MODBUS 智能网关用户手册

http://www.brimesh.com

目录

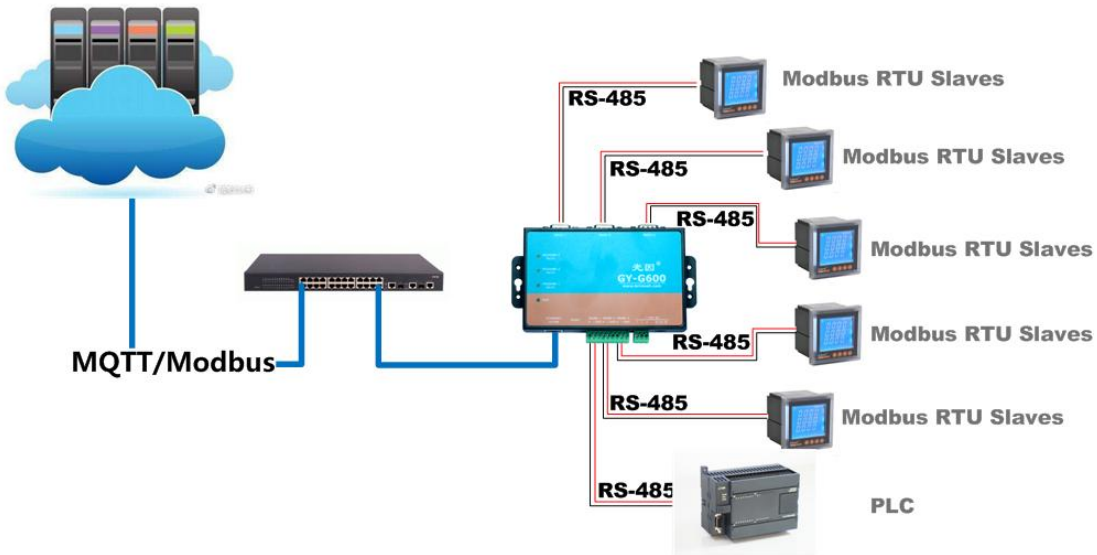
1 产品简介

1.1 概述

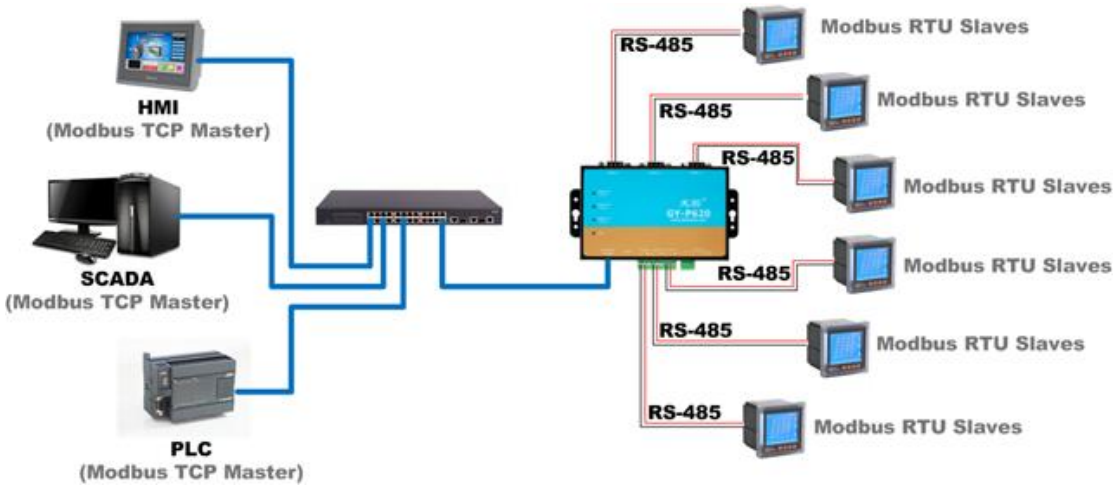
Modbus 智能网关是我们新推出的产品，专门针对 PLC，HMI 或者组态软件的采集。传统的 Modbus 网关只是 Modbus TCP 和 Modbus RTU 协议做个转换，网关本身并不主动采集数据，而我们的采集网关是可以主动采集从站数据并缓存到网关内部，从而上位机可以通过 TCP 或者 RS-485 总线直接从我们采集网关取数据，这样就减少了上位机主站与从站的交互次数，极大地提高了采集速度，实时性，可靠性以及稳定性；对于多 RS-485 主站想要访问相同的从站数据，传统的 Modbus 网关是无法解决的，而由于我们的网关是自动采集并存储数据，因此完全支持多主站同时通过 RS-485 总线读取数据；另外，针对一些没有以太网口的 PLC 或者 HMI，也可以通过我们的 Modbus 智能网关的 RS-232/RS-485 端口，采集到 Modbus TCP 从站的数据。

1.2 应用场景

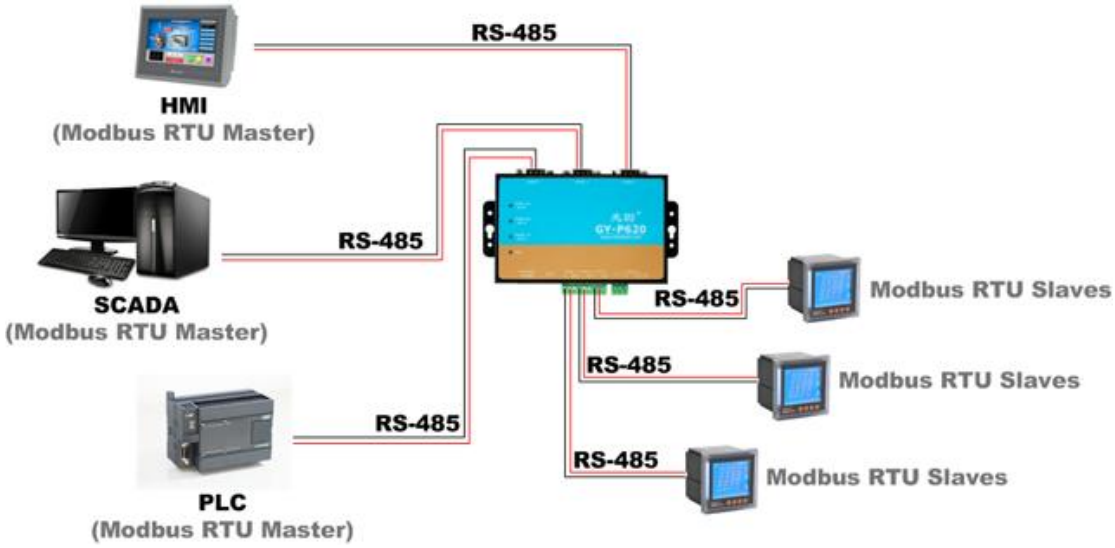
- **通过 MQTT/Modbus 连接到云端**



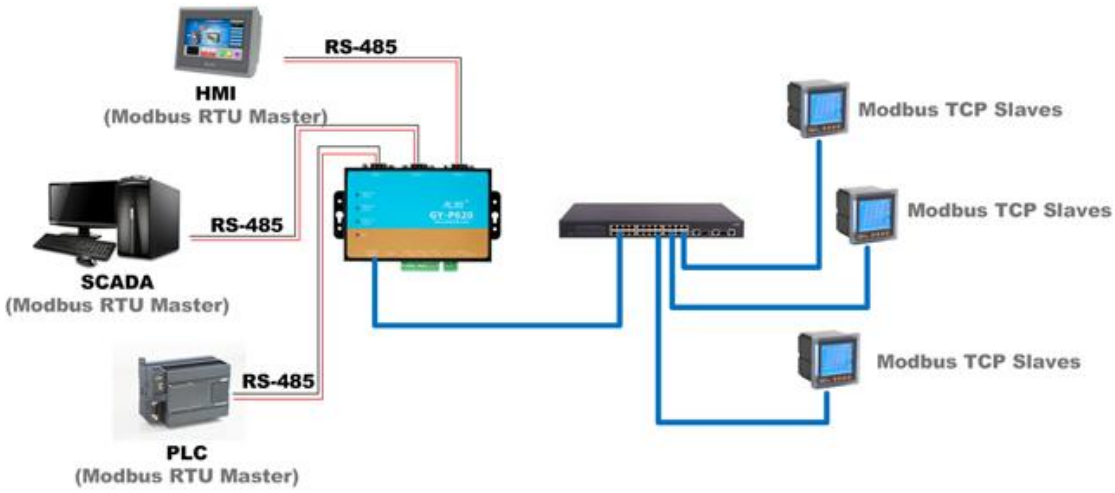
- **多 Modbus TCP 主站同时访问，毫秒级响应速度：**



- **多 RS485 主站同时访问多 RS485 从站:**



- **Modbus RTU 主站访问 Modbus TCP 从站:**



1.3 产品优势

Modbus 协议在工业中得到广泛的应用，而随着工业 4.0 的发展趋势，越来越多的支持 Modbus 协议的仪器仪表要求联网监控，传统的 Modbus 网关就是把原来支持串口 Modbus 的设备适配成支持网口，这样就能方便联网。而我们的 Modbus 智能网关相比传统 Modbus 网关做了很多的改进，如下：

- **更高的采集速度。**在采集多个 Modbus 从站数据时比传统 Modbus 网关采集速度快很多，传统的 Modbus 网关只是 Modbus RTU 与 Modbus TCP 的协议转换，本身并不处理数据，而我们的智能网关是自动采集数据并缓存到本地。因此，如果主站要采集 100 台从站数据，原本要 100 次读命令，而通过 Modbus 智能网关，最优的情况是只需要一次读命令就可以。
- **更高的可靠性。**将多个读命令中的数据集中到几个命令来完成，减少数据采集过程中来回交互的次数，也就减少了出错概率。
- **支持 MQTT/Modbus 连接云端。** 用户可自定义上传数据的格式，然后网关通过支 MQTT 协议发布到云端。同时也支持 Modbus TCP 协议连接云端，网关作为 TCP Client，云端主动发送 Modbus 读写请求。
- **支持数学运算处理。**支持高低字节交换，浮点运算。
- **编程更简单。**通过 PLC 或组态软件采集数十个设备的数据，编程过程比较繁琐。而通过智能网关，编程过程变得简单很多，只需要集中处理所采集的数据。
- **支持多 RS485 上位机。**传统 Modbus 网关是不能支持 485 总线上多个主站的，原因是多主站会产生数据冲突。而使用 Modbus 智能网关就不存在这个问题。多主站是直接与智能网关通讯，读写智能网关缓冲中的数据。
- **支持多 Modbus TCP 上位机。**传统 Modbus TCP 网关在多个主站同时访问从站时，采用排队的机制，这受制于 485 总线同时只能有一个主站。因此同时访问的主站越多，回复会越慢。而我们的智能网关是把数据缓冲在设备内部，实时读取，没有延时。
- **支持 64 位浮点运算。**网关可以在本地处理整型，单精度浮点，双精度浮点运算；处理字节调换等复杂的数学运算，简化上位机（PLC/SCADA/HMI）的工作量，特别是 PLC，在处理字节序转换，浮点运算比较麻烦。
- **支持透传模式。**网关也可以配置成透传模式，并且支持多 TCP 上位机访问。
- **支持串口集线器功能。**多个 RS232/RS485 主站同时访问一个 RS232/RS485 从站，支持任何主从串口通讯的协议。
- **完善的错误跟踪机制。**传统 Modbus 网关只数据转换，不做数据交互的逻辑判断，所以不会记录错误信息，这些错误分析都需要在上位机处理。在采集很多从站时，若有一台从站出问题，很难在很短时间内排查出来。而我们的智能网关内部集成了

详细的日志功能，可以快速找到出问题的从站，前期调试非常方便和迅速。同时，上位机也可以读取所有从站的工作状态，及时发现问题并通知工作人员。

- **降低工程成本**。原先需要多台 PLC 采集数据，现在一台数据采集器配合一台 PLC 就可以完成采集。

1.4 产品外观

型号	图片	说明
GY-G2A		1 路 RS232 + 1 路 RS485
GY-G300		3 路串口， 每一路 RS232/RS485 二选一 存储 1 万个字，500 条指令，支持 8K 数学运算
GY-G3B		3 路串口， 每一路 RS232/RS485 二选一 存储 1 万个字，1000 条指令，支持 128K 数学运算
GY-G600		6 路串口， 3 路 RS232 + 3 路 RS485

GY-G6A		6 路 RS485，存储 1 万个字，500 条指令，支持 8K 数学运算
GY-G6B		6 路 RS485，存储 1 万个字，1000 条指令，支持 128K 数学运算
GY-G6H		6 路 RS485，存储 3 万个字，500 条指令，支持 8K 数学运算

1.5 规格参数

1.5.1 GY-G2A

以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	2
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-485 x 1 + RS-232 x 1
	接口	RS-485：5.08 mm 间距绿色端子 RS-232：DB9 针式
	串口信号	RS-232：RX,TX, GND, RTS, CTS RS-485：A(+), B(-), GND
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站；最多可采集 500 条从站纪录
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	500
	数学运算数量	4K 个字符
机械特性	外壳	铁
	重量	200g
	尺寸	92 × 84 x 23 mm （含挂耳，不含端子）
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.2 GY-G300

		GY-G300
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	3
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-232/RS-485 通过软件二选一
	接口	RS-485：3.81 mm 间距绿色端子 RS-232：DB9 针式
	串口信号	RS-232：RX,TX, GND, RTS, CTS RS-485：A(+), B(-), GND
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	500
	数学运算数量	4K 个字符
机械特性	外壳	铝合金
	重量	200g
	尺寸	无挂耳：134 x 85 x 25mm 有挂耳：156 x 85 x 25 mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C

	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.3 GY-G3B

		GY-G3B
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	3
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-232/RS-485 通过软件二选一
	接口	RS-485：3.81 mm 间距绿色端子 RS-232：DB9 针式
	串口信号	RS-232：RX,TX, GND, RTS, CTS RS-485：A(+), B(-), GND
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	1000
	数学运算数量	128K 个字符
机械特性	外壳	铝合金
	重量	200g

	尺寸	无挂耳：134 x 85 x 25mm 有挂耳：156 x 85 x 25 mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.4 GY-G600

		GY-G600
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	6
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-232 x 3 + RS-485x 3
	接口	RS-485：3.81 mm 间距绿色端子 RS-232：DB9 针式
	串口信号	RS-232：RX,TX, GND, RTS, CTS RS-485：A(+), B(-), GND
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	500

	数学运算数量	4K 个字符
机械特性	外壳	铝合金
	重量	200g
	尺寸	无挂耳：134 x 85 x 25mm 有挂耳：156 x 85 x 25 mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.5 GY-G6A

		GY-G6A
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	6
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-485 x 6
	接口	3.81 mm 弹簧端子(RS-485)
	串口信号	RS-485: A(+), B(-)
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站；最多可采集 500 条从站纪录
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站

	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	500
	数学运算数量	4K 个字符
机械特性	外壳	铁
	重量	200g
	尺寸	无卡扣：160 x 95 x 33mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.6 GY-G6B

		GY-G6B
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	6
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-485 x 6
	接口	3.81 mm 弹簧端子(RS-485)
	串口信号	RS-485: A(+), B(-)
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一路 RS-485 最多挂 32 个从站；最多可采集 500 条从站纪录
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数

		据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	1 万个字(20K Bytes)
	Modbus 指令数量	1000
	数学运算数量	128K 个字符
机械特性	外壳	铁
	重量	200g
	尺寸	无卡扣：160 x 95 x 33mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

1.5.7 GY-G6H

		GY-G6H
以太网	端口数量	1
	速率	10/100 Mbps，自适应 MDI/MDIX
	接口	8 针 RJ45
	电磁隔离	内建 1.5 KV
串口	串口数量	6
	电磁隔离	芯片级隔离方案，隔离电压 2.5KV
	串口类型	RS-485 x 6
	接口	3.81 mm 弹簧端子(RS-485)
	串口信号	RS-485: A(+), B(-)
串口通信	波特率	1200 ~ 115200
	数据位	7，8
	停止位	0.5，1, 1.5, 2
	校验位	None，Even，Odd
Modbus	主站采集	除了可采集 Modbus RTU/TCP 从站，还支持写操作；每一

