用户自定义模板，用于生成不同格式的实时数据、控制主题、重启主题等订阅到的数据的解析。在此以实时数据主题的模板为例，介绍如何新建自定义模板。具体操作如下：

1. **Mqtt**模板在配置工具安装目录下mqtt文件夹下，其中realtime文件夹为实时数据的模板目录；

2. 在realtime文件夹下新建一个js，在其中进行模板的编写；
3. 参照其他模板，如default.js:

```
(function() {  
    var timestamp = Math.round(new Date().getTime() / 1000)  
    result = { "clientid": base["Clientid"], "time": timestamp.toString() }  
    data.forEach(function(tag) {  
        if (!result[tagExt[tag.Id].deviceCode]) {  
            result[tagExt[tag.Id].deviceCode] = {};  
        }  
        result[tagExt[tag.Id].deviceCode][tagExt[tag.Id].tagCode] = parseFl  
    });  
    return JSON.stringify(result);  
})();
```

其中 (function() {

.....

})();

为主体，js需要写在其中；

data为经过过滤后的数据，格式为

```
[  
  {  
    "Id": "Device1.tag0001",  
    "Status": "Good",  
    "Timestamp": 1574926563,  
    "Val": "1.000000"  
  },  
  {  
    "Id": "Device1.tag0002",  
    "Status": "Good",  
    "Timestamp": 1574926563,  
    "Val": "3.000000"  
  }  
]
```

base为mqtt配置填写的参数；

tagExt为tag点的额外属性，包括采集页面中的名称（description）、tag点所在的设备名称、读写类型、tagID，还可以包含“高级参数”中填写的数据（“高级参数”需要填写json格式的数据）

```
{
  "Device1.tag0001":{
    "description":"",
    "deviceCode":"Device1",
    "objectType":"AV",
    "tagCode":"tag0001"
  },
  "Device1.tag0002":{
    "description":"",
    "deviceCode":"Device1",
    "objectType":"AV",
    "tagCode":"tag0002"
  }
}
```

模板最后返回主题需要上传的字符串；

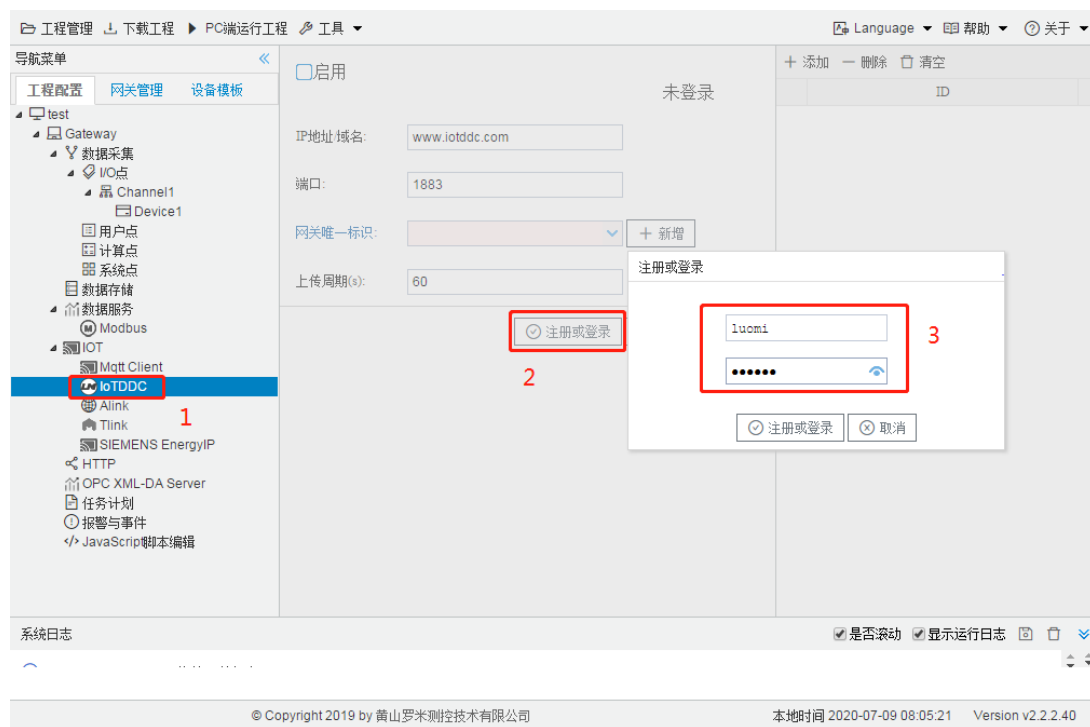
1. 模板编写完成之后，需要在配置工具中选择新建的模板；
2. 可以先用配置工具“PC端运行工程”测试模板能否正确运行；
3. 需要下载模板到网关：在“系统设置”中选择“开发者模式”，密码为“luomi”，选择“下载MQTT模板”，之后下载工程，网关就会根据新建的模板上传数据。



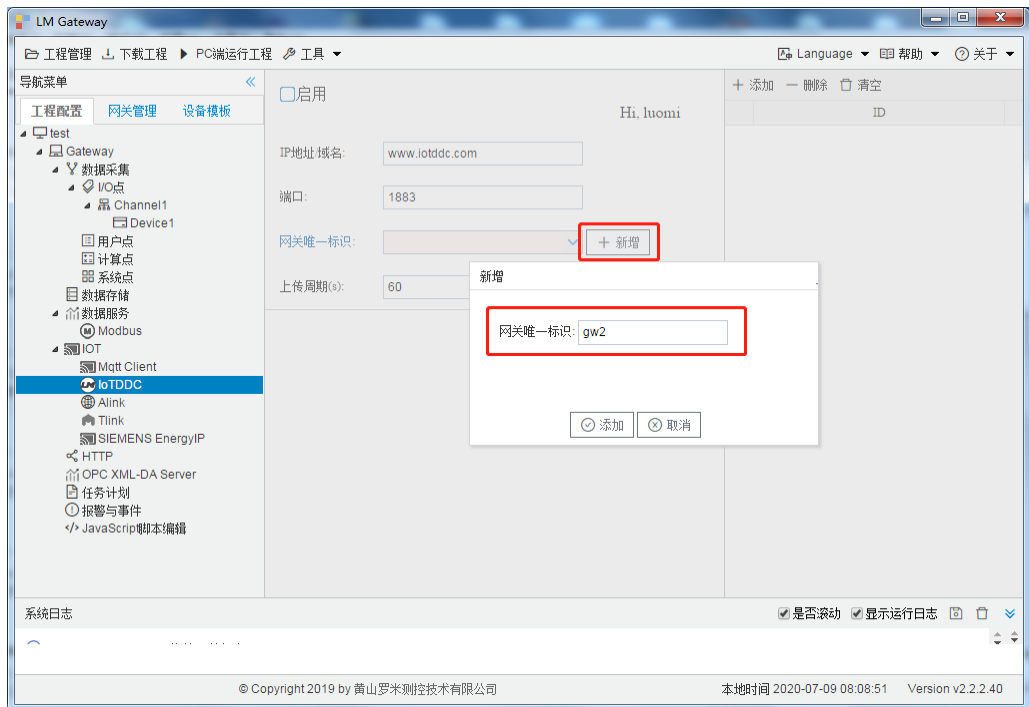
6.2 IoTDDC

应用于快速连接罗米测控IoT数据中心。

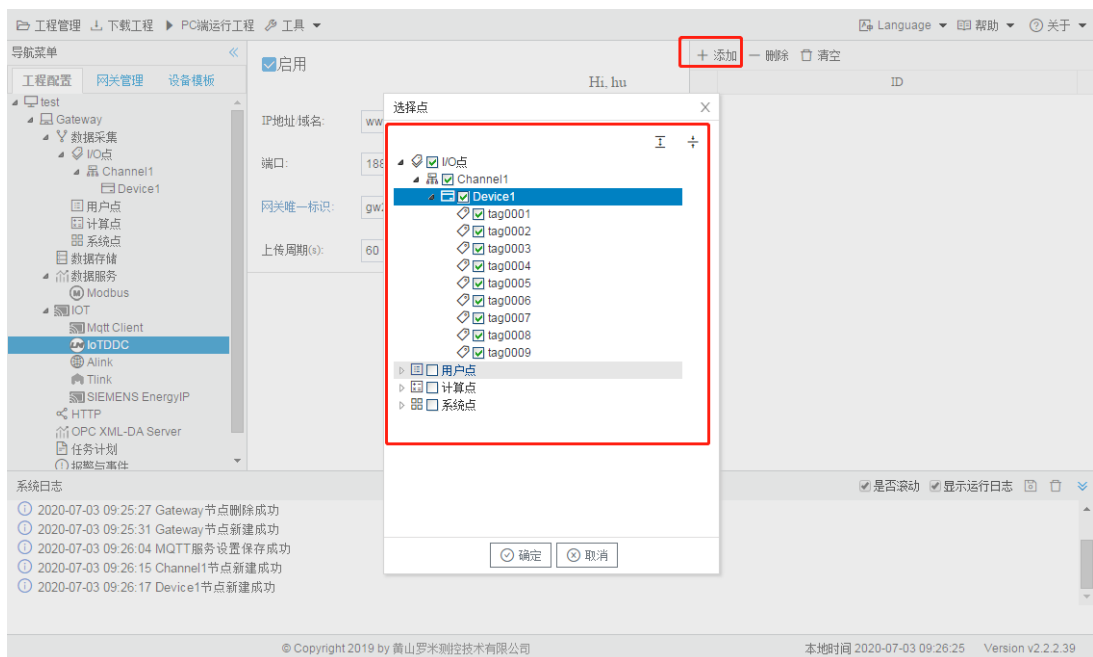
1. 在GC中注册或登录云平台；



1. 新增或者选择已有的网关唯一标识；



2. 点击“添加”按钮，完成需要上传的数据点的添加；



1. 点击“保存并上传平台”按钮，将该页面保存，并将整体工程上传到云端平台，进行下一步数据点含义；
2. 点击“下载工程”按钮将工程下载到网关当中；
3. 在浏览器中输入www.iotddc.com，登录用户为GC中注册的用户，查看网关上传的实时数据；