		路 RS-485 最多挂 32 个从站；最多可采集 500 条从站纪录
	本地从站	PLC/HMI 等主站可通过 Modbus RTU/TCP 读取采集的数据，每一个 Modbus TCP 服务最多同时支持 8 个 TCP 主站连接访问，最多可创建 12 个本地主站
	存储空间	3 万个字(60K Bytes)
	Modbus 指令数量	500
	数学运算数量	4K 个字符
机械特性	外壳	铁
	重量	200g
	尺寸	无卡扣：160 x 95 x 33mm
工作环境	工作温度	-40 ~ 75 °C
	存储温度	-40 ~ 85 °C
	工作湿度	5% ~ 95% (无凝露)
电源要求	输入电压	7 - 24 VDC
	功耗	135 mA @ 12V，69 mA @ 24V
保修	保修期	3 年

2 入门指南

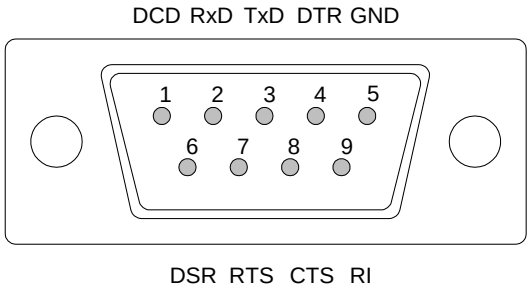
2.1 连接网络

出厂 IP 地址统一是 **192.168.1.222**。先将电脑网络配置成 192.168.1.XXX 网段，对于即有网口又有 WIFI 的笔记本电脑，如果通过网口对设备配置，建议关闭 WIFI（当两个网络接口都是 192.168.1 网段，默认路由走 WIFI，可能导致无法通过网口配置设备）。 然后通过 http://192.168.1.222 访问设备配置，出厂的用户名是 **admin**，密码是 **admin**，打开的 WEB 配置页面如下：

建议使用 Chrome 浏览器对网关进行配置，并且配置网关后要重启后才能生效。

2.2 连接串口

RS-232 接口是 DB9（针式）的，支持的信号脚如下。



DB9 - Male(RS-232)

2.3 连接电源

每个型号提供圆孔，内正外负的 DC 电源座子，电压在“规格参数”中有详细说明，在产品外观上也有指示。同时，可以接 3.81mm 间距的绿色端子的正，负供电。

2.4 RESET 键

如果需要恢复到出厂值，例如忘记之前配置的 IP 地址或者密码，就可以使用复位键。具体操作是，先断电，然后一直按住复位键，上电超过 5 秒钟，松开复位键，系统就自动恢复出厂值。

3 网关的基本概念

3.1 网关的 MODBUS 功能码定义

Modbus 标准中的常用功能码是 1,2,3,4,5,6,15,16。网关定义的功能码含义稍微有点差别，**这里特别说明一下，网关定义的写功能码 5,6,15,16 是循环写，只用于数据搬运，一般很少用。用户需要写网关的时候，在网关定义的功能码 1,3,130,131 已经包含了。网关在接收到上位机写数据时，会自动选择用功能码并把数据写到下位机。这种写只有在收到上位机写命令的时候才会被触发，在写完之后会恢复到循环读的状态。如果扫描周期设置成 0，那么网关就不会做读操作，只有在上位机写数据时才会把数据写向下位机。**具体请看下面的定义：

[1] Read Coils & Write Multiple Coils：网关周期性（周期长度可以设置，如果周期为 0 就是不做读操作）的使用功能码 1 读取数据，当上位机往网关写入位时，网关先存储数据，同时使用功能码 15 将虚拟地址中被更新的多个位写入从站。 网关的功能码 1 包含了读（1）也包含了写（15），这个写只有在上位机往网关写数据的时候才会被触发。

[2] Read Discrete Inputs：使用功能码 2 读取数据位。

[3] Read Holding Registers & Write Multiple Registers：使用功能码 3 读取数据，当上位机有数据写入时，使用功能码 16 将虚拟地址中的数据写入从站。

[4] Read Input Registers：使用功能码 4 读取数据。

[5] Write Single Coil：使用功能码 5 将虚拟地址中的数据循环写入从站。

[6] Write Single Register：使用功能码 6 将虚拟地址中的数据循环写入从站。

[15] Write Multiple Coils：用功能码 15 将虚拟地址中的数据循环写入从站。

[16] Write Multiple Registers：使用功能码 16 将虚拟地址中的数据循环写入从站。

[130] Read Coils & Write Single Coil：跟[1] Read Coils & Write Multiple Coils 的唯一区别是，在上位机往网关写入数据时，网关使用功能码 5 一个一个分多次的往下位机写入线圈，有些下位机只能支持写单个线圈功能码，比如继电器。

[131] Read Holding Registers & Write Single Register：用功能码 3 读取数据，当上位机有数据写入时，使用功能码 6 将单个 Register 写入从站。

3.2 MODBUS 数据地址

当网关采集下位机数据的时候，采用的数据地址是从 0 开始的标准地址，区别于组态或者 PLC 的地址（0x, 1x, 3x, 4x）。有些下位机手册在描述数据地址是采用的是 PLC 的编写方法，对应关系如下：

若设备手册地址是 40001，等同于 Modbus 标准地址的 0

若设备手册地址是 30001，等同于 Modbus 标准地址的 0

若设备手册地址是 10001，等同于 Modbus 标准地址的 0

若设备手册地址是 00001，等同于 Modbus 标准地址的 0

也有些设备手册采用的是 0 开始的标准地址，若是这样就不需要转换。

3.3 MODBUS 的字节序

数据在 Modbus 协议里面通常是以大端（Big Endian，也就是 MSB,Most Significant Byte，高字节在前）或者小端（Little Endian，LSB，Least Significant Byte，低字节在前）存储。但也有一些其它的字节序，这里用 A 表示最高（左边）的字节，字的低（最右边）字节用 B 表示，双字的低（最右边）字节用 D 表示，64 位双精度浮点的低（最右边）字节用 H 表示。这里举个例子，比如十进制的无符号字 4386，换成十六进制就是 0x1122，这里 0x11 就是高字节，0x22 就是低字节。如果依次存储在内存中是 0x11, 0x22，这种存储字节序就是 Big Endian，可用 AB 来表示。如果依次存储在内存中是 0x22, 0x11，那么就是 Little Endian，就用 BA 来表示。

字节序	描述
AB	字的字节序是 MSB
BA	字的字节序是 LSB
ABCD	双字/整数（32 位）的字节序是 MSB
DCBA	双字/整数（32 位）的字节序是 LSB
CDAB	字的 Byte 顺序是 MSB，双字/整数（32 位）的 Word 顺序是 LSB
BADC	字的 Byte 顺序是 LSB， 双字/整数（32 位）的 Word 顺序是 MSB
ABCDEFGH	长整型/双精度浮点（64 位）的字节序是 MSB

HGFEDCBA	长整型/双精度浮点（64 位）的字节序是 LSB
GHEFCDAB	字的 Byte 顺序是 MSB，长整型/双精度浮点（64 位）的 Word 顺序是 LSB
BADCFEHG	字的 Byte 顺序是 LSB，长整型/双精度浮点（64 位）Word 顺序是 MSB

举个例子:

比如有个 32 位的整数(双字),数值是 287454020,换成十六进制就是 0x11223344。如果用 ABCD 放置在内存就是 11 22 33 44（十六进制），如果按照 CDAB 放置在内存就是 33 44 11 22。

反之，如果内存的 4 个字节分别是 11 22 33 44（十六进制），按照 ABCD 的字节序取值就是 0x11223344(十进制 287454020);如果按照 CDAB 取值就是 0x33441122，这个稍微有点绕，其实简单，先画个表，如下

C	D	A	B
11	22	33	44

CDAB 分别对应 11 22 33 44，然后把 CDAB 顺序排列成 ABCD，对应的数值就是 0x33441122（十进制 860098850）。

3.4 网关虚拟地址空间

网关的虚拟地址是一个连续的内存块，一共有 20000 个字节，最后 100 个字（Word9900~Word9999）系统保留。系统上电后自动分配好内存，所以用户在设置采集下位机并把下位机数据映射到虚拟地址的时候，虚拟地址可以任意分配，前提是不超出这 20000 个字节。

因此，网关最多可存储 9900 个字（点，模拟量）。

Modbus 标准中的数据地址的单位可分为两类，一个是从 0 开始的**位地址**，功能码 1,2,5,15 使用的地址就是位地址。另外一个是从 0 开始的**字地址**，功能码 3,4,6,16 使用的地址就是字地址。 位地址的单位就是比特位，而字地址的单位就是字（占用 2 个字节）。

网关的虚拟地址空间是统一并且重合的，使用不同的 Modbus 功能码访问的都是同一个地址空间。比如要访问网关空间的 Bit 16-31 一共 16 个位，可以用访问位的功能码（1,2）也可以用访问寄存器（字）的功能码（3,4）。

用功能码 1 访问时，Modbus 标准数据地址是 16，组态或者 PLC 设置地址是 00016（0x），数量是 16。

用功能码 2 访问，Modbus 标准数据地址也是 16，组态或者 PLC 设置地址是 10016（1x），数量是 16。

用功能码 3 访问，Modbus 标准数据地址是 1（字地址），组态或者 PLC 设置地址是 40002（4x），数量是 1（1 个字，也就是 16 个位）。

用功能码 4 访问，Modbus 标准数据地址是 1（字地址），组态或者 PLC 设置地址是 30002（3x），数量是 1（1 个字，也就是 16 个位）。

3.5 网关位地址和字地址的关系

Modbus 协议标准中的寻址只有两种，位寻址和字寻址。通过网关可以变换寻址方式，比如下位机只能通过位寻址（功能码 1,2,5,15）进行操作，而上位机只能进行字寻址（功能码 3,4,6,16），这时就要涉及位地址和字地址相互转换。

首先，位在字节中的排列是从低（右）往高（左）依次递增（总共 8 个 bit），例如，字节数据 0x0A 的二进制是 00001010，从右往左依次是 bit0 ~ bit7。

字节在字中的排序是高字节在前（下图中的左边），低字节在后（下图中的右边，这里默认字节序是 Big endian，也是 modbus 协议的默认字节序，同时也是网络传输的默认字节序，例如 TCP/IP 协议）。

那么如何把位映射到寄存器地址中？ 这样方便功能码 3，4 来读，6 和 16 来写。有个简单的办法，先规划好要存的寄存器地址，例如有 16 个位，放到寄存器地址 3，那么字地址 3 转换成位地址就是 3 * 16 = 48，如下图：

	寄存器高字节(MSB)								寄存器低字节(LSB)								数值
字(WORD)地址	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	十六进制
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0xAA55
位地址	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56	

上图中绿色部分是对寄存器 3 中的每一个位的位地址标识，一定注意是高字节在前，低字节在后。

注意：如果位存储在网关后，上位机依旧通过位来访问（功能码 1，2，5，15），那么就不需要 16 的整数倍对齐。

3.6 网关故障标志的地址空间

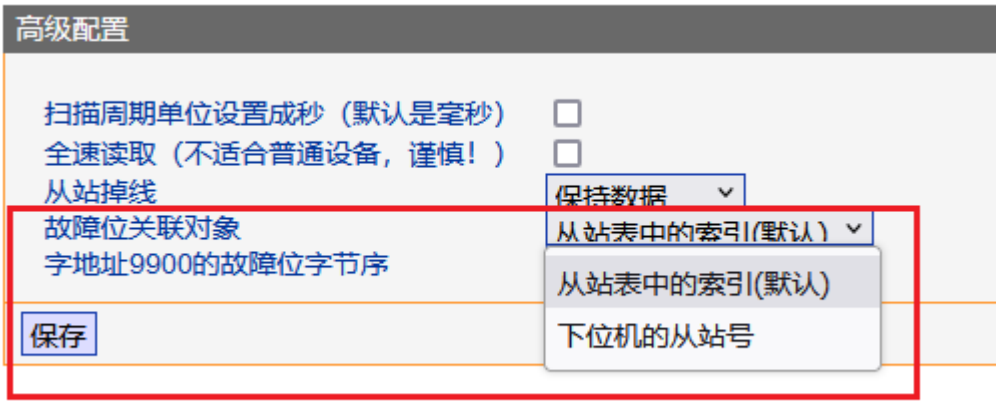
网关的故障位存储在系统空间的 Word9900~Word9931（*型号 GY-G6H 内存是 60K Bytes，故其故障位所在的地址是 Word29900 ~ Word29931*），一共 32 个字（64 个字节，512 个位）。在主站模式下，每一条记录有个 ID，ID 从 1 到 500，类似数据库中的索引号。每一个 ID 对应一个比特位，顺序依次从第一个字节的低位开始到高位，然后第二个字节的地位到高位，以此类推。

举个例子，比如 Word9900 = 0x010B，换成二进制就是 0000 0001 0000 1011，其中 0000 0001 是高字节 0x01，Bit0~Bit7 对应 ID1~ID8，最低比特是 ID1,高比特是 ID8。这里 0x01 中的 Bit0 是 1，表示 ID1 出问题了，通常是掉线了。0000 1011 是低字节，Bit0~Bit7 对应 ID8~ID16，其中 Bit0, Bit1,Bit3 都是 1，表示 ID8, ID9,ID11 出问题了。简单说，Word9900.0 ~ Word9900.7 对应 ID8 ~ ID16， Word9900.8 ~ Word9900.15 对应 ID1 ~ ID8。

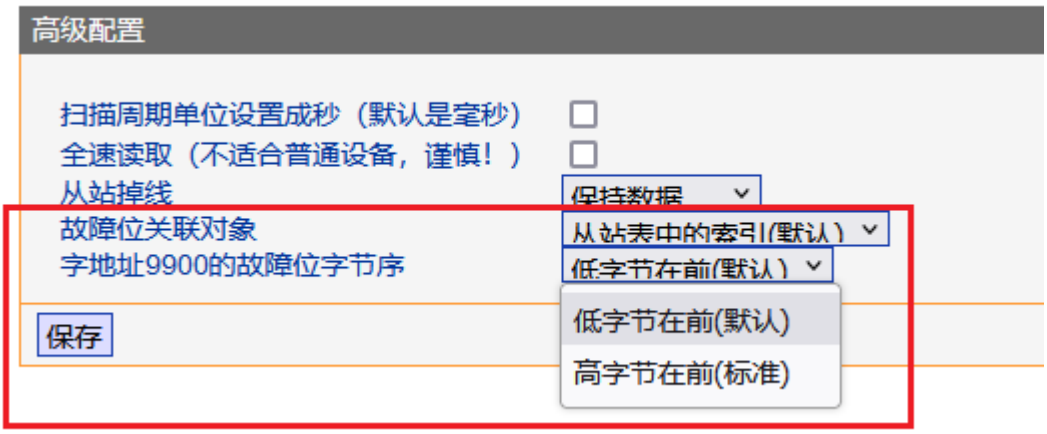
通常组态或者 PLC 用 4 号功能码（只读），对应的起始地址是 39901，这样就可以访问网关的故障位区域。

另外，故障位也可以跟下位机的从站号绑定（仅限于 485 从站），总共 256 个比特位，位的顺序跟指令 ID 存放的顺序是一样的，这样上位机通过读取 9900 开始的 16 个字，只要是哪个位被置，可以很容易哪个从站有故障。这种设置的优点是上位机软件解析很方便，不要根据指令来做动态解析。缺点是设置的下位机站好必须是唯一的，即使不同的 485 口也不能一样。用户可以通过实际情况来做选择，出厂默认设置是，故障位是绑定表中的 ID 号。