1. 用户可根据云平台开放的数据接口，将数据推送到云平台，获取实时数据、历史数据等；



6.3 Alink

Alink是阿里云定义的设备与云端之间的通信协议。**Alink**协议是针对物联网开发领域设计的一种数据交换规范，数据格式是**JSON**，用于设备端和物联网平台的双向通信，更便捷地实现和规范了设备端和物联网平台之间的业务数据交互。

在配置GC中Alink页面之前，需要在阿里云的物联网平台中进行以下步骤：

1.先在阿里云的物联网平台中创建产品。

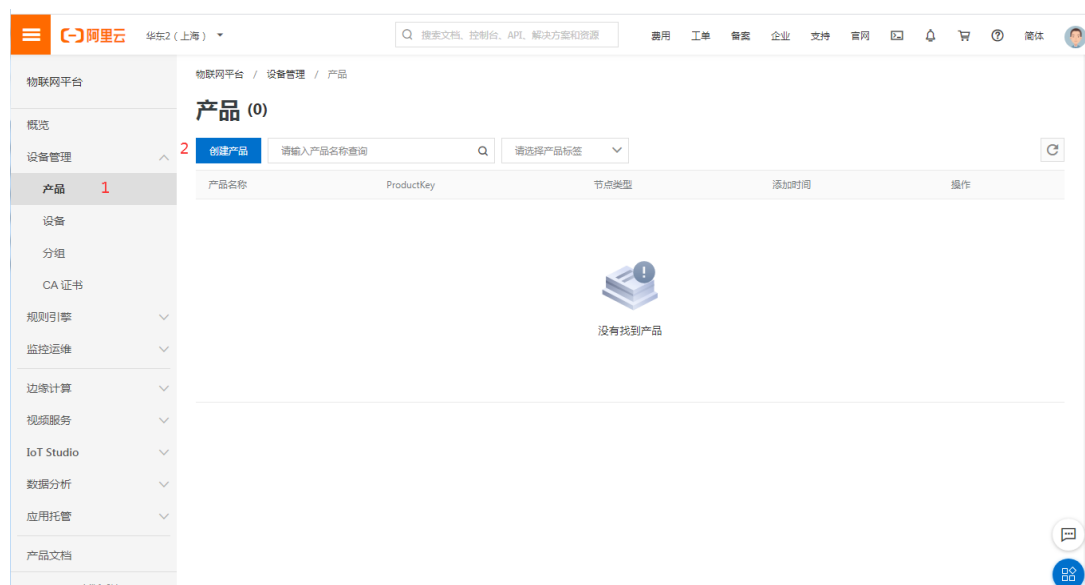
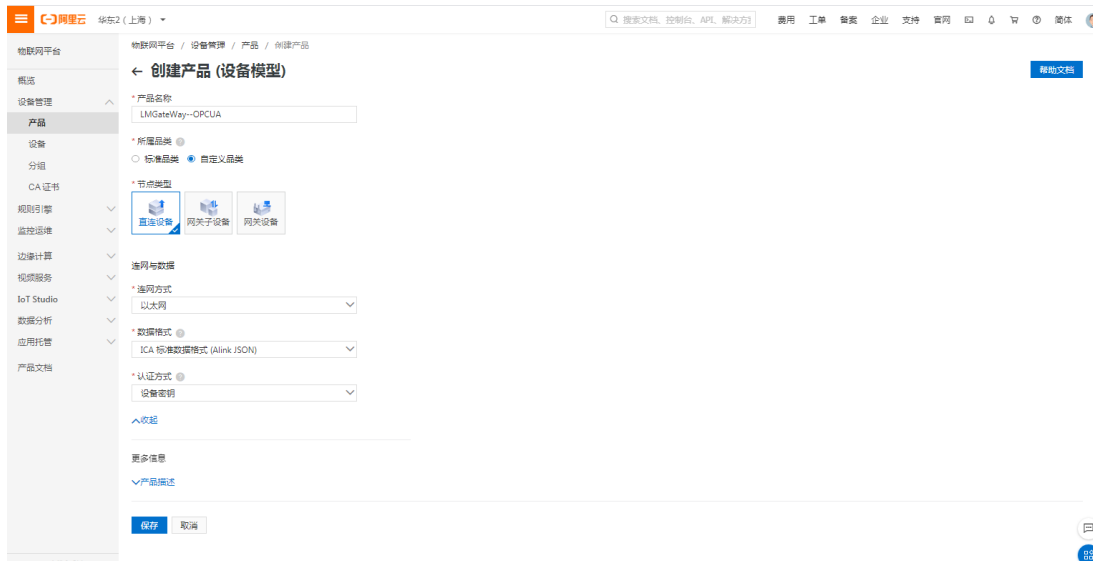


图6-11 产品

点击"设备管理"——"产品"——"创建产品"，出现以下画面。



创建产品

产品名称:自定义, 本实例中填写的是"LMGateWay--OPCUA"

所属品类: 选择"自定义品类"

节点类型: 选择"直连设备"

联网方式: 根据需求选择"WIFI"、"蜂窝(2G\3G\4G\5G)"、"以太网"中的一个; 本实例中选择是"以太网"

数据格式: 选择 "ICA标准数据格式 (Alink json) "

产品描述: 自定义, 可不填

点击"保存", 即可完成创建。

2.创建产品下属的设备。

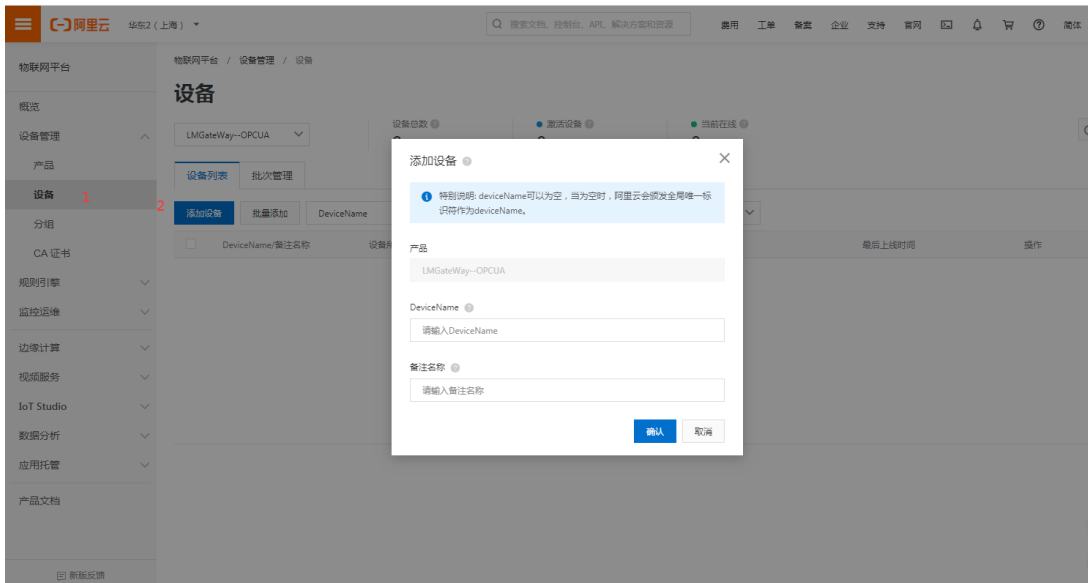


图6-13 添加设备

DeviceName: 自定义, 本实例中填写的是"OPCUA--TIME"

备注名称: 自定义, 本实例中填写的是"网关时间"

点击"确认"后, 出现以下弹框

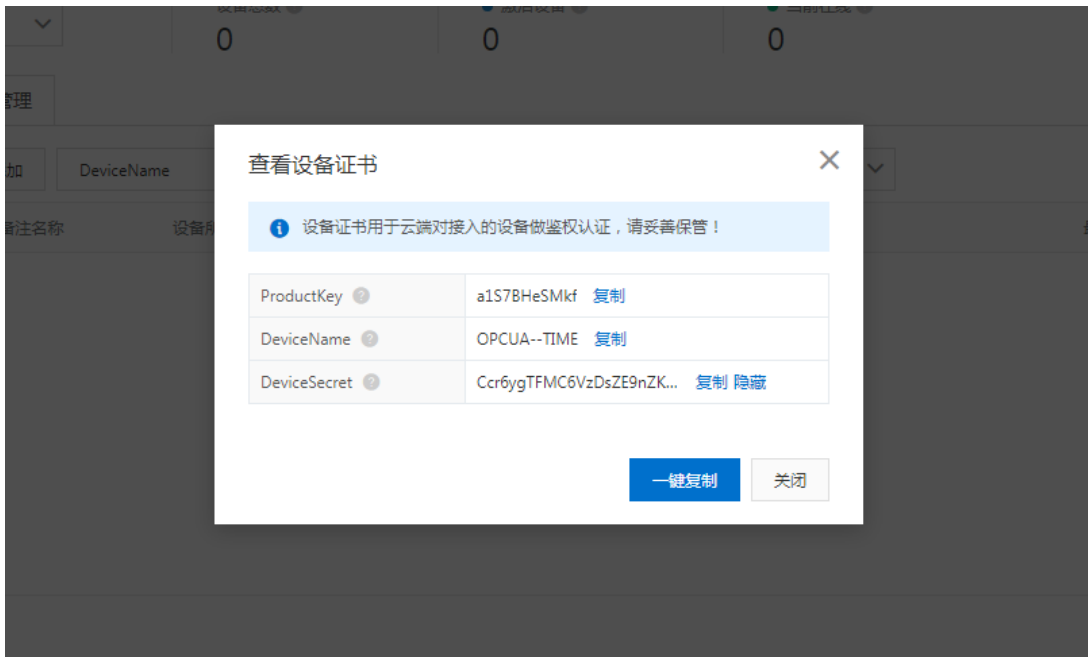


图6-13 设备证书

上图为该设备的“三元组”：ProductKey、DeviceName和DeviceSecret，三元组和实际的一个物理设备(网关)一一对应。万一现实中出现几个三元组一样的设备，那么后上网的设备，会让前一个设备下线。同时仅有一个这样的设备在线。