注意：该功能仅最新版本支持。

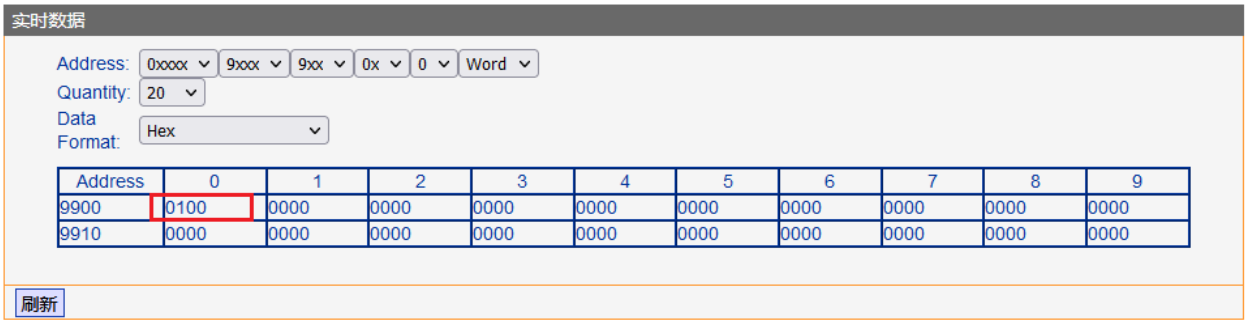


另外，新版本中故障位的字节序也是可以设置的，如下图，出厂默认是低字节在前，所谓“低字节在前”就是在每个字（word）中放故障位的时候从低字节开始，由右边往左边放。而“高字节在前”就是由高字节开始放。还是以例子来说明比较清楚，

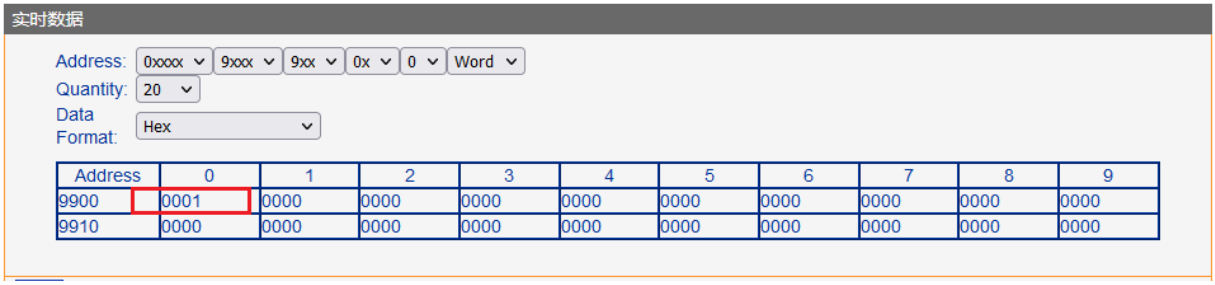


举个例子，如果字节序选“低字节在前（默认）”，并且配置了一条指令，

如果下位机掉线，在【实时数据】页面就可以看到，地址 9900 的数值是 0x0100。



如果选择“高字节在前”，那么同样的配置，在 9900 看到的就是 0x0001，如下图：


实时数据
Address: 0xxxx 9xxx 9xx 0x 0 Word
Quantity: 20
Data Format: Hex
Address 0 1 2 3 4 5 6 7 8 9
9900 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000
9910 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

3.7 网关内变量的表达方式

本小节的概念只有【数学运算】和 MQTT 通讯中才会用到，可直接跳过。
变量定格式定义：

V [Data Type] [Byte Address] # [Byte Order]

V	变量开始的标识符号
Data Type	由 B W D R L C S I 表示 B 8 位无符号字节 W 无符号字 D 无符号双字 R 32 位浮点 L 64 位双精度浮点 C 8 位有符号字节 S 16 位有符号字 I 32 位有符号双字（整数） N/A V 后面直接跟字节地址，表示访问位
Byte Address	字节地址，如果配置时把数据放在某个字地址上，例如 100，那么换成字节地址就要乘以 2，也就是 200。
#	井号表示后面跟的是字节序表达式，可以省略掉，表示默认的 MSB。
Byte Order	参考 Modbus 的字节序章节

3.8 组态软件如何读取网关数据

组态设置的地址通常用 0x,1x,3x,4x 表示，这跟网关的地址有所不同。网关的地址是 Modbus 标准协议的地址（都从 0 开始），并且是统一的地址空间。

组态软件读线圈通常用 0x 地址，比如 DO。组态配置地址时只需要把网关的地址加 1。假设网关的地址是 0，那么组态就配置成 00001 地址（加 1），**注意有些组态配置的线圈地址前面必须用 0 补全成为 5 位数的地址，比如 32，那么就用 00032 表示**。其实组态软件读（功能码 1）或者写线圈（功能码 5,15）的时候，会自动把用户配置的地址减 1，变成标准 Modbus 数据地址（Modbus Data Address），然后发送给网关。

组态软件读只读的离散位通常用 1x 地址，比如 DI 类型。配置的时候，组态软件需要把目标网关的地址加 10001。组态软件用功能码 2 读取数据。

组态软件读可读可写寄存器时，通常用 4x 地址，比如 AO 类型。配置的时候，组态软件需要把目标网关的地址加 40001。组态软件用功能码 3 读取数据。

组态软件读只读寄存器时，通常用 3x 地址，比如 AI 类型。配置的时候，组态软件需要把目标网关的地址加 30001。组态软件用功能码 4 读取数据。

总结： 无论 0x,1x,3x,4x，这些都会被转换成标准的从 0 开始的 Modbus 数据地址，跟网关通讯。而网关本身就是采用的就是标准 Modbus 数据地址（绝对地址），内部配置（在网页上配置）不需要做+1,+10001,+30001 或者+40001 的偏移。

3.9 网关中主站模式和从站模式的区别

主站模式只跟下位机（从站）通讯，而从站模式只跟上位机（主站）通讯。

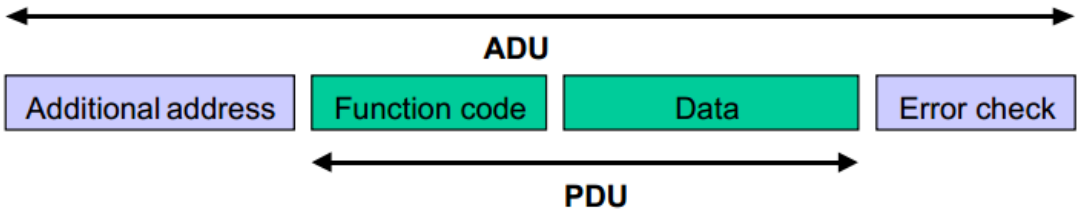
在主站模式下配置时，可以添加基于串口的 Modbus RTU 从站，或者添加基于 Modbus TCP 的从站。网关会对用户配置的从站不停地轮训采集数据并储存在网关内存中。

在从站模式下，网关也可以创建基于某个串口的 Modbus RTU 从站或者基于网口的 Modbus TCP 从站，这样上位机就可以读走网关所采集的数据。

主站模式下的从站不要跟从站模式下的从站混淆，前者是下位机，后者可以理解为网关内建的从站。

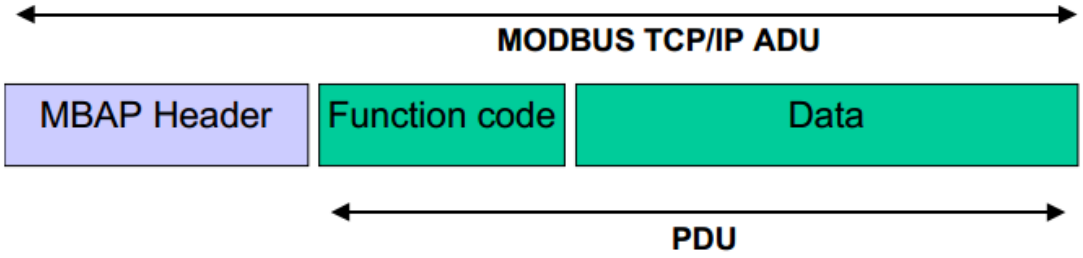
3.10 MODBUS TCP 和 MODBUS RTU OVER TCP 的区别

在 RS-232/RS-485 上通讯的 Modbus 称为 Modbus RTU 协议，如下图，绿色的是数据帧部分，第一个字节是从站地址，最后 2 个字节是 CRC 校验。



而 Modbus RTU 协议除了串口也可以在 TCP 上运行，整个帧格式不变，这种通讯方式称为 Modbus RTU Over TCP。而 Modbus TCP 协议是有改动的，去掉最后 2 个字节的 CRC 校验（底层 TCP 协议已经包含了 CRC 校验，并且网口不像串口通讯那样

干扰大，因此不需要 CRC），并在数据帧前面加了一个协议头，称为 MBAP Header，如下图：



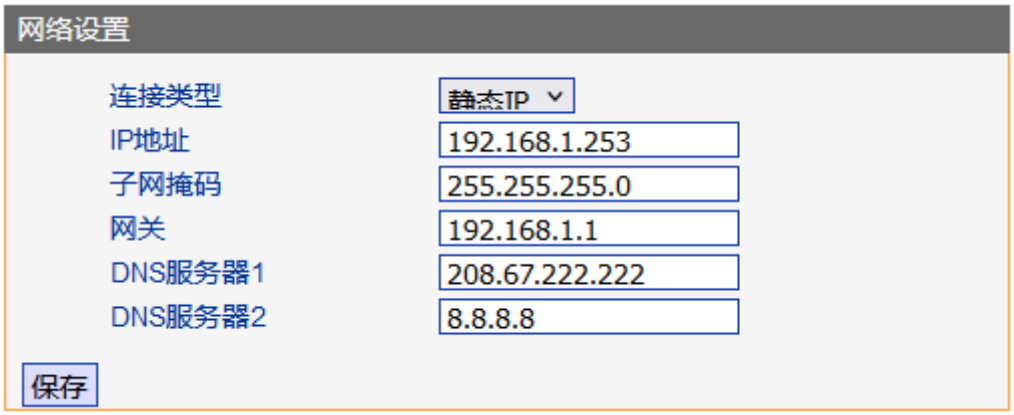
MBAP 协议头包含了事务标识（Transaction Identifier，2 个字节，主站每发送一个请求，该标识都加 1，用于回复与请求的匹配验证），协议标识（始终是 0，占用 2 个字节），数据长度（2 个字节）以及单元标识（Unit Identifier 对应 Modbus RTU 协议中第一个字节的从站地址，1 个字节），总共 7 个字节。

4 远程管理

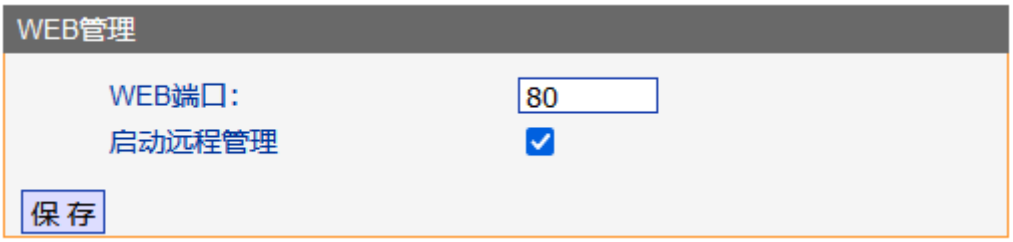
现场只要有网络，就可以配置好网关，登录后台添加设备，这样就可以远程配置和维护设备，极大解决出差成本，或者由于某些原因而不能到现场。（注意，远程管理需要最新的固件）

4.1 网关设置

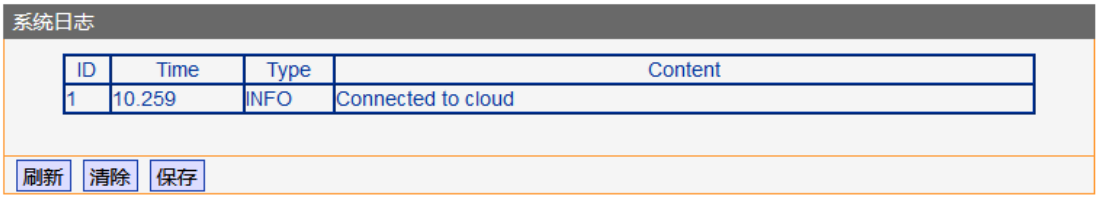
首先，设置网络，并且正确设置网关（路由）的 IP 地址，如下图



其次，在网页【系统设置】-> 【WEB 管理】里面打开“启动远程管理”，并重启网关。



最后在【系统日志】可以看到连接成功的提示。



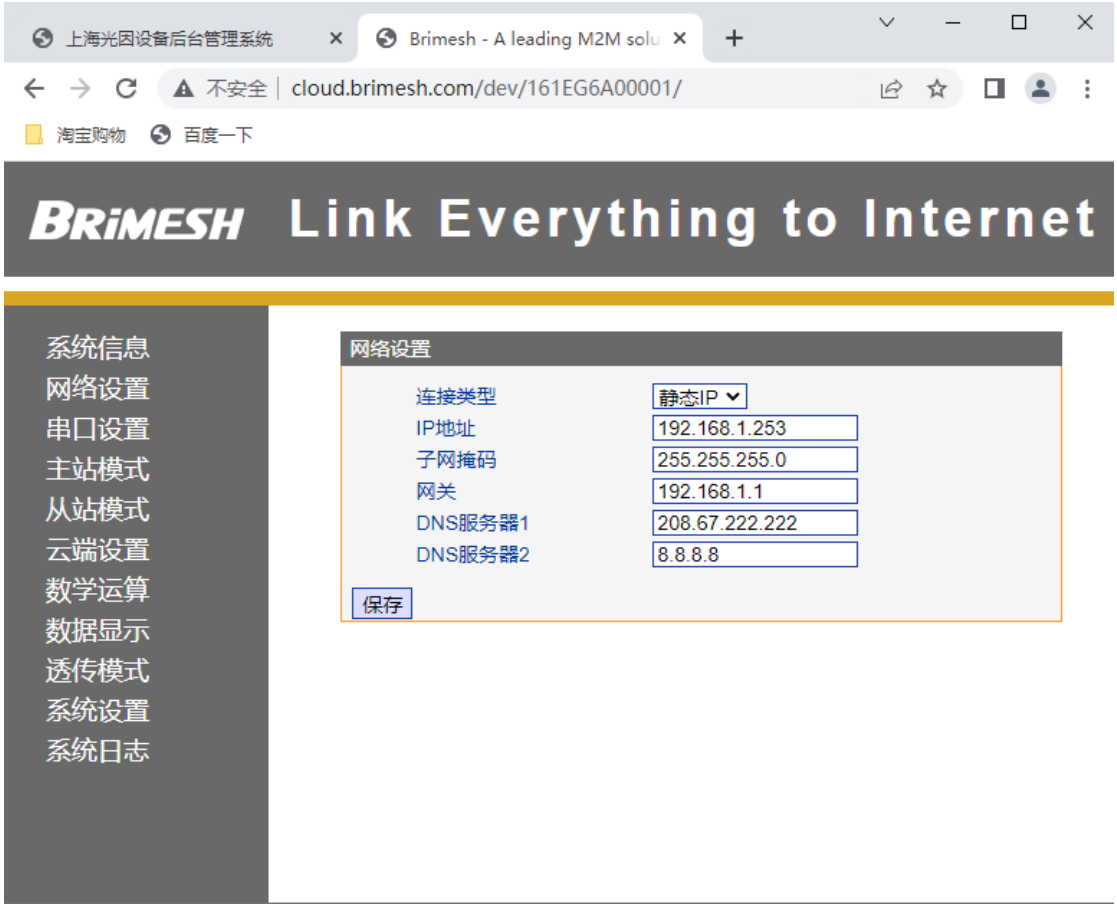
4.2 后台设置

登录 http://cloud.brimesh.com，注册新用户，如下图：

新用户使用注册一个账户，然后登录，点击左下角的【添加设备】，

序列号和 Token 可以在网关的【系统信息】页面找到，用户名和密码默认是 admin，admin。 如果用户在网页内更改过密码，就填写当前密码。最后一个是备注，一般填写客户或者地点，否则设备多了无法区分哪个设备是哪个客户的。保存之后就可以看到设备列表，如果【在线】这一列显示 0，稍等一下，再按 F5 刷新页面，就可以看到在线标志 1。

点击序列号，就可以远程登录设备的管理页面，如下图：


上海光因设备后台管理系统
Brimesh - A leading M2M solu
不安全 | cloud.brimesh.com/dev/161EG6A00001/
淘宝购物
百度一下
BRIMESH Link Everything to Internet
系统信息
网络设置
串口设置
主站模式
从站模式
云端设置
数学运算
数据显示
透传模式
系统设置
系统日志
网络设置
连接类型 静态IP
IP地址 192.168.1.253
子网掩码 255.255.255.0
网关 192.168.1.1
DNS服务器1 208.67.222.222
DNS服务器2 8.8.8.8
保存

5 主站模式

主站模式和从站模式一起，对 Modbus 从站的数据进行重新组态。网关可以通过 RS-232/RS-485，也可以通过以太网采集从站数据。 打开网页，以 GY-G600 为例，左侧的菜单可以看到如下图的子菜单。

点击子菜单，将显示所有的基于对应的串口或网络所要采集的从站信息

5.1 采集 MODBUS RTU 从站

点击“添加从站”按钮，网页会弹出“从站配置”窗口，如下图，

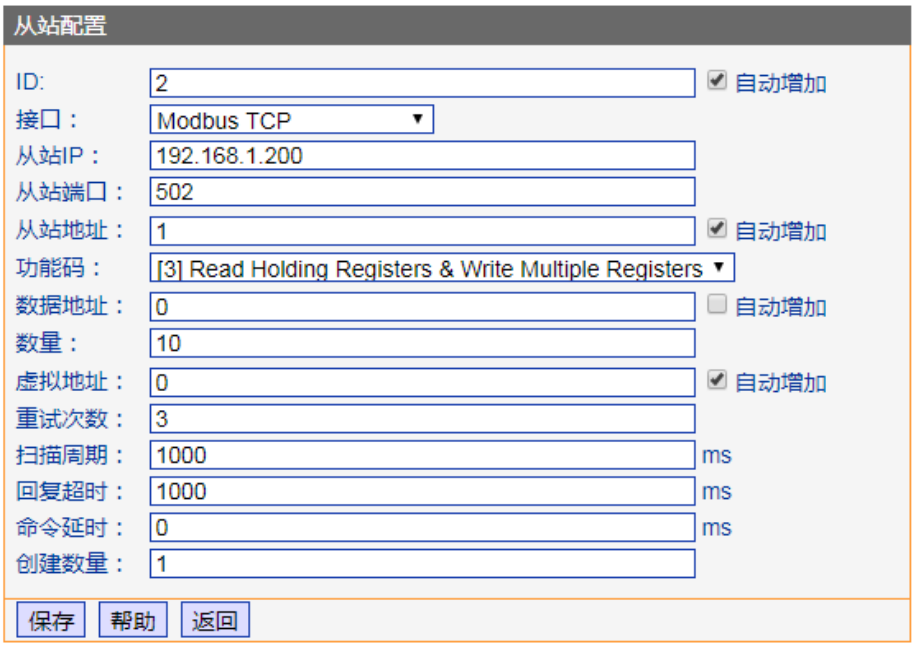
在点击保存之后，从站列表中会显示增加的这条记录，如下图：

注意：对从站进行配置的时候，Modbus 采集会停止，新配置在重启后才会生效。

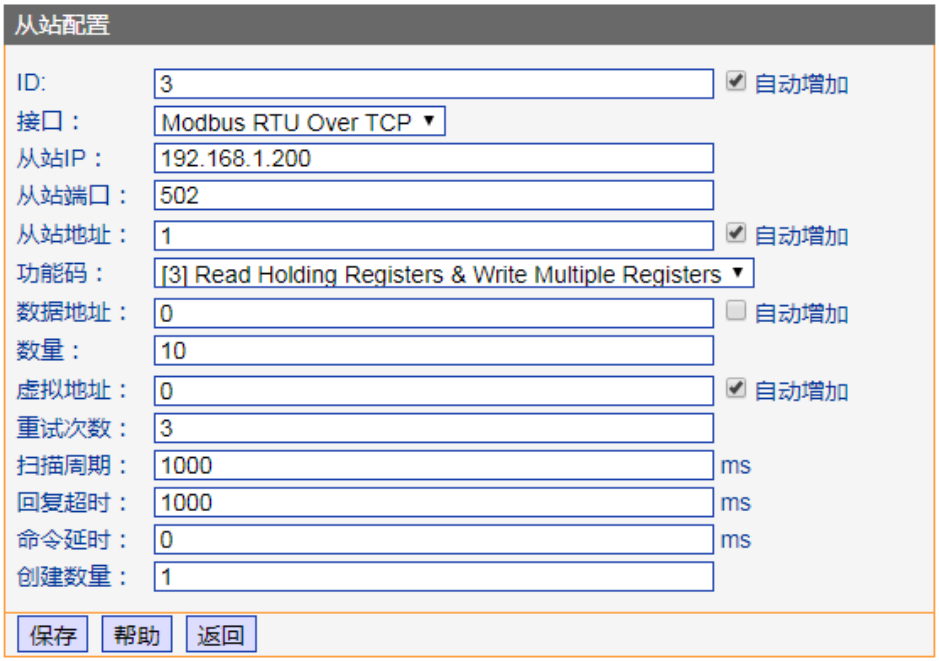
字段名称	说明
ID	范围 1～500，最大可以存储 500 条记录，这个只是表中的索引号，不要跟 Modbus 从站地址混淆。网关会根据指定的 ID 存放在表中对应的位置。
自动增加	【自动增加】选项用于连续创建多条记录。【ID】和【从站地址】每次增加的值是 1，【数据地址】和【虚拟地址】增加的值就是【数量】个单位。再创建完一条或多条记录后，对应的字段值会自动增加，方便创建新记录。
从站地址	要采集的 Modbus 从站地址，设置范围是 1～255。
功能码	参考 网关的 Modbus 功能码定义章节的描述。
数据地址	参考 Modbus 数据地址章节的描述。
数量	如果功能码是对寄存器操作，这的单位就是寄存器个数，一个寄存器占用 2 个字节宽度。如果功能码是对线圈操作，那么单位就是 Bit。
虚拟地址	参考网关虚拟地址空间章节的描述，注意，每条新添加的记录的虚拟地址不能跟之前的重合。
重试次数	通讯失败后连续重试的次数，设置范围 1～255。重试次数超过设定值，该

	记录对应的故障位被置位，网关继续以【扫描周期】的时间间隔不间断重试，直到恢复通讯并清除故障位。
扫描周期	执行本操作所间隔的时间，单位是 ms，设置范围 0～65535。注意：特殊值 0 将不做扫描动作，这样的设置通常用于只写不读的场景。
回复超时	ModBus 命令发出后，等待回复的时间，单位毫秒。
命令延时	网关收到 ModBus 回复后，在发送下一条 ModBus 命令前的等待时间（毫秒），特殊值 0 表示由系统自动设置。
创建数量	一次创建采集记录的数量，如果记录数量大于 1，请参考'自动增加'的规则进行设置。

5.2 采集 MODBUS TCP 从站



5.3 采集 MODBUS RTU OVER TCP 从站



参考： Modbus TCP 和 Modbus RTU Over TCP 的区别章节。

5.4 批量导出导入配置

当网关内的采集的从站比较多的时候，配置导出导入功能就很有用，方便测试或者现场更换设备。比如用户配置了 3 个从站，如下图

ID	接口 [A]	从站地址 [B]	功能码 [C]	数据地址 [D]	数量 [E]	虚拟地址 [F]	重试次数 [G]	扫描周期 [H]	回复超时 [I]	命令延时 [J]	配置
1	RS232-1	1	3	0	10	0	3	1000	1000	0	Edit Del
2	Modbus TCP 192.168.1.200:502	1	3	0	10	10	3	1000	1000	0	Edit Del
3	Modbus RTU Over TCP 192.168.1.201:502	2	3	0	10	20	3	1000	1000	0	Edit Del

点击【下载从站】按钮，默认文件名是 Slaves.csv，用 Excel 打开，A-J 列对应上图的字段。

	A	B	C	D	E	F	G	H	I	J
1	1	1	3	0	10	0	3	1000	1000	0
2	100:192.168.1.200:502	1	3	0	10	10	3	1000	1000	0
3	101:192.168.1.201:502	2	3	0	10	20	3	1000	1000	0

接口的配置：

接口	描述
RS-232/RS-485	数值 1-6 分别标识串口 1-6
Modbus TCP	格式：100:192.168.1.200:502，中间用冒号分割 接口：**100** 表示从站是 Modbus TCP Server IP：　192.168.1.200 表示服务器（从站）的 IP 地址 端口：502 表示 TCP 端口
Modbus RTU Over TCP	格式：100:192.168.1.201:502，中间用冒号分割 接口：**101** 表示从站是 Modbus RTU Over TCP Server IP：　192.168.1.201 表示服务器（从站）的 IP 地址 端口：502 表示 TCP 端口

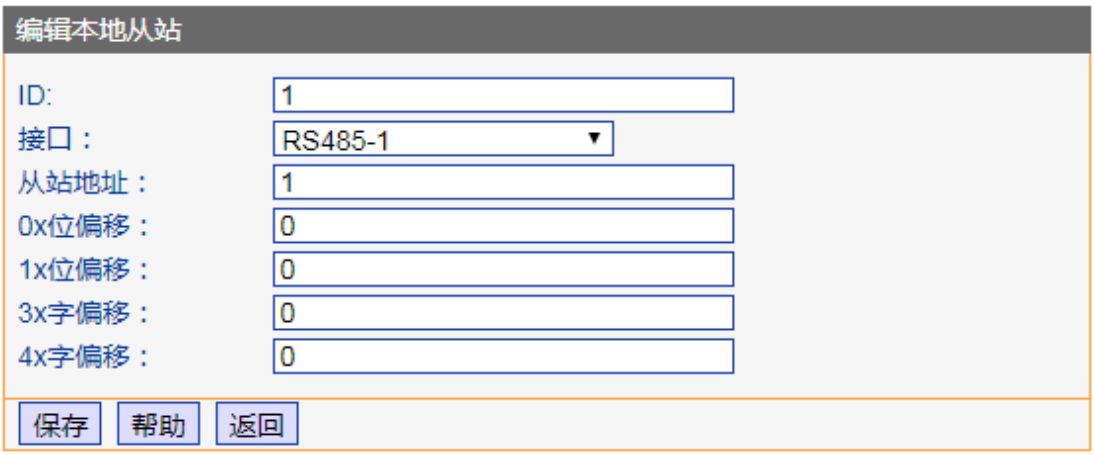
当需要批量导入配置的时候，选择要导入的配置文件，点击【上传从站】按钮，配置就可以导入到网关，重启网关就可以生效。

6 从站模式

从站模式是跟上位机通讯，可以创建基于 RS232/RS485 的从站，也可以创建基于 Modbus TCP 从站。

6.1 创建 RS-232/RS-485 从站

在【从站模式】下选择【添加从站】按钮

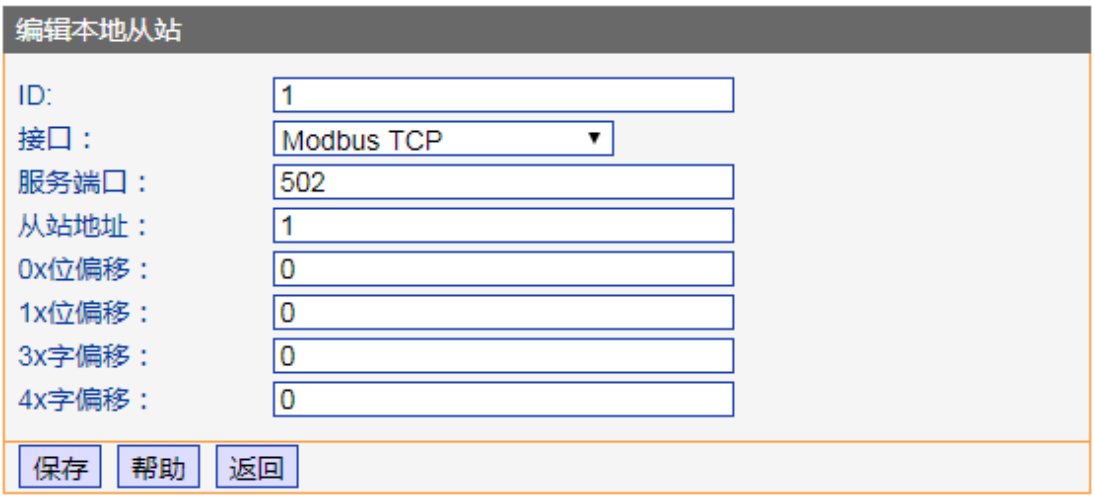


名称	说明
ID	本地从站在表中的索引号，范围 1～16。
接口	RS-232/RS-485 指设备选择对应的串口作为从站接口，或者 Modbus TCP（Server），Modbus RTU Over TCP（Server）。
服务端口	TCP 服务器端口，只有接口是 Modbus TCP 或者 Modbus RTU Over TCP 才需要配置。
从站地址	本从站的 Modbus 设备地址，范围是 1~255。
0x 位偏移	Modbus 功能码 1，5, 15 在虚拟地址中的起始位偏移，设置范围 0 ~ 65535。
1x 位偏移	Modbus 功能码 2 在虚拟地址中的起始位偏移，设置范围 0 ~ 65535。
3x 字偏移	Modbus 功能码 4 在虚拟地址中的起始字偏移，设置范围 0 ~ 65535。
4x 字偏移	Modbus 功能码 3,6,16 在虚拟地址中的起始字偏移，设置范围 0 ~ 65535。

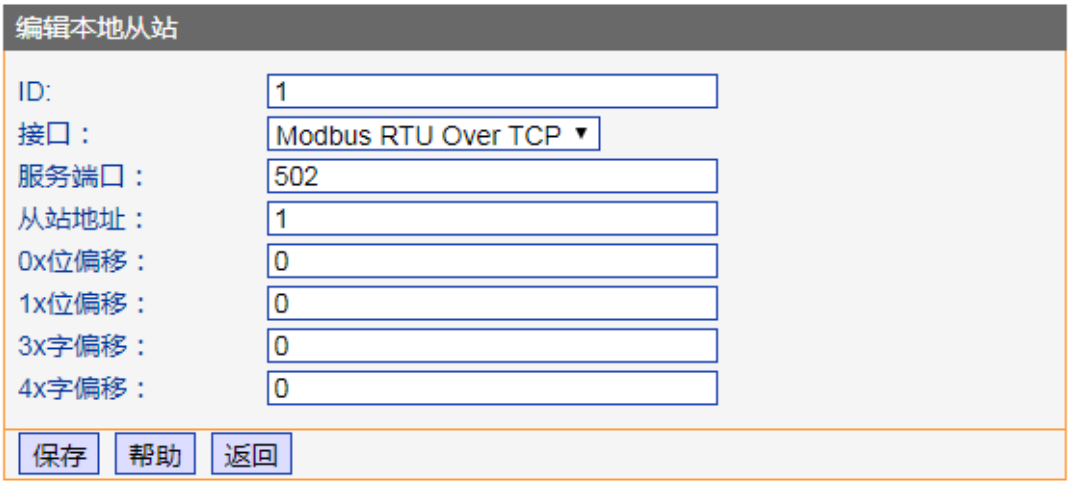
注意： 偏移设置只在上位机跟网关通讯时有效，而网关跟下位机通讯时无效。并且偏移都是正偏移，也就是基于上位机发给网关的数据地址上再增加用户设置的偏移值。

比如上位机的程序的数据地址是固定的，比如 40001（对应 3 号功能码，4x），而下位机采集来的数据存储在寄存器地址 100，这里就可以在创建的从站里面的 4x 设置 100，这样上位机通过 40001 就可以访问寄存器地址 100 开始的数据，网关会把上位机给的地址自动加上 100。