可以点击“一键复制”按钮将三元组自定义保存，也可点击设备列表中相应的设备查看三元组。

在阿里云中进行了上述步骤之后，在GC中进行Alink页面的配置，具体步骤如下：

1.勾选“启用”按钮；

填写地域（[地域和可用区](#)）和三元组；

clientid可默认为12345，不需要修改；

上传周期自定义。

点击“保存”按钮即可完成Alink的连接配置；

2.点击“添加”按钮，完成需要上传的数据点的添加。

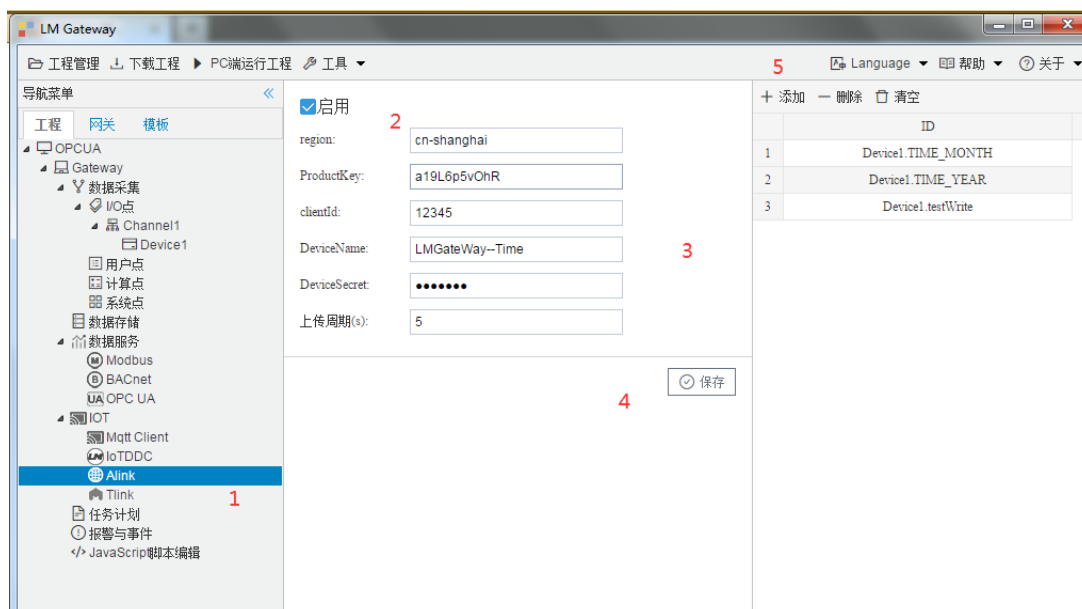


图6-15 Alink连接配置

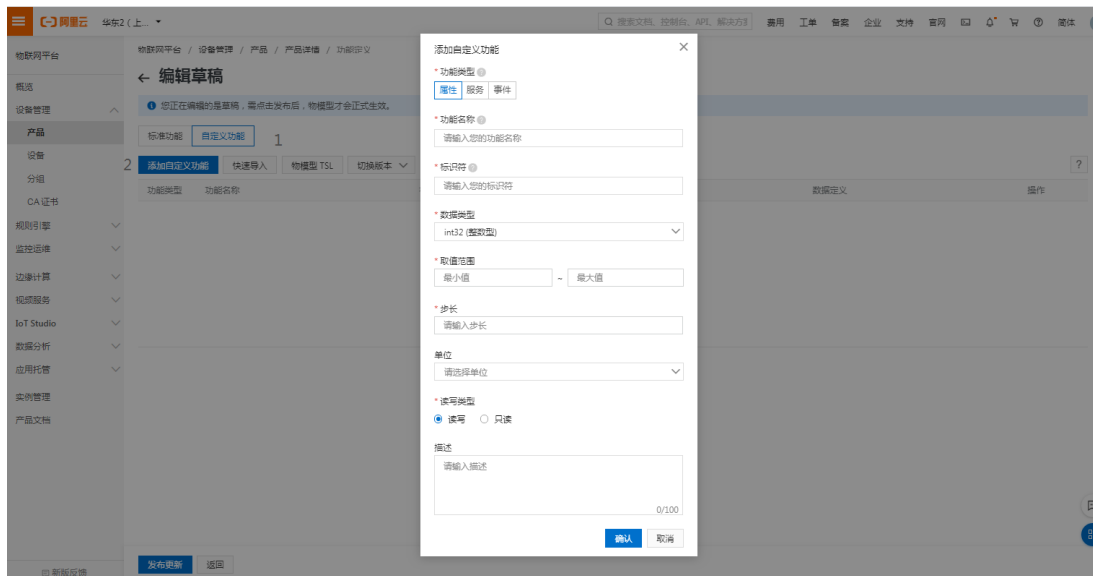
网关配置完成之后，接下来配置阿里云物联网。回到浏览器页面。



功能定义

依次点击"产品"---"LMGateway--OPCUA"(根据实际创建名称)---"功能定义"---"自定义功能"---最后点击"编辑草稿"。(确定右上角不处于发布状态，否则无法显示"编辑草稿"这一功能)

点击"自定义功能"-----"添加自定义功能"



配置tag点

功能名称：自定义填写

数据类型：根据采集值的类型进行选择，实例中选择 float

取值范围：根据实际情况填写

步长：可默认为1.(例如 数组中{1, 2, 3, 4}，步长为1，{1, 3, 5, 7}步长为2)

单位：根据实际情况填写

读写类型：根据实际填写

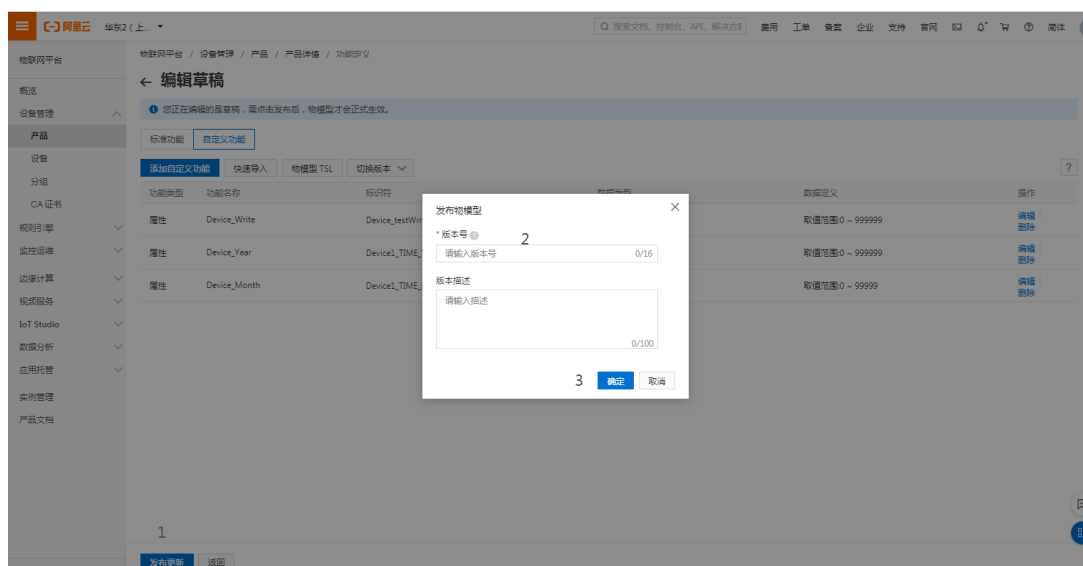
标识符：属性唯一标识符，在产品中具有唯一性。需要设置为："设备名称" + "_" + "tagID",如下图中（节选图2-2添加tag点）所示，只需将"Device1.TIME_MONTH"更改为"Device1_TIME_MONTH"即可。

因为Alink中 设备与tag点之间的连接为 "_",而网关中则为"."

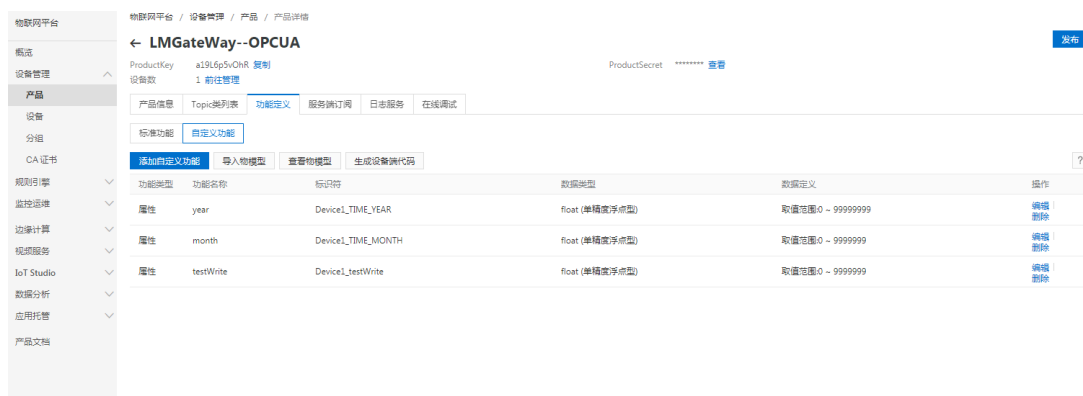
| ID |
|--------------------|
| 1 |
| Device1.TIME_MONTH |
| 2 |
| Device1.TIME_YEAR |
| 3 |
| Device1.testWrite |

网关tag点显示

添加完成后，点击"发布更新"---输入“版本号”（自定义）---点击"确定"



下图即为配置完成的样子。



Alink配置完成

6.4 Tlink

用户可在Tlink页面中配置连接Tlink物联网平台。

在配置GC中Tlink页面之前，需要在Tlink的物联网平台中进行以下步骤：

1.在Tlink控制台中添加设备（设备即为网关），添加设备下的传感器。

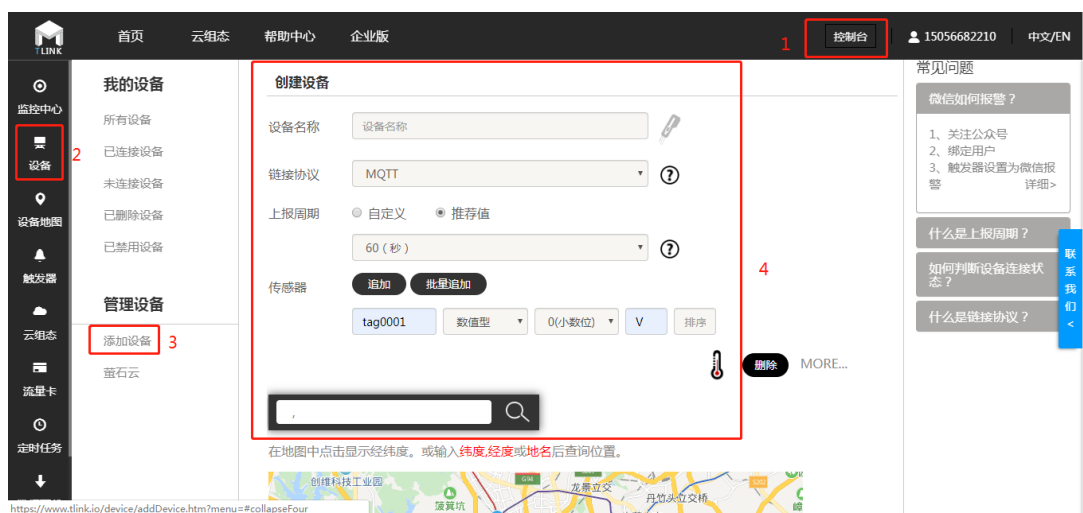


图6-16 Tlink添加设备

链接协议：需要选择“MQTT”。

上报周期：需要设置大于在GC的Tlink页面设置的上传周期。

传感器：网关采集的数据点。

2.在设备页面点击“设置连接”按钮，可以看到该设备的序列号和设备下的所有传感器。



图6-17 Tlink设置连接

在Tlink物联网平台中进行了上述步骤之后，在GC中进行Tlink页面的配置，具体步骤如下：

- 1.勾选“启用”按钮，填写Tlink物联网平台设备信息中的IP和序列号，上传周期自定义。点击“保存”按钮即可完成Tlink的连接配置；



图6-18 Tlink连接配置

- 2.点击“添加”按钮，完成需要上传的数据点的添加。
- 3.将Tlink页面中生成的传感器ID填写到对应的数据点表格当中。

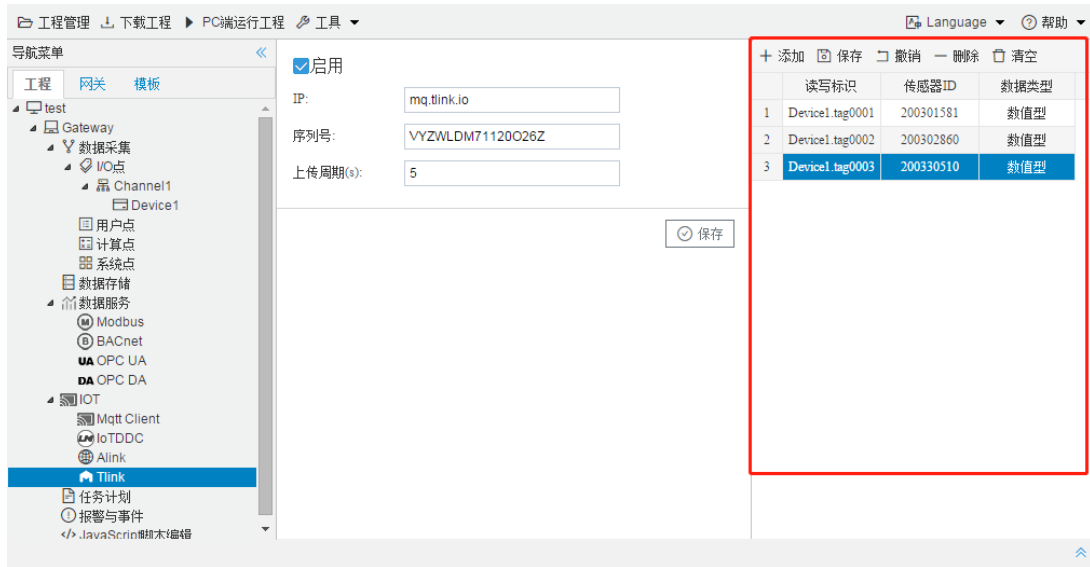


图6-19 上传点配置

第七章 任务计划

用户建立任务计划，指定星期和时间段设置多个点的值。

在使用任务计划功能前，请先校准网关的时间。

具体操作如下：

1. 单击“添加”按钮；
2. 在弹框中添加唯一的任务计划名称，设置起始时间、结束时间和星期，必须保证起始时间小于结束时间。在任务计划的时间段内保持并设置数据点的值，如果发现数据点值不是设置的值，发送写操作命令给采集驱动；
3. 点击“+”按钮新增Tag点添加文本框和写入值的文本框；
4. 点击“Add”按钮，在弹出的“选择点”窗口中选择一个需要写入值的数据点；
5. 在写入值文本框中填入需要写入的值；
6. 可重复3,4,5的步骤进行多个点的写入；
7. 点击“确定”按钮完成任务计划的添加。

如图7-1所示。



图7-1 任务计划配置

如果想修改已经添加的任务计划，只需要双击需要修改的任务计划，就会弹出上图的任务计划的窗口。重复上述步骤完成任务计划的修改。

第八章 报警与事件

报警与事件页面允许用户设置事件的触发条件，当满足条件时触发事件，当状态从满足条件转变为不满足条件时执行事件解除。

可以添加、修改、删除事件。

- 添加：弹出事件编辑页面，新建一个事件。
- 修改：双击需要修改的事件，弹出事件编辑页面，修改选中的事件。
- 删除：删除选中的事件。

8.1 设置报警条件

报警与事件按照报警条件分为“点值超出范围”和“质量不为Good”两种类型，用户根据需要建立相应的报警事件。

报警事件触发和报警事件解除记录都会保存到数据库。当数据库大小超过5M时，网关缓存新的一条报警记录时，删除最早的一条报警记录。

事件编辑步骤如下：

1. 单击“添加”按钮；
2. 编辑唯一的事件名称；
3. 编辑“事件来源”：通过选择“事件种类”进一步填写事件参数
 - 间隔时间：一次事件触发后，在间隔时间内不再触发相同事件。
 - 关联点名称：为I/O点、用户点、计算点和系统点中的一个点。
 - 最大值、最小值：关联点值的范围，最小值 \leq 当前值 \leq 最大值为正常情况，当最小值 $>$ 当前值或者最大值 $<$ 当前值都会触发报警。
 - 抖动时间：当点值超过范围或质量不为good持续的时间小于抖动时间，此时不触发事件。
4. 编辑“当事件触发”：添加事件触发时写入的点和值，也可以“在处理动作选择”中选择“不响应”。
5. 编辑“当事件解除”：添加事件解除时写入的点和值，也可以“在处理动作选择”中选择“不响应”。
6. 点击“确定”按钮完成事件的添加。