6.2 创建 MODBUS TCP 从站



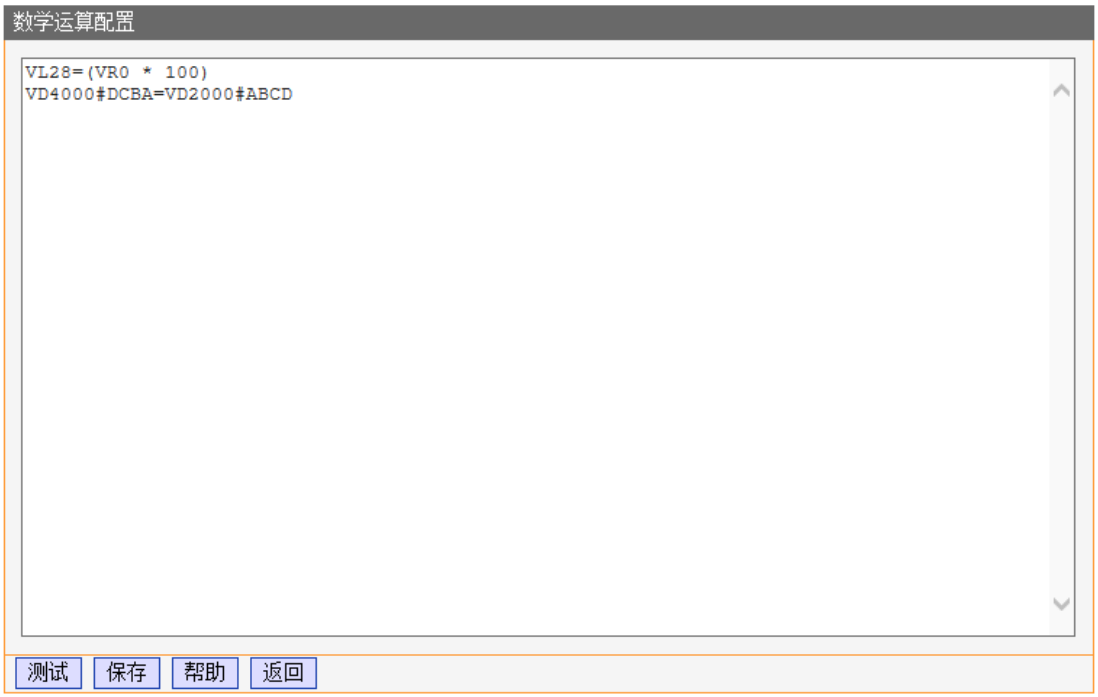
6.3 创建 MODBUS RTU OVER TCP 从站



7 数据运算处理

数学运算的功能可对网关采集的数据进行运算，比如简单的高低字节调换，这个在 PLC 里面处理比较麻烦。

数学运算表达式由运算脚本组成，一行代表一个运算表达式，行的长度不限，可以有多行，整个运算脚本不超过 4096 个字符。表达式由变量和加减乘数，括号构成，类似 C 语言。下图是运算脚本输入框，保存重启后运算脚本会一直循环执行。



关于变量的定义请参考 网关内变量的表达方式章节。

7.1 运算例子 1：高低字节序调换

VD4000#DCBA=VD2000#ABCD

把字节地址 2000 开始的双字由 MSB 转换成 LSB，并且存储在字节地址 4000 开始的内存中。运算结果不建议覆盖，这样方便调试数据。

7.2 运算例子 2： 由不同字节组成一个整数

VD2010=VB2001*16777216+VB2003*65536+VB2005*256+VB2007

把 VB2001,VB2003,VB2005,VB2007 由高到低字节组成一个 32 位整数，并存储在地址 2010。16777216 就是 16 进制的 0x1000000，乘以该值表示左移 24 位，乘以 65536 就是左移 16 位，乘以 256 就是左移 8 位。

7.3 运算例子 3： 浮点转整型

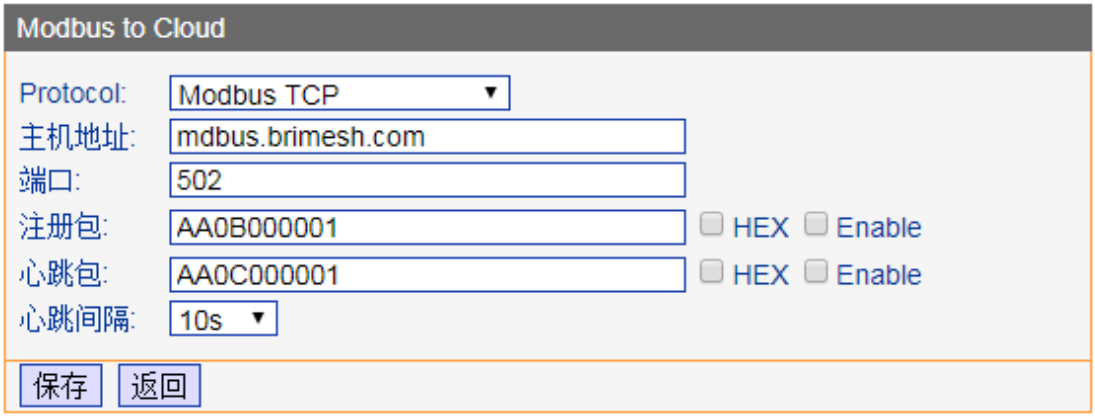
VD20=(VR0 * 10)

把地址 0 的浮点乘以 10 并取整存储在地址 20 开始的双字中。

8 上传数据到云端

网关可以通过 Modbus 协议或者 MQTT 协议连接到云端，以此实现向云端上传数据或者接收云端的命令。

8.1 通过 MODBUS 连接云端



Protocol： 用户可以选择 Modbus TCP 或者 Modbus RTU Over TCP 协议。网关作为 TCP Client 连接到服务器。在 Modbus TCP 通讯中，一般作 TCP Client 的都是 Modbus 主站，也就是说主动发送 Modbus 命令请求。但在连接云端的时候正好反过来，作为 TCP Client 的网关，同时也是 Modbus 从站，Modbus 命令请求由云端发送，网关只返回结果。

主机地址： 云端的域名地址或者 IP 地址。

端口： 云端的服务器 TCP 端口。

注册包： 用户自定义注册包数据，可以 ASCII 文本，也可以十六进制数据。当网关上电后，网关在与服务器建立 TCP 连接之后，如果 Enable 被勾选，网关就会主动发送注册包。如果 TCP 断开重连的时候，注册包也会被发送一次。平常通讯时是不会发送注册包的。 注册包主要是服务器用来识别网关。

心跳包：　用户自定义的心跳包数据，可以 ASCII 文本，也可以十六进制数据。每间隔一段时间，网关就会主动发送心跳包给服务器。

云端（Modbus 主站）

8.2 通过 MQTT 连接云端

MQTT 的配置比较简单，但用户需要对 MQTT 有一定的了解。关于 MQTT 协议的说明请参考 http://mqtt.org。

8.2.1 MQTT 配置

MQTT 调试工具 MQTT.fx 下载链接：http://www.jensd.de/apps/mqttfx

MQTT 配置参数	说明
MQTT Broker	MQTT 的服务器域名或者 IP 地址
Port	MQTT 服务器端口，一般是 1883
ClientID	用户自定义的唯一标识网关的识别码，不同网关不能重合，否则会被服务器踢掉。
KeepAlive	MQTT 心跳包（也就是 PING 包）的发送间隔时间，单位是秒
Publish Topic	网关主动发布数据的 Topic，具体格式参考下一节的定义
Command Topic	网关订阅的命令 Topic，控制端可通过该 Topic 发布写命令
Response Topic	网关返回命令执行结果，后面小节有详细介绍
Publish Interval	网关发布数据的时间间隔，单位秒
AUTH Enable	使能密码设置

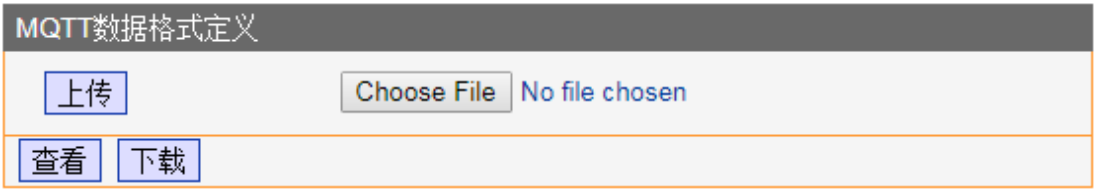
User name	MQTT 用户名
Password	MQTT 密码

8.2.2 PUBLISH TOPIC 数据格式

MQTT Publish 文本最大支持 48K 个字符，支持多行，每一行以回车换行符号结束，并且行的最大长度不超过 48K 个字符。

MQTT Publish 格式定义的核心是变量定义，请参考网关内变量表达式章节。而 MQTT 上传时若引用某个变量，需要在引用符号${}，比如${VW0}，表示取 16 位无符号数。下面是一个简单的 JSON 定义格式，

{"Temperature":${VW0},"Humidity":${VW2}}



然后选择刚定义的文本文件，并点击【上传】按钮，成功后重启网关。网关在发布数据的时候就会把变量名替换成实际值进行 Publish，如下：

{"Temperature":370,"Humidity":980}

注意：网关通过 MQTT 发布数据的时候，一个消息只对应一行，若想把所有数据放在一个消息里面，请在一行内定义完。

8.2.3 COMMAND TOPIC 数据格式

网关同时支持两种下发指令，一种是 JSON 格式，另外一种是数学运算格式。该主题最大支持 2048 个字节大小的消息。

8.2.3.1 JSON 格式

JSON 格式以中括号开头并结尾。下发的任意 KEY 数值在用户定义的 MQTT 模板中能找到，网关就会自动把 VALUE 写入模板配置的寄存器中。例如，模板是
{ “Temperature” : ${VW2000}}

如果网关在订阅的 topic 上收到消息，

{ “Temperature” : 270}

网关就会把 270 写道寄存器 VW2000 上，如果 VW2000 对应到下位机某个寄存器，网关就会把温度值写给下位机。

8.2.3.2 数学运算格式

MQTTCommand Topic 格式跟数学运算格式一样，最大支持 2048 个字符，可以多行，具体可参考数据运算章节的介绍。所不同的是，数学运算里面的目标变量是不能被写到下位机的，而 MQTT 中的对目标变量赋值，则网关会将数据写入对应的下位机。

例子：

```
VW10=1234
V0.1=1
```

第一行是对第五个字（字节地址 10）赋值 1，如果第五个字有下位机对应，那么网关就会将数据写入下位机，效果等同于上位机通过 modbustcp/rtu 协议，功能码 6 或者 16 往网关地址 5 写入一个字数据 1234。

第二行是对第 0 个字节的 Bit 1 设置 1，如果有继电器对应，那么网关就会将对应的继电器打开。效果等同于上位机通过 modbustcp/rtu 协议，功能码 5 或者 15 往网关地址 1 写入 1（ON）。

8.2.4 RESPONSE TOPIC 数据格式

Response Topic 是在网关通过 Command topic 接收到命令后返回执行结果，这样订阅者可以很快的查看命令执行结果，而不必等待下一次的 Publish 数据。如果 Response Topic 是空的，网关则不会返回执行结果。返回的数据格式跟 Command Topic 一样。

8.2.5 MQTT 调试

在配置好 MQTT 并重启后，可以在【系统日志】里面看到如下信息，表示 MQTT 与服务器成功连接上。

系统日志

ID	Time	Type	Content
1	0.002	INFO	MQTT: Start connecting to www.youdomain.com:1883
2	1.014	INFO	MQTT: Connected to www.youdomain.com:1883
3	1.035	INFO	MQTT: Connected to host www.youdomain.com
4	1.057	INFO	MQTT: Successfully subscribed the topic

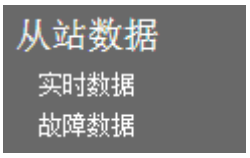
刷新 清除 保存

订阅者通过 Command Topic 发布命令时，用户可以在【系统日志】里面看到接收到消息的信息，如下图。这样方便用户调试查问题。

5	5190.024	INFO	MQTT RECV: message len = 10

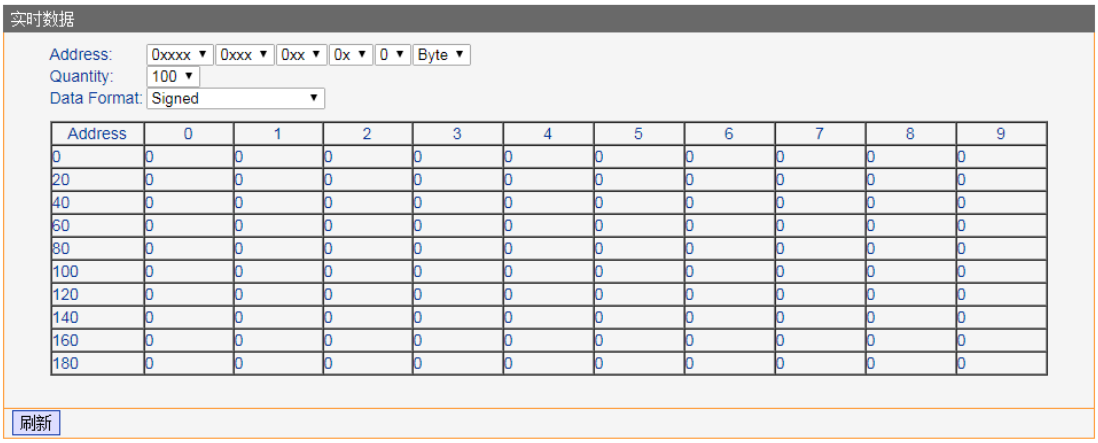
9 数据显示

从站数据分为实时数据和故障数据， 如下图



9.1 实时数据

“**实时数据**”页面显示智能网关内存空间所有的数据。

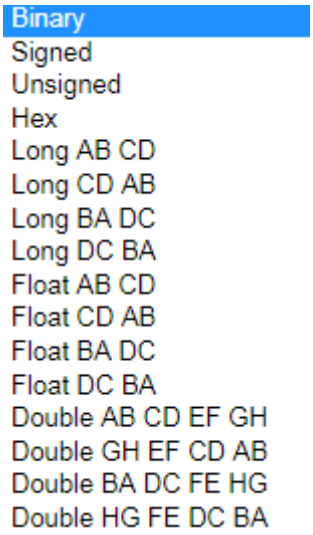


Address	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0	0	0
140	0	0	0	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0	0	0
180	0	0	0	0	0	0	0	0	0	0

Address 后面的 0xxxx 表示高 5 位数，0xxx 表示高 4 位数，0xx 表示高 3 位数，0x 表示高 2 位数，最后一个表示个位数。最后面单位可以选择 Bit，Byte 或者 Word。***用户可以通过上述数值的选择组合成任意的位地址，字节地址，或者字地址。***

Quantity 表示显示多少个数据，数据长度和格式在 Data Format 里面定义。

Data Format 有如下的选项，

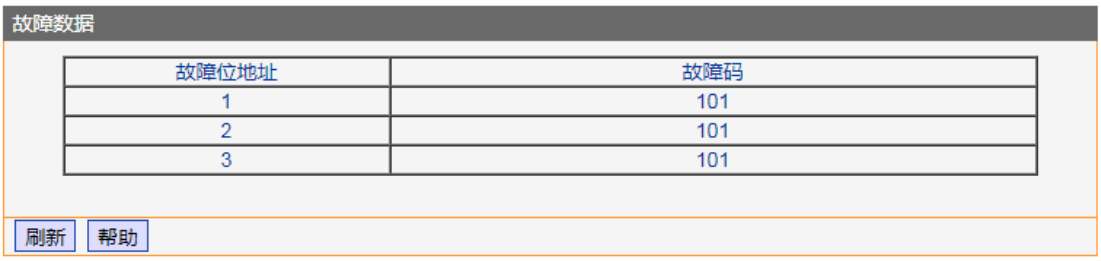


Binary, Signed, Unsigned, Hex 都是显示一个字（2 个字节）的数据，其它的数据定义请参考 Modbus 的字节序章节。

那么这个表怎么看？很简单，行号加列号就是单元格所对应的内存地址。例如，如果单位选 Byte，行号 60，列号 2 的单元格对应的字节地址就是 62。如果单位选 Word，行号 60，列号 2 的单元格对应的字地址就是 62。

9.2 故障数据

“**故障数据**”页面显示所有出故障的从站，并详细标明出故障的代码，而正常运行的从站在该页面不显示，如下图：



“故障位地址”的范围是 1～500，等同于从站表中的 ID 号。关于如何访问故障位的方法，请参考网关的故障标志的地址空间章节。

“故障码”表示具体的出错原因。

异常代码	说明
1	无效的功能码

2	无效的地址空间，访问了下位机不存在的地址
3	无效的数据
101	超过最大重试次数，通常是下位机掉线导致

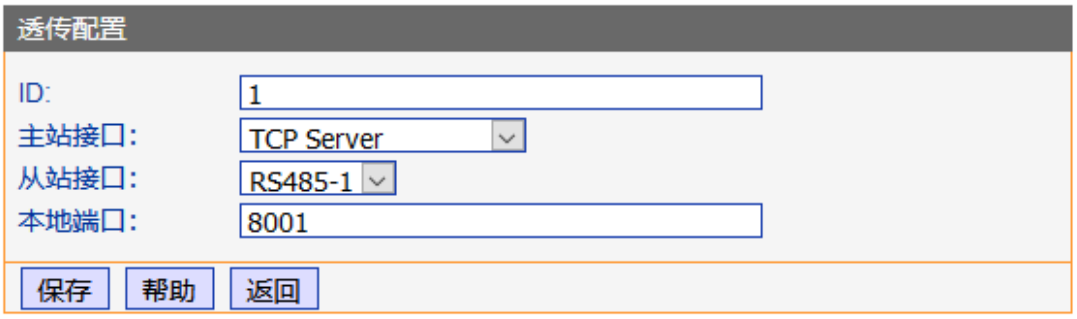
10 透传模式

透传模式可以用来调试设备，多 485 上位机（上位机配置没法改变）访问一个 485 下位机，或者透传数据到上位机或者云端处理私有协议。

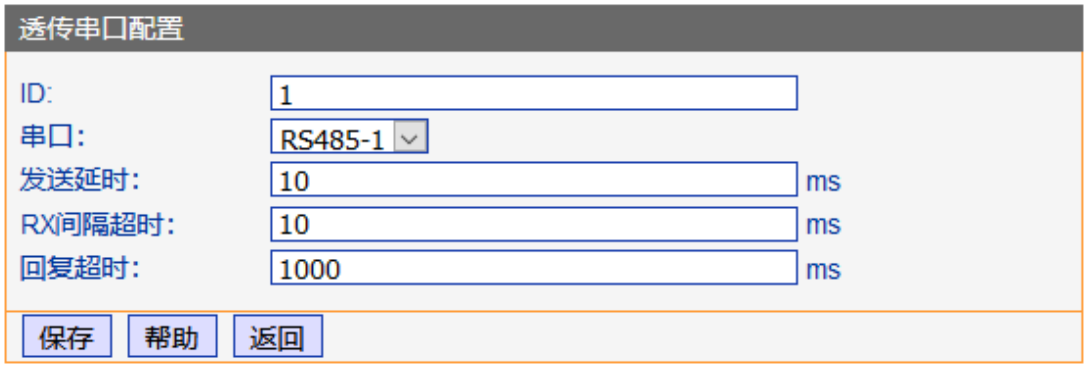
10.1 远程调试 MODBUS 下位机

在配置透传之前，记得把【主站模式】和【从站模式】内的配置都删除，否则可能会冲突。点击【透传模式】页面，如下图：

建立 TCP Server 与对应 RS485 的映射关系，这里选择 TCP Server 透传，如下图，其中本地端口就是 TCP Server 的端口，不同的 485 口 TCP 端口不能一样。



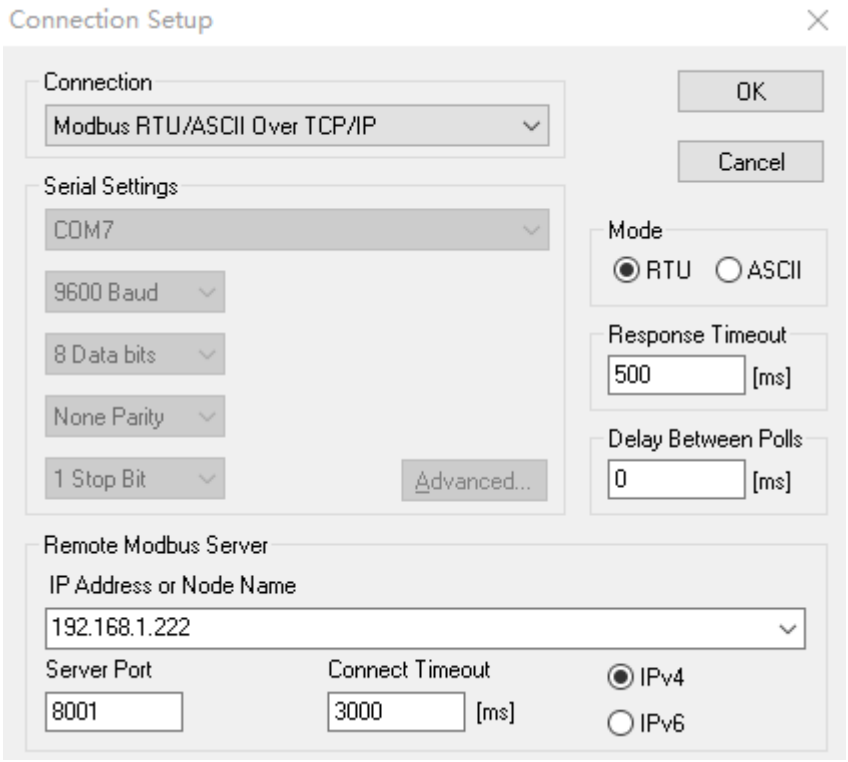
配置要透传的串口，点击【添加】按钮，如下图：



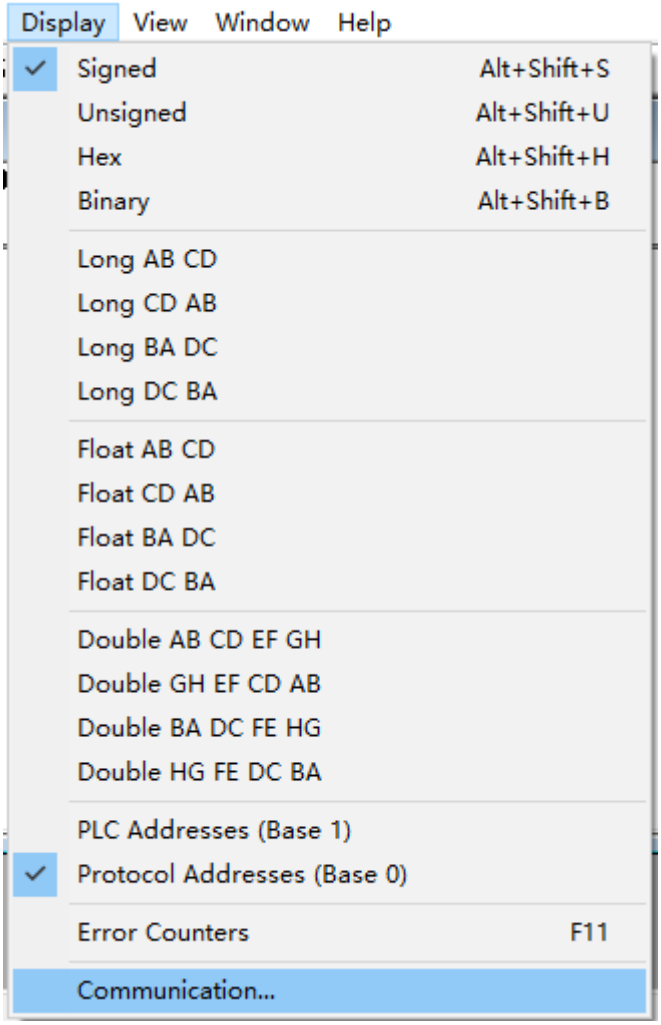
ID:	在表中的索引号，配置范围 1 ~ 1 6。
发送延时:	命令发送前的延时（单位 ms），防止多从站通讯时粘包。
RX 间隔超时:	接收数据包内任意相邻两个字节所间隔的最大时间（单位 ms），超过设定时间，网关将认为两个数据包。
回复超时:	从发送命令开始到收到回复的最大时间（单位 ms）。

配置好之后如下图，记得重启网关后配置才能生效

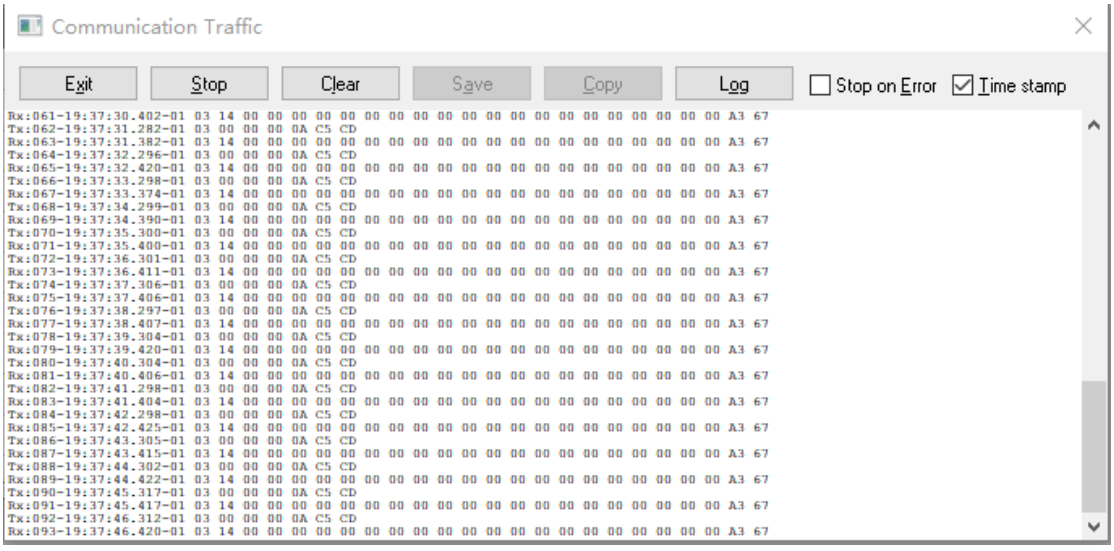
打开 Modbus Poll 调试工具，如下图:



这里 IP Address 就是网关的 IP 地址，Server Port 就是刚才设置的端口，Connection 选择“Modbus RTU/ASCII Over TCP/IP”，然后点击菜单[Display] -> [Communication]



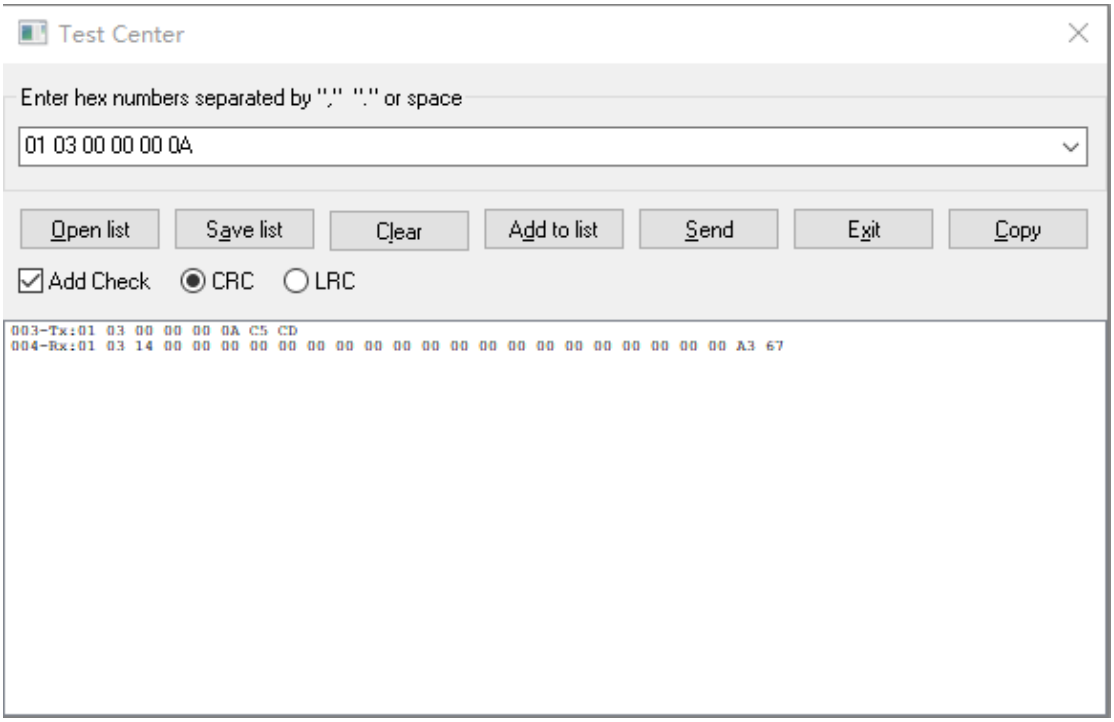
就可以看到交互的原始报文，



由此就可以分析下位机回复的原始报文

另外，也可以手动发送原始报文进行设备调试，点击下图中的图标 TC，

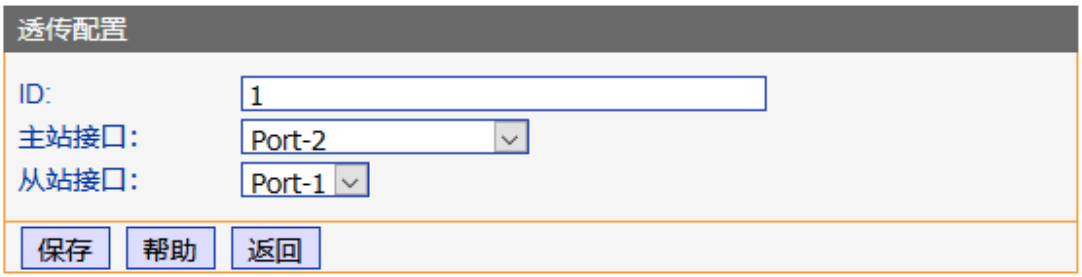
例如，要测试设备 1，功能码 3，地址 0，数量 10 的交互，在下面输入 01 03 00 00 00 0A，打钩 Add Check 会自动加上 CRC。