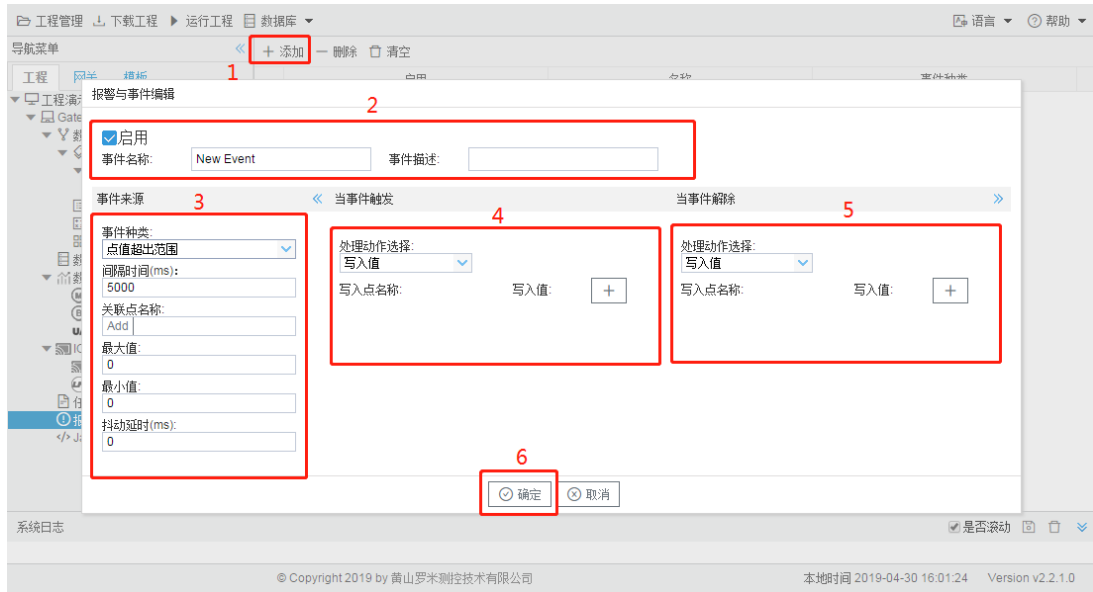


图8-1 事件编辑

第九章 JavaScript

JavaScript脚本编辑器内置功能函数，用户可以通过编辑脚本语言实现自定义逻辑控制。

9.1 操作步骤

具体操作如下：

1. 单击“添加”按钮；
2. 在弹出的编辑框中添加唯一的js名称，执行方式总共有三种：开机运行、循环和定时。
 - 开机运行：程序启动时运行JS脚本；
 - 循环：根据配置的执行周期（ms）循环执行JS脚本；
 - 定时：每天规定的时间（时：分：秒）执行JS脚本；
3. 单击选中需要运行的函数；
4. 点击“Add”按钮，在弹窗中选择一个需要读取或者设置的点；
5. 单击“插入”按钮，将新建的函数添加到js编辑框的光标后面；
6. 可重复3,4,5的步骤进行多个函数的添加；
7. 点击“确定”按钮完成js脚本的编辑。

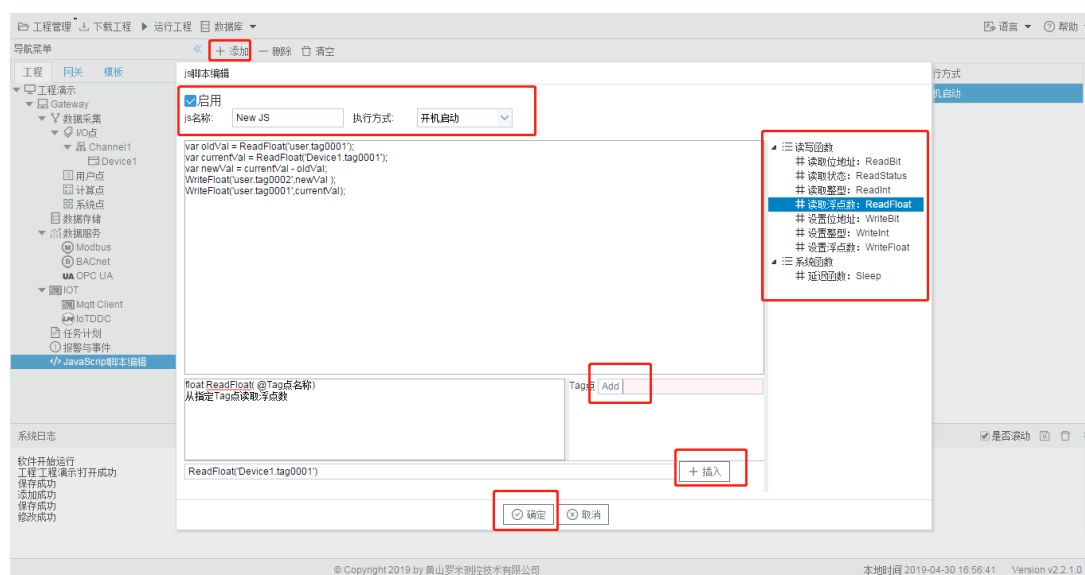


图9-1 JavaScript脚本编辑步骤

用户可以在编辑框中自行添加符合js语法的脚本，如while、if、else等。

9.2 函数说明

JS 脚本编辑器内置的函数说明如下：

读写函数

1. **ReadBit**从指定Tag点读取Boolean，输入参数为Tag点名称的字符串，如果该点不存在或者质量戳不为Good，返回undefined。

语法：

```
var tagValueBit = ReadBit('Device1.tag0001');
```

1. **ReadStatus**从指定Tag点读取状态，输入参数为Tag点名称的字符串，返回为Boolean，True为采集成功，False为采集失败，如果该点不存在或者质量戳不为Good，返回undefined。

语法：

```
var tagStatus = ReadStatus('Device1.tag0001');
```

1. **ReadInt**从指定Tag点读取整数，输入参数为Tag点名称的字符串，如果该点不存在或者质量戳不为Good，返回undefined。

语法：

```
var tagValueInt = ReadInt('Device1.tag0001');
```

1. **ReadFloat**从指定Tag点读取浮点数，如果该点不存在或者质量戳不为Good，返回undefined。

语法：

```
var tagValueFloat = ReadFloat('Device1.tag0001');
```

1. WriteBit向指定Tag点写入整型，输入参数为Tag点名称的字符串，和需要设置该Tag点的值JS的Boolean类型，无返回值。

语法:

```
var value = true;

WriteBit('Device1.tag0001', value);
```

1. WriteInt向指定Tag点写入整型，输入参数为Tag点名称的字符串，和需要设置该Tag点的值的JS的Number类型，无返回值。

语法:

```
var value = 10;

WriteInt('Device1.tag0001', value);
```

1. WriteFloat向指定Tag点写入浮点型，输入参数为Tag点名称的字符串，和需要设置该Tag点的值的JS的Number，无返回值。

语法:

```
var value = 1.23;

WriteFloat('Device1.tag0001', value);
```

系统函数

1. Sleep延迟函数，输入参数值为毫秒，无返回值。

语法:

```
Sleep(1000); //延迟等待1000ms
```

9.3 使用示例

开机运行示例

```
while (true) {
    var status;
    status = ReadStatus("system.TIME_SECOND"); //ReadStatus: 读取数据点质量戳
    console.log("system.TIME_SECOND status is ", status);
    if(status){ // 如果该点存在并且质量戳为Good
        var curVal;
        curVal = ReadInt("system.TIME_SECOND"); //ReadInt: 从指定数据点读取整数
        console.log("system.TIME_SECOND value is ", status);
        WriteInt("user.test",curVal); //WriteInt: 向指定数据点写入整数
    }
    Sleep(1000); //休眠1秒钟
}
```

循环示例

```
var status;
status = ReadStatus("system.TIME_SECOND"); //ReadStatus: 读取数据点质量戳
console.log("system.TIME_SECOND status is ", status);
if(status){
    var curVal;
    curVal = ReadInt("system.TIME_SECOND"); //ReadInt: 从指定数据点读取整数
    console.log("system.TIME_SECOND value is ", status);
    WriteInt("user.test",curVal); //WriteInt: 向指定数据点写入整数
}
```

定时示例

```
var status;
status = ReadStatus("system.TIME_SECOND"); //ReadStatus: 读取数据点质量戳
console.log("system.TIME_SECOND status is ", status);
if(status){
    var curVal;
    curVal = ReadInt("system.TIME_SECOND"); //ReadInt: 从指定数据点读取整数
    console.log("system.TIME_SECOND value is ", status);
    WriteInt("user.test",curVal); //WriteInt: 向指定数据点写入整数
}
```


第十章 运行工程

GC运行

工程配置完成之后，用户可以选择运行工程：点击工具栏当中的“运行工程”按钮，以PC为载体，使用PC的串口和网口进行数据采集和数据服务。在系统日志栏中会打印程序运行时输出的日志和报文，便于用户在PC环境中调试项目。

在GC运行工程时，可以通过点击通道下的设备，查看该设备的实时数据。

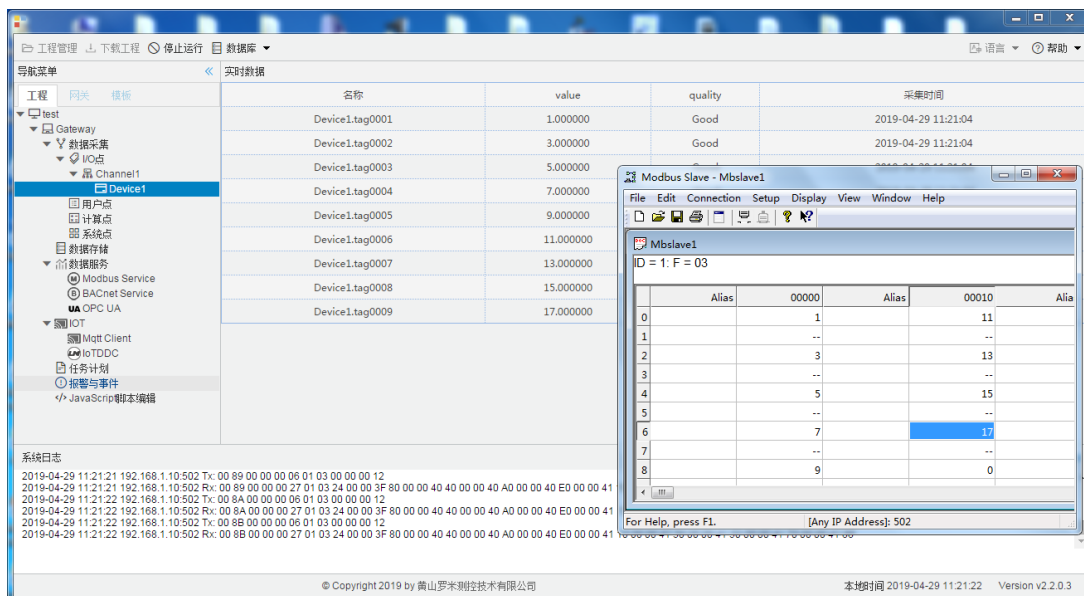


图10-1 软件网关运行时

LMGateway运行

工程配置完成之后，通过工具栏中的"下载工程"按钮将工程下载到LMGateway当中。下载成功之后，LMGateway会根据新下载的工程运行程序。

第十一章 模板

GC具有保存模板和装载模板的功能，用户可选择驱动新建自己的设备模板。

11.1 新建模板

点击工具栏中的“新建模板”按钮，在弹出框中填写唯一的模板名称，选择模板的驱动协议。

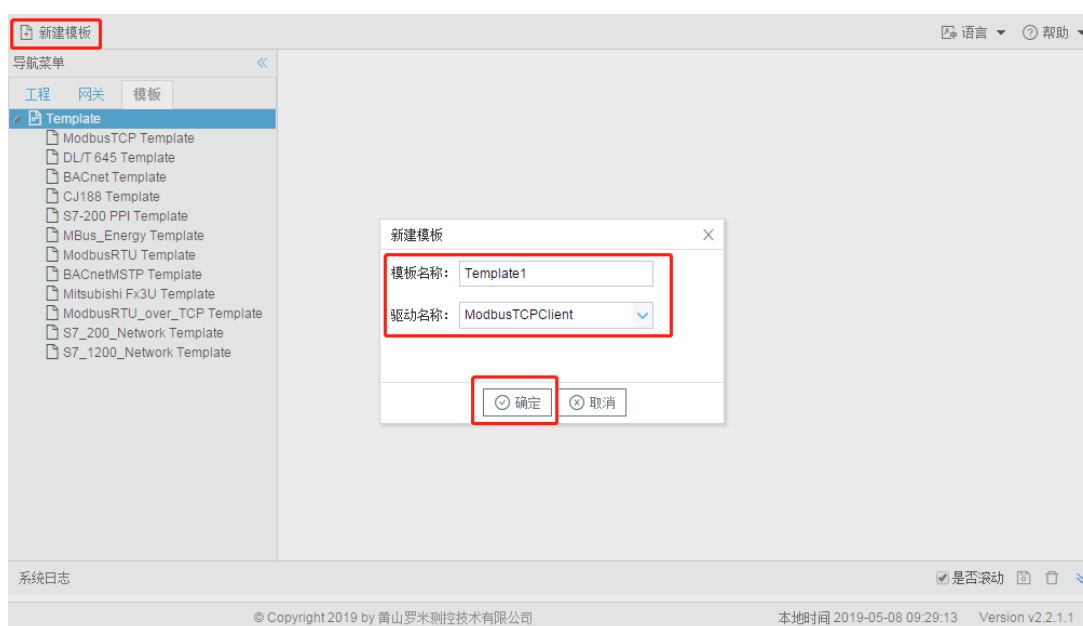


图11-1 新建模板

单击选中左侧模板，编辑模板电表，具体操作详见“驱动通讯说明文档”。

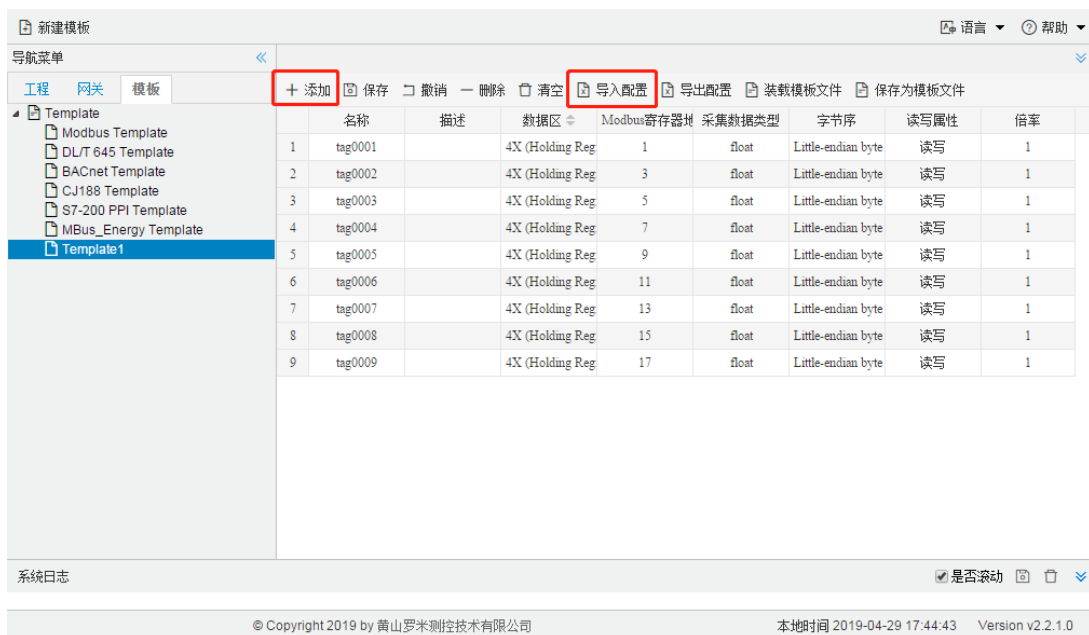


图11-2 编辑模板

11.2 修改模板名称

双击需要修改名称的模板，在弹出框中修改模板名称，点击“确定”按钮。

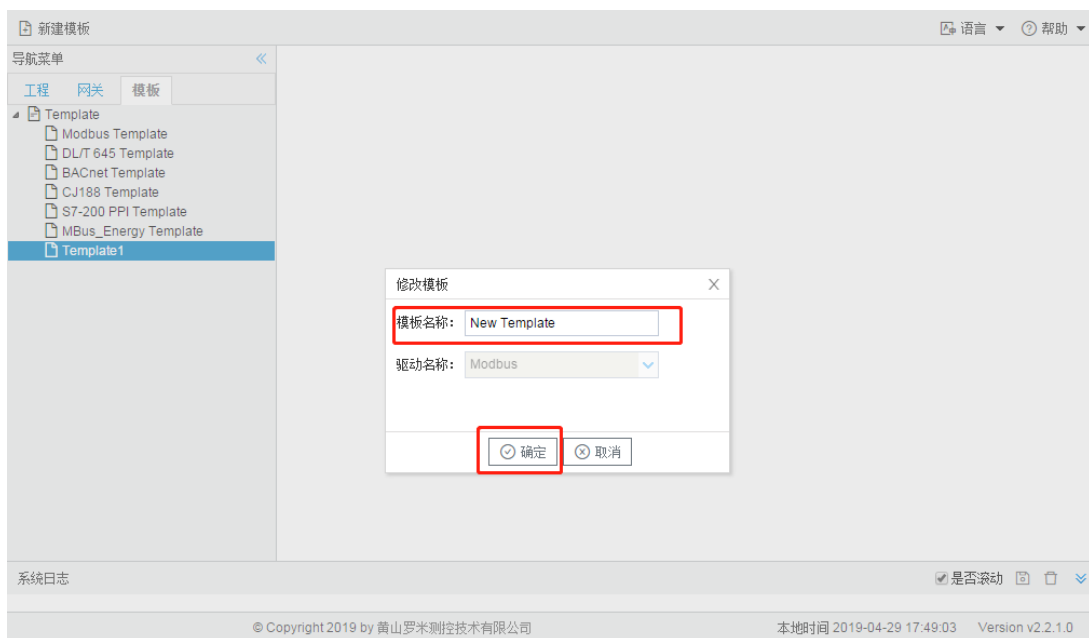


图11-3 修改模板名称

11.3 删除模板

单击选中需要删除的模板，右键选择“删除模板”。

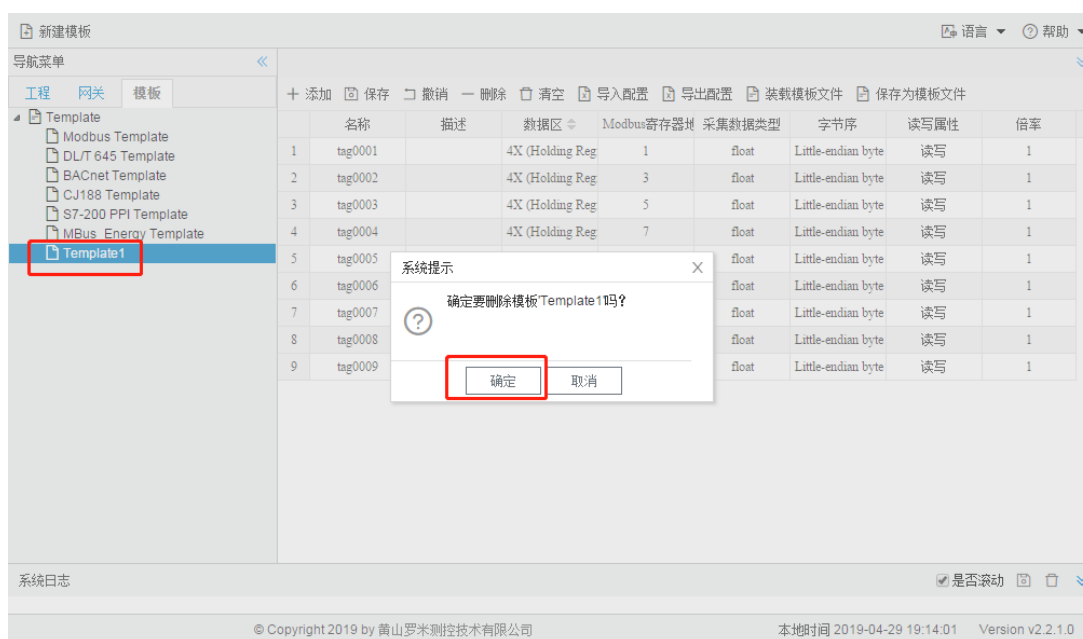


图11-4 删除模板

第十二章 应用实例

下面以工程名称为“demo”为例，介绍GC从无到有配置工程中采集协议和服务的过程。