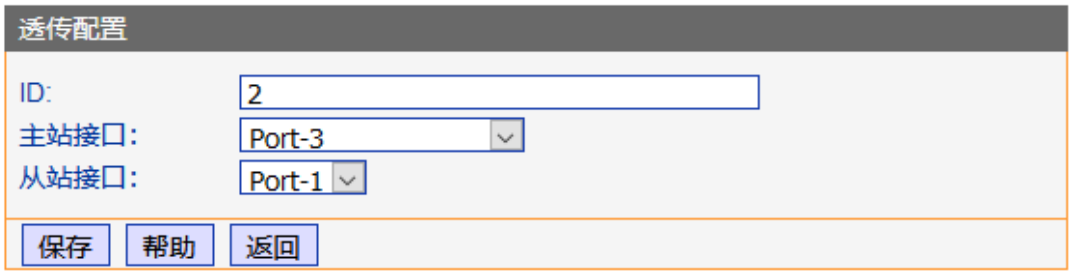


10.2 多路 485 主站访问 1 路 485 从站

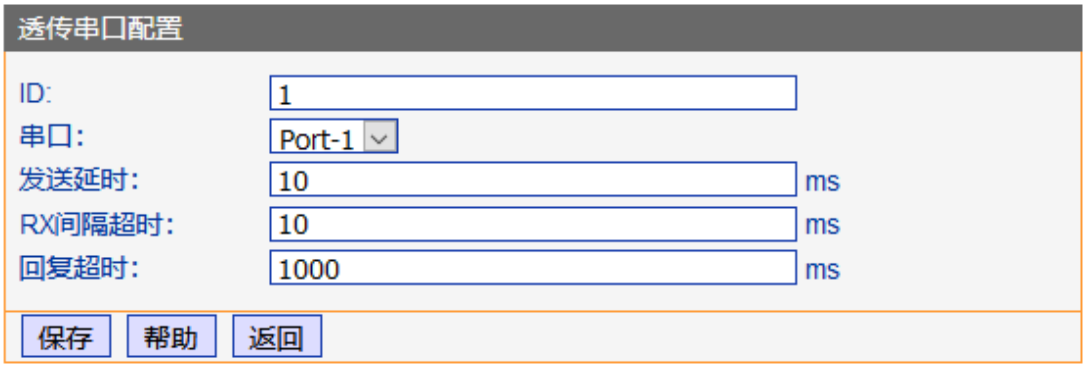
多主站访问 485 从站，通常的是通过网关自动读到虚拟地址内，然后实现数据共享。但这种方式会改变下位机的地址（映射到网关的一个连续唯一的虚拟地址空间），而有些现场的开始是 1 对 1 通信，后来甲方要求增加一台上位机。这时原先的上位机，可能是 PLC, HMI 或者组态软件，都是几年前部署的，源程序都找不到了，更改配置是不可能了，因此只能通过这种透传排队的方式实现多上位机读同一台下位机。

这个例子使用 3 路的型号 GY-G300，配置 RS485-1 接下位机，RS485-2 接上位机 1，RS485-3 接上位机 2。下面两个图是配置映射关系





开始配置下位机对应的串口，这里特别留意“RX 间隔超时”，如果一条指令读的数量很大，例如 50 以上（读的寄存器多，回复的报文容易被分片），建议间隔超时调大一些，例如 50，这样可以避免回复的报文被分片。若通讯质量好再往下减时间。另外，如果波特率特别低，例如 1200，那么这个参数也建议调大一些，例如 100。因为在 1200 的波特率下，传输一个字节需要 8 个毫秒左右，RX 间隔小也容易被分片导致通讯问题。



配置完重启一下网关就可以开始调试了。

10.3 485 主站和 MODBUS TCP 主站同时访问 1 路 485 从站

通过网口和 485 口同时访问一路 485 下的 Modbus 设备，配置跟上一章类似，只是把【主站接口】改成 Modbus TCP Server 模式，整个配置表如下：

透传表配置

ID	主站接口	从站串口	配置
1	Port-2	Port-1	Edit Del
2	Modbus TCP Server : 8001	Port-1	Edit Del

添加数据流 全部删除 帮助

透传串口配置

ID	串口	发送延时 (ms)	RX间隔超时 (ms)	回复超时 (ms)	配置
1	Port-1	10	10	1000	Edit Del

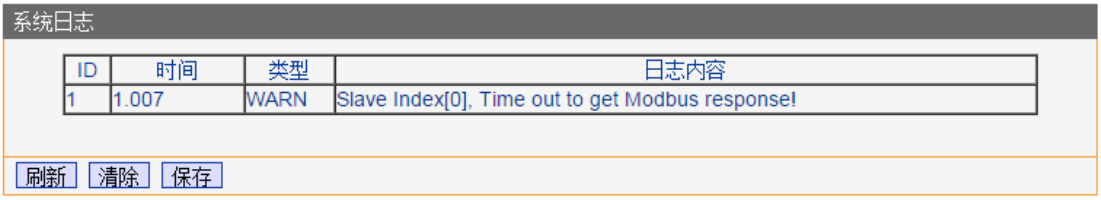
添加 全部删除 帮助

RS485-1 接下位机，RS485-2 接上位机，另外一台上位机连接网口，通过 Modbus TCP Server 8001 端口连接上，以此实现 2 台上位机同时读下位机的目的。

11 系统日志解读

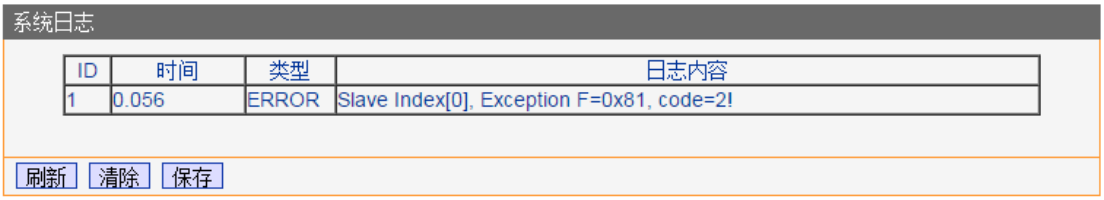
系统日志在前期调试的时候非常有帮助，里面记录采集从站时的出错信息，以及主站读取智能网关时出现的问题。

11.1 回复超时



出现此告警是由于从站在规定时间内没有回复，这里 Slave Index[0]表示从站表中的索引为 0 的记录，请检查物理连接是否正常，比如 RS485 接线是否正确（A 接 A, B 接 B），从站是否运行正常，参数是否设置正确（从站地址是否设置正确），波特率是否匹配。

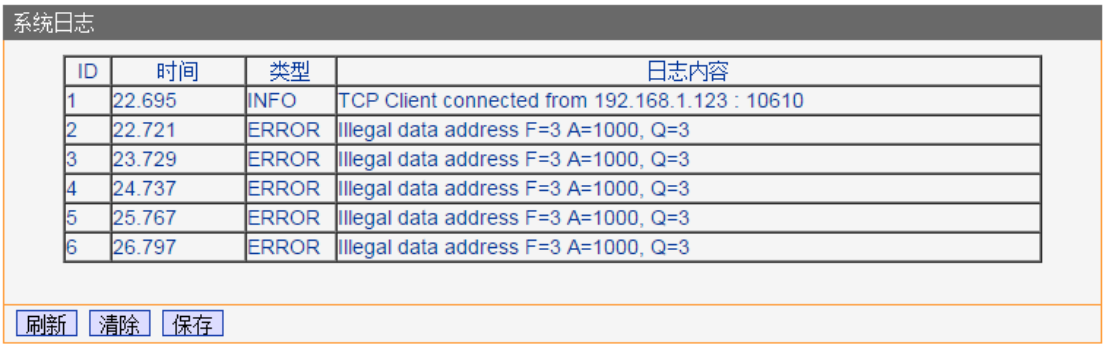
11.2 采集从站异常



这条错误信息表示从站表中的索引为 0 的记录在采集数据时遇到异常，F=0x81，表示出现异常的功能码， code=2 表示异常码，这里的 2 表示无效的数据地址，出现该错误请检查“数据地址”和“数量”是否超出从站的数据范围。具体代码的含义请参考上一章的“故障数据”中的“异常代码”。

11.3 上位机主站异常

上位机在采集数据网关中数据的时候，如果地址不正确，数据网关会报错，如下图



F=3，表示功能码是 03， A=1000，表示地址是 1000，Q=3，表示读取的数量是 3。出现此错误的时候，请检查上位机读取的数据范围是否在“实时数据”页面范围内。

12 案例

12.1 网关和变频器与继电器模块通讯

目的： 通过 Modbus 智能网关，采集变频器和继电器模块，然后上位机通过 RS232/RS485 或者 Modbus TCP 读取设备数据，并设定频率和开关继电器。

工具： 一台安装 Windows 系统的电脑。

一台型号为 GY-G300 的 Modbus 智能网关（GY-G6A/G600 都可以）

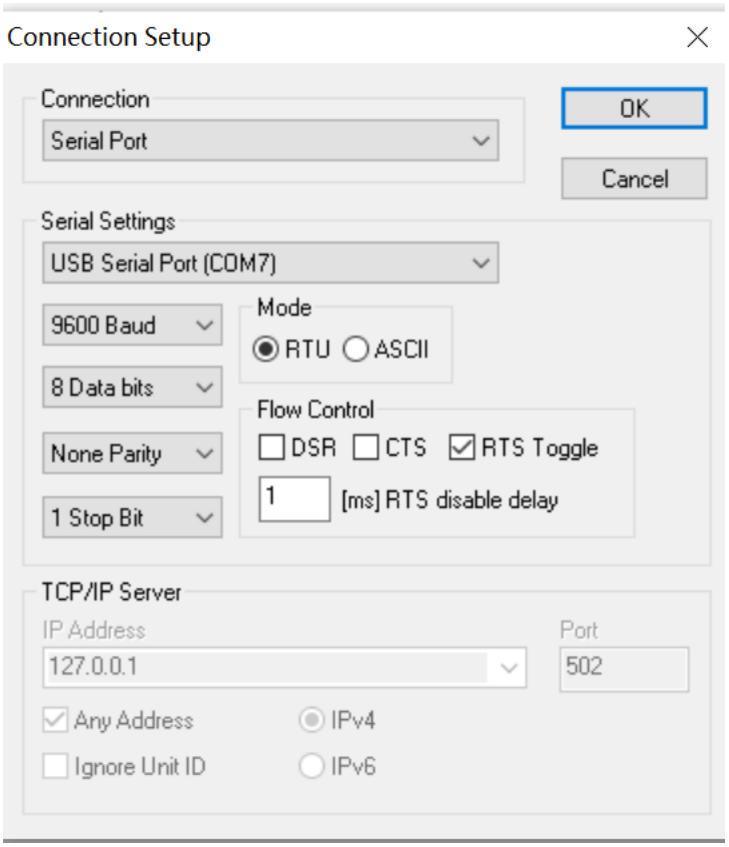
一个 USB 转 RS-232 的转换器加一个 RS-232 转 RS-485 的转接头（或者直接 USB 转 485）

Modbus Slave, Modbus Poll（www.modbustools.com 官网可以下载试用版，网络上也有很多下载的试用版本）

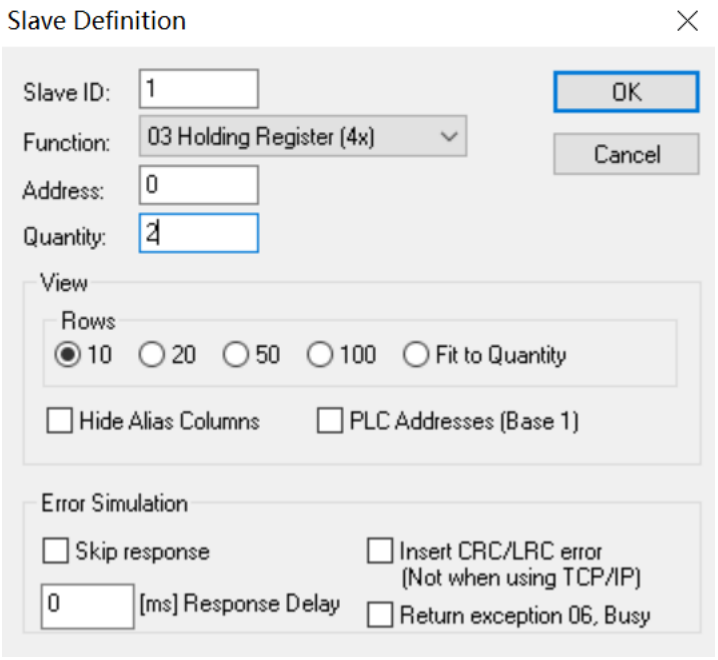
接线：将电脑串口接到 GY-G300 的 Port-1（第一个 485 口）

12.1.1 模拟变频器和继电器模块

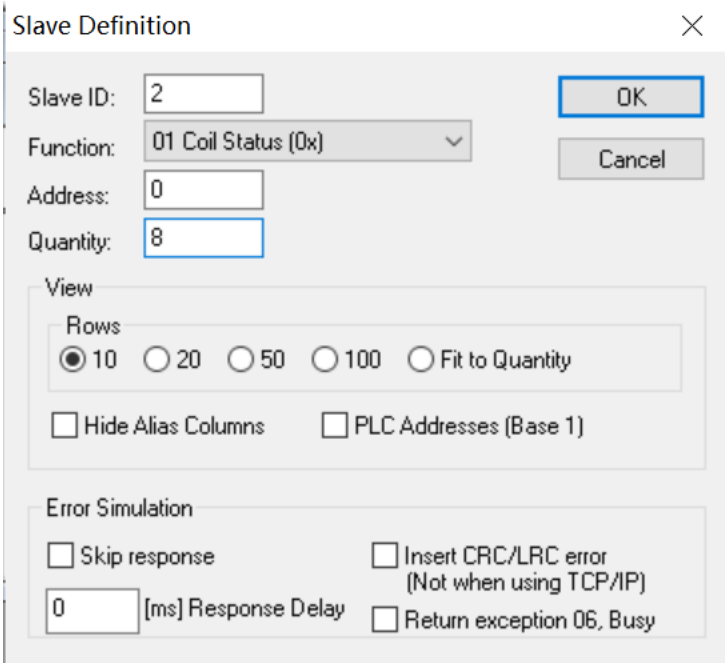
基于电脑 485 口（通常配一个 USB 转 485 接头），通过 Modbus Slave 调试软件，模拟变频器和继电器。