12.1 新建工程

打开GC，单击左上方的“工程管理”按钮，选择“新建工程”，在弹出的“新建工程”对话框中输入工程名称，如此处的"demo"，单击“确定”按钮。如图12-1所示。



图12-1 新建工程

此时，GC就会将该工程所包含的内容直接呈现在导航菜单栏中，根节点的名称即为工程名称。此后的工程配置都在根节点之上进行。

12.2 新建网关设备

右键单击“demo”根节点，选择“添加网关”，输入网关名称，选择需要配置的网关类型，单击“确定”按钮，如图12-2所示。

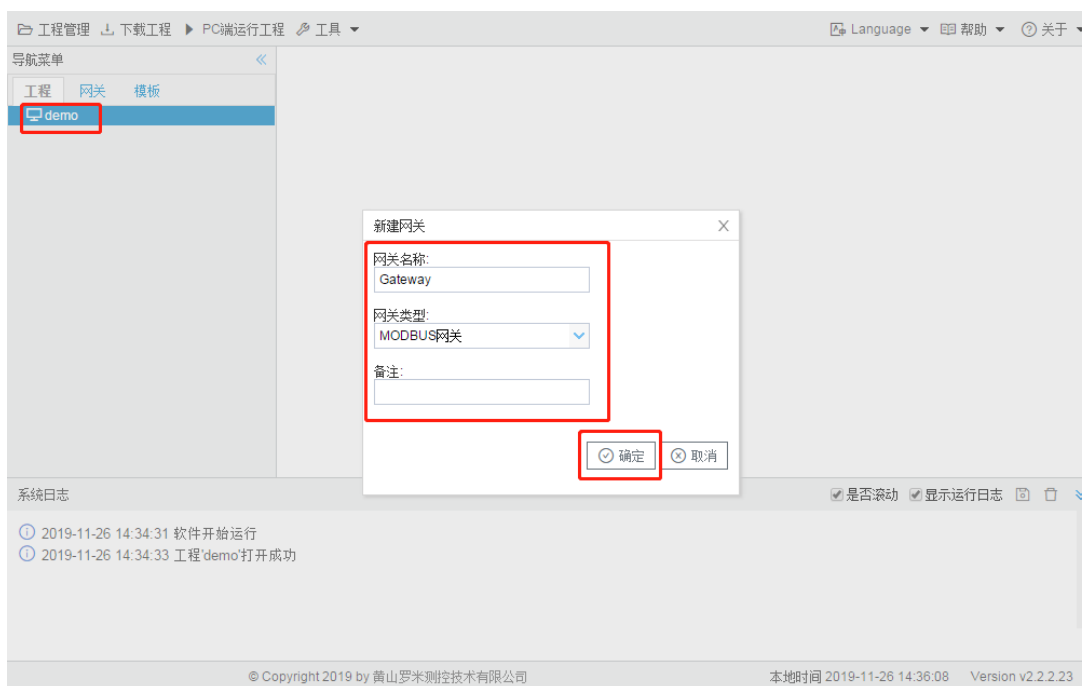


图12-2 新建网关设备

此时我们就可以在“demo”根节点下，看到该网关类型所有的可供配置项。

其中“数据采集”为数据来源，分为I/O点（采集驱动）、用户点、计算点和系统点；

“数据存储”是将需要存储的数据点，按照设定周期存储到TF卡当中；

“数据服务”是该网关类型提供的服务类型：例如上图选择“MODBUS网关”，此处就会显示“Modbus”服务。

“任务计划”、“报警与事件”和“JavaScript脚本编辑”的说明，详见第七、八、九章节。



图12-3 工程树

12.3 新建通道

右键单击“I/O点”标签，选择“新建通道”，输入通道名称，选择通道类型（串口/网口），选择采集驱动类型，这里我们选择“ModbusTCPClient”协议作为演示协议。通道的其他参数详见《罗米测控网关通讯协议说明》。如图12-4所示。

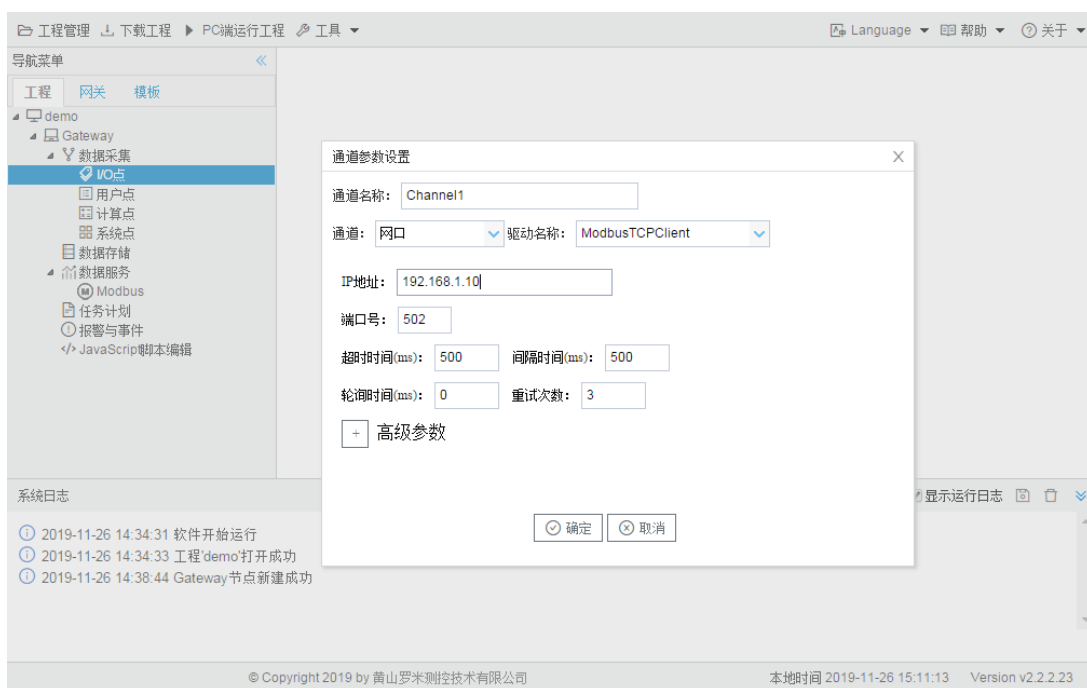


图12-4 新建通道

12.4 新建设备

选择当前新建的通道，右键单击选择“新建设备”，输入设备名称和设备地址，如图12-5所示。

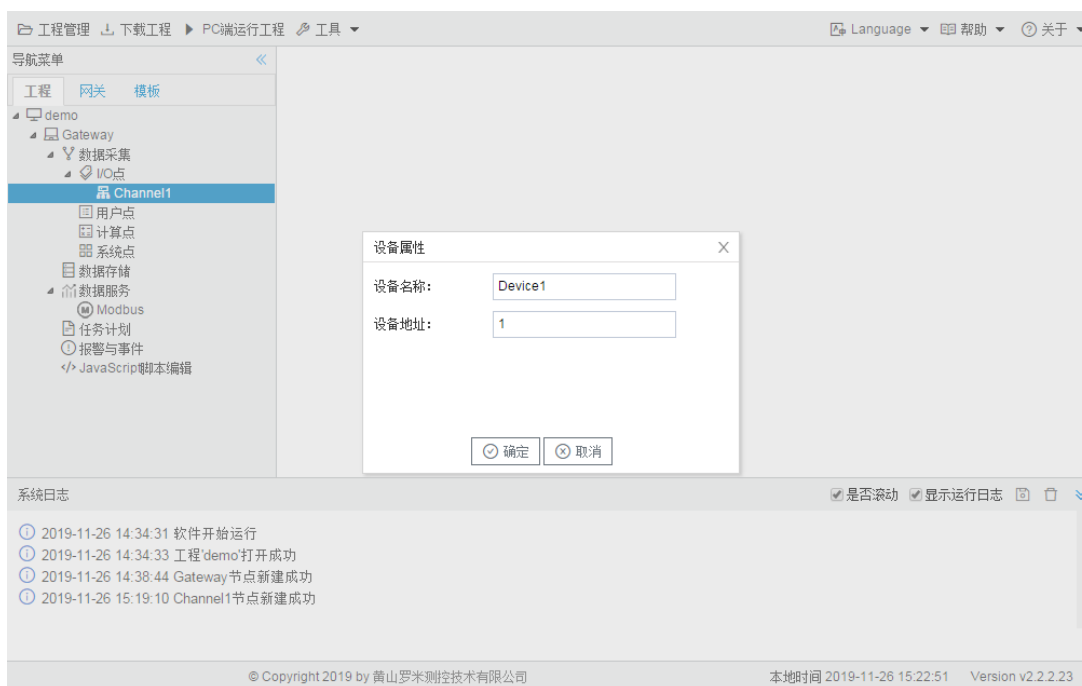


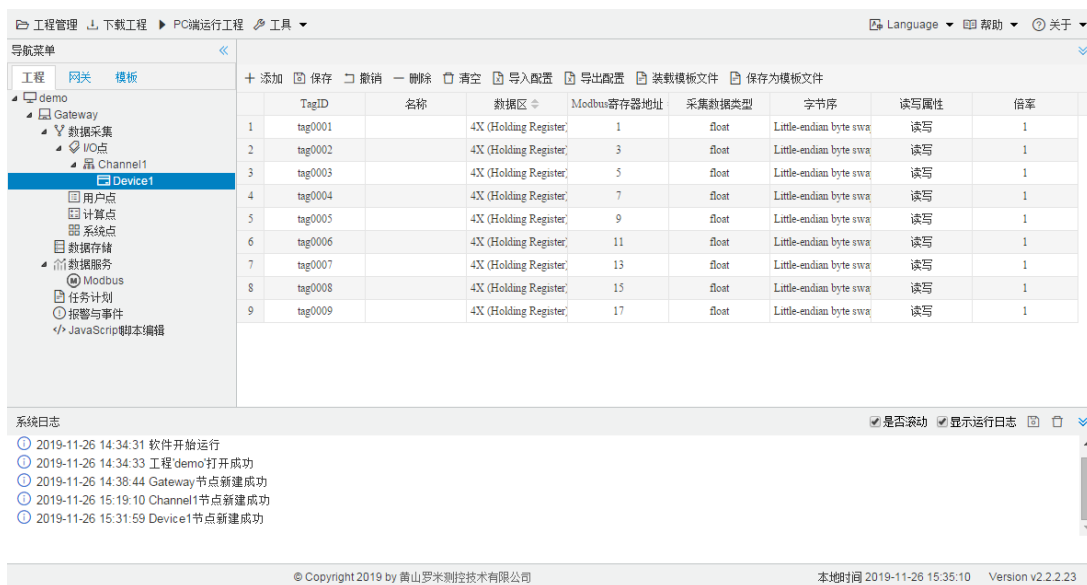
图12-5 新建设备

12.5 添加数据点

左键单击选择需要新建数据点的设备，右侧会显示该设备下已编辑的数据点，可以在表格当中进行增删改。

可通过“导出配置”和“导入配置”将当前配置导出成Excel，用Excel软件配置完成后导入到配置工具当中；

可通过“保存为模板文件”和“装载模板文件”将当前配置保存成模板文件，在当前工程或者其他工程中可直接导入保存的模板，方便批量配置。



| TagID | 名称 | 数据区 | Modbus寄存器地址 | 采集数据类型 | 字节序 | 读写属性 | 倍率 |
|-------|---------|-----------------------|-------------|--------|------------------------|------|----|
| 1 | tag0001 | 4X (Holding Register) | 1 | float | Little-endian byte swa | 读写 | 1 |
| 2 | tag0002 | 4X (Holding Register) | 3 | float | Little-endian byte swa | 读写 | 1 |
| 3 | tag0003 | 4X (Holding Register) | 5 | float | Little-endian byte swa | 读写 | 1 |
| 4 | tag0004 | 4X (Holding Register) | 7 | float | Little-endian byte swa | 读写 | 1 |
| 5 | tag0005 | 4X (Holding Register) | 9 | float | Little-endian byte swa | 读写 | 1 |
| 6 | tag0006 | 4X (Holding Register) | 11 | float | Little-endian byte swa | 读写 | 1 |
| 7 | tag0007 | 4X (Holding Register) | 13 | float | Little-endian byte swa | 读写 | 1 |
| 8 | tag0008 | 4X (Holding Register) | 15 | float | Little-endian byte swa | 读写 | 1 |
| 9 | tag0009 | 4X (Holding Register) | 17 | float | Little-endian byte swa | 读写 | 1 |

图12-6 添加数据点

协议表格的参数说明详见《罗米测控网关通讯协议说明》。

12.6 数据服务

单击选择“数据服务”下的“Modbus”，在右侧弹出页面中进行Modbus服务的配置。

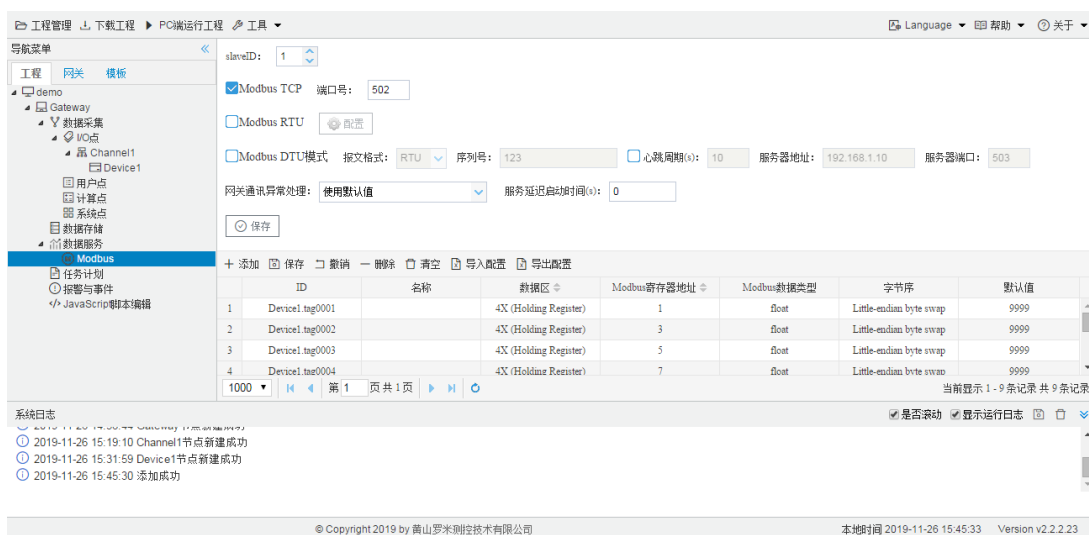


图12-7 Modbus服务

在Modbus服务中，上方是Modbus服务的协议配置，下方为映射到Modbus服务上的数据点，映射的数据区、寄存器地址、数据类型以及字节序。

本示例为Modbus网关，在12-2章节“新建网关设备”时选择其他类型的网关，数据服务会有所区别，但是配置思路都是一致的，需要先进行协议方面的配置，然后进行数据点的映射（需要上传的数据点、数据点映射到的地址等）。

12.7 应用工程

工程配置完成后，可以进行两方面的操作：下载工程和PC端运行工程。

下载工程：将当前工程下载到指定的网关当中，使网关以当前工程配置运行；

PC端运行工程：在PC端运行当前工程，使用PC的串口和网口进行数据采集和数据服务。在系统日志栏中会打印程序运行时输出的日志和报文，便于用户在PC环境中调试项目。

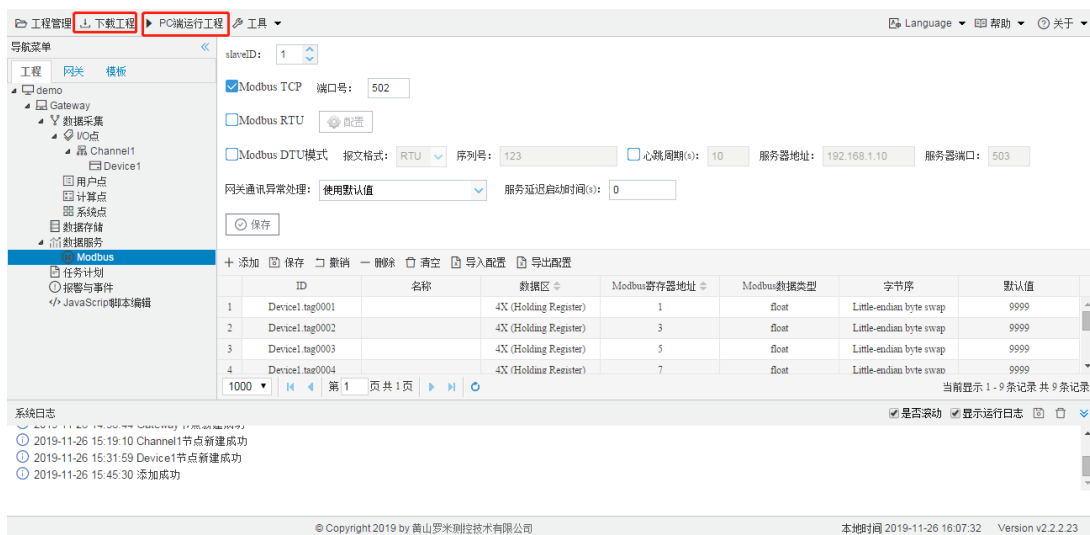


图12-8 应用工程

第十三章 附录

网关采集仪表数据时，RS485通讯线手拉手正常接法：

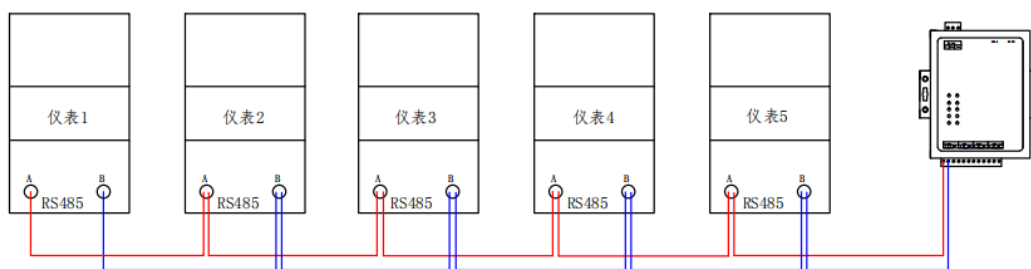


图 13-1 手拉手接线