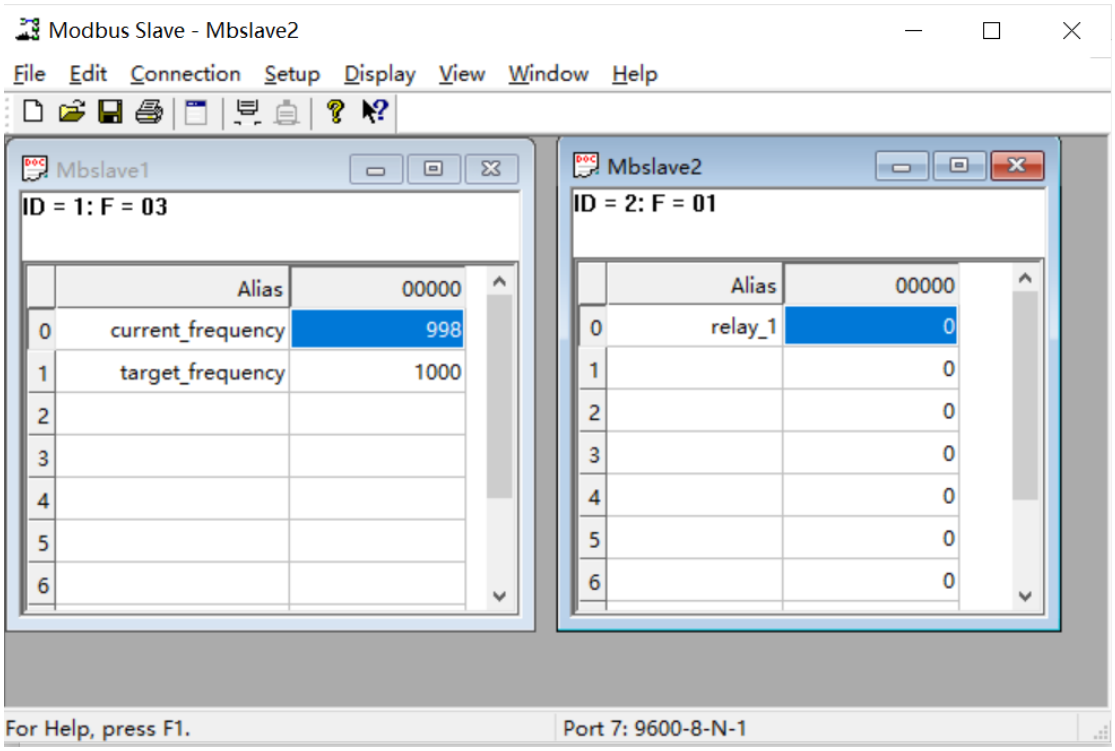
模拟变频器，从站号 1



模拟 8 路继电器模块，从站号 2



对当前频率，设定频率，以及继电器进行标注并设置初始值，如下：

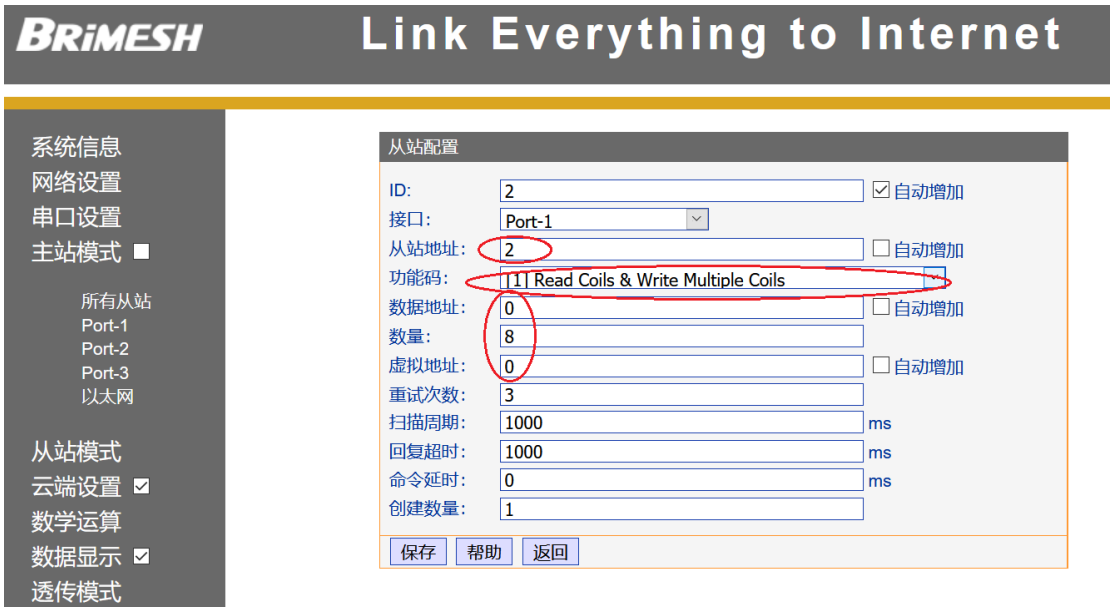


12.1.2 [主站模式]下配置变频器和继电器模块

配置好 485 通讯参数。

添加变频器

添加继电器模块



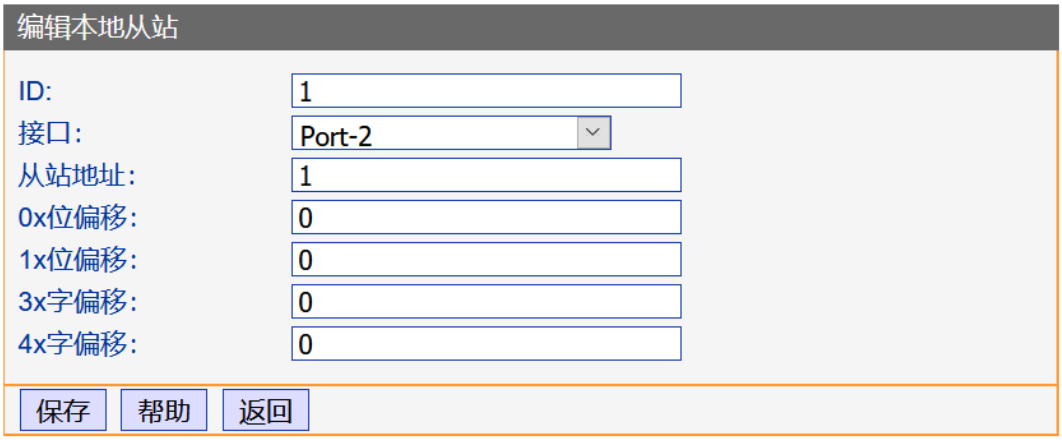
配置完的列表如下

重启网关

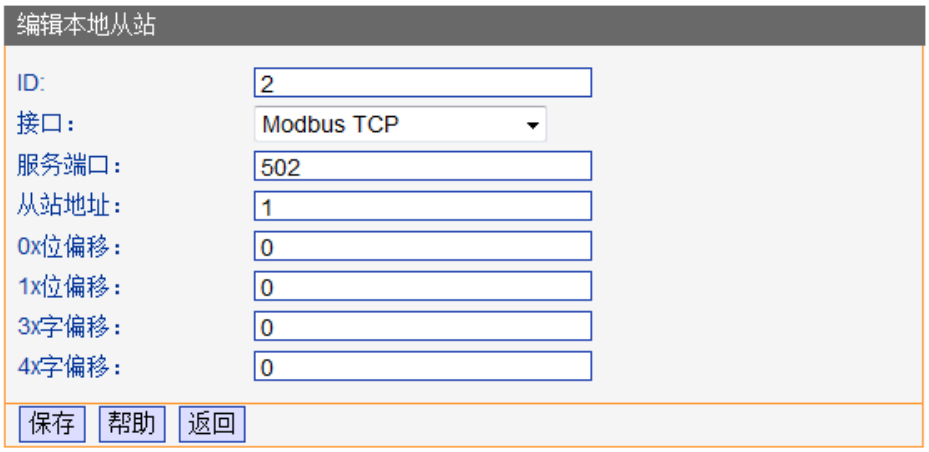
12.1.3 [从站模式]下创建本地从站

本地从站可以给予 RS485/RS232 串口或者网口创建，方便上位机的主站读取数据。

创建一个基于 Port-2（RS485-2）的从站



创建一个基于 Modbus TCP 的从站



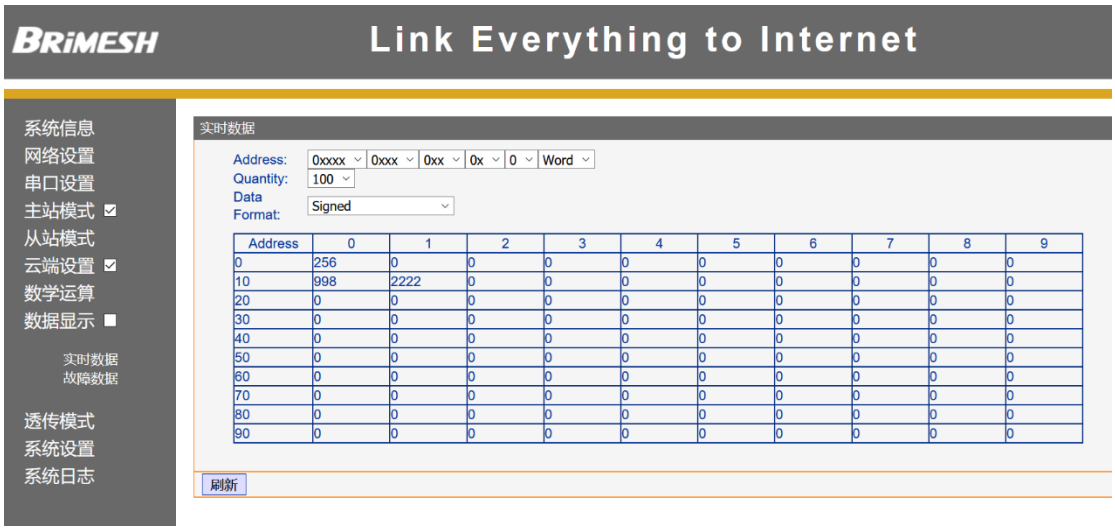
创建完之后，在本地从站表中会显示两个从站

重启智能网关，然后上位机就可以通过 Port-2（中间的 485 口）或者 Modbus TCP，使用 Modbus 功能码 1,2,3,4 来读取网关虚拟地址中的数据。

12.1.4 上位机读取网关数据

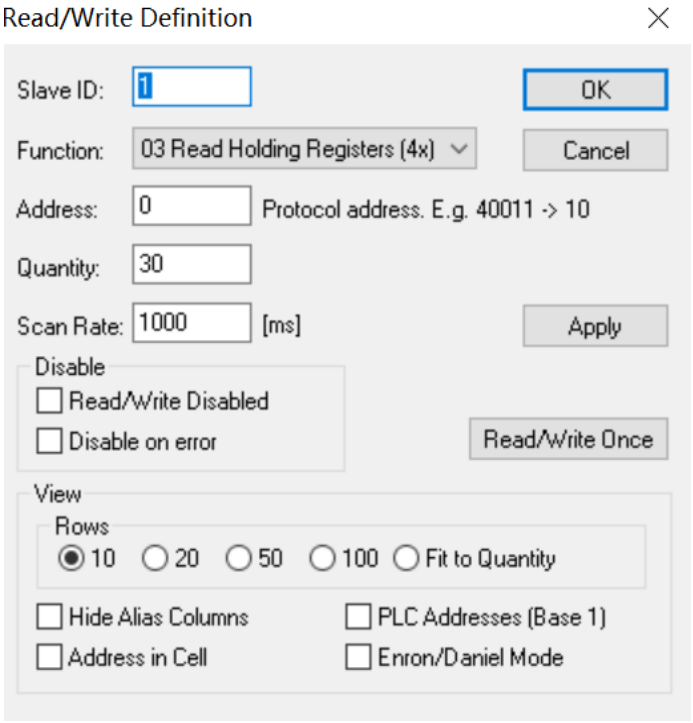
在上位机读取网关数据之前，请检查“系统日志”页面，看看是否有报错，系统日志在调试时很有帮助，详细的信息请看后面的“系统日志”介绍。

然后打开“实时数据”页面，就可以在网页上显示被采集到的数据。

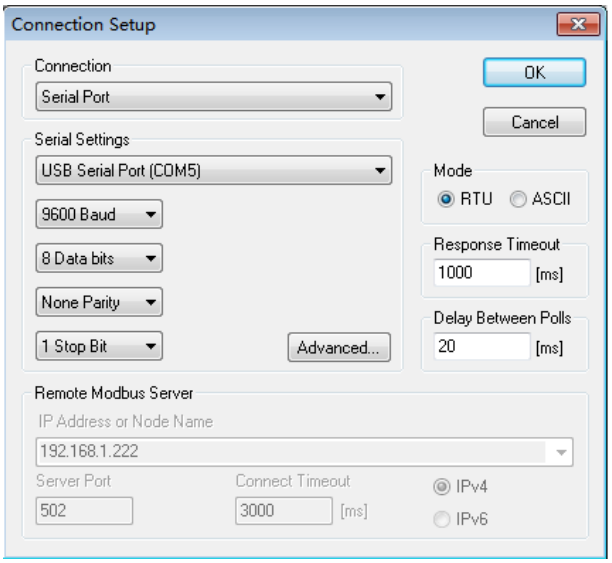


Address	0	1	2	3	4	5	6	7	8	9
0	256	0	0	0	0	0	0	0	0	0
10	998	2222	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0

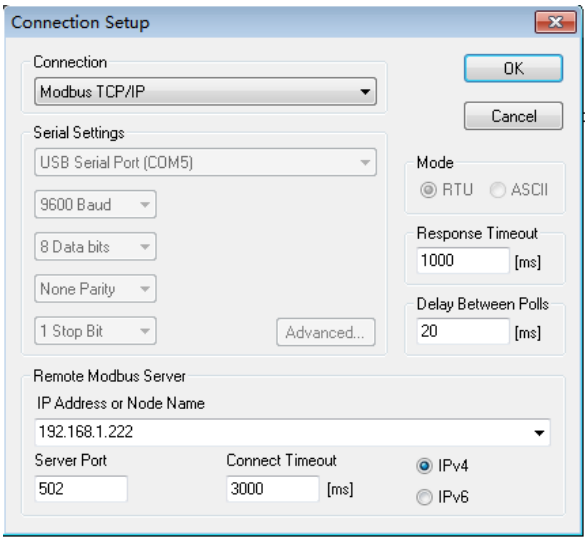
您可以选择 Port-2 来读取数据，也可以选择 Modbus TCP 来读取数据。打开 Modbus Poll 程序，点击"Setup"菜单中的"Read/Write Definition"，弹出如下对话框



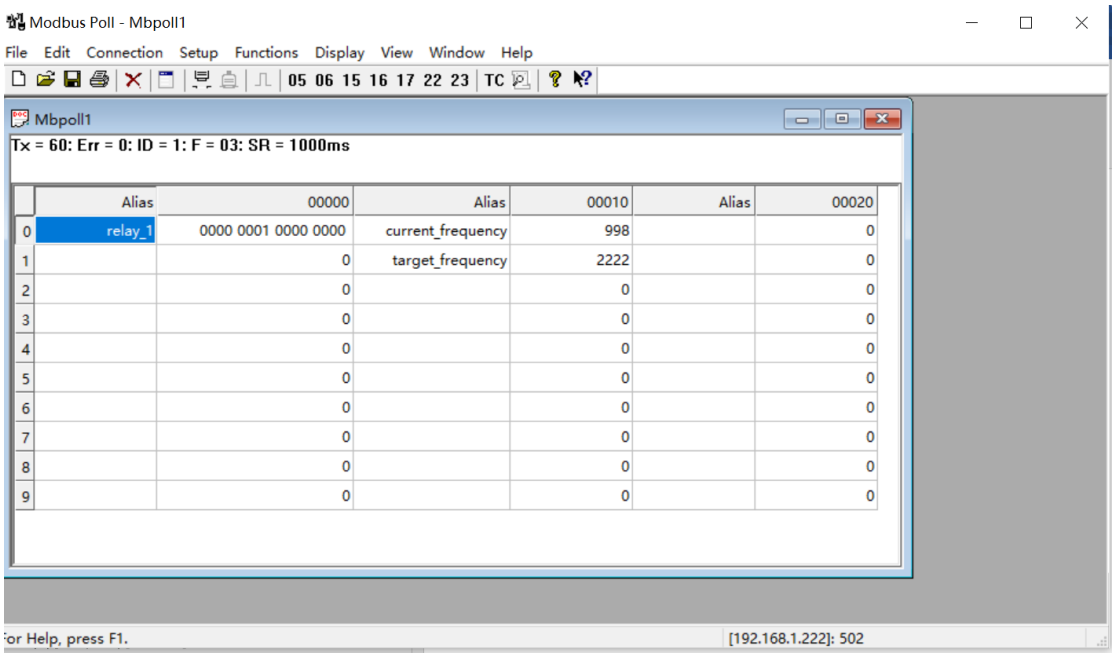
然后点击 Connection 菜单中的 Connect，



如果要通过计算机串口通过网关的 RS485-2 来采集数据，那么在 Connection 里面选择"Serial Port"，选择对应的与网关 RS485-2 相连接的计算机串口，并配置好波特率。如果要通过 Modbus TCP，那么在 Connection 里面选择 Modbus TCP/IP，配置如下图：



这里的 Server Port 设置成 502 是由于我们在“本地从站”中设置好的，建立连接后就会看到如下的数据，跟网关采集到的数据一致（数据可以设置十进制或者十六进制显示）。


Modbus Poll - Mbpoll1
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 17 22 23 TC
Mbpoll1
Tx = 60: Err = 0: ID = 1: F = 03: SR = 1000ms
Alias 00000 Alias 00010 Alias 00020
0 relay_1 0000 0001 0000 0000 current_frequency 998 0
1 0 target_frequency 2222 0
2 0 0 0 0
3 0 0 0 0
4 0 0 0 0
5 0 0 0 0
6 0 0 0 0
7 0 0 0 0
8 0 0 0 0
9 0 0 0 0
or Help, press F1.
[192.168.1.222]: 502

12.2 通过 MQTT 连接阿里云物联网平台

本案例通过网关把变频器的数据和灯光控制器的状态传给阿里云，并通过阿里云远程控制继电器或者设定变频器频率。阿里云物联网平台的配置地址：
https://iot.console.aliyun.com

12.2.1 阿里云配置

阿里云的帮助信息

https://help.aliyun.com/document_detail/131611.html

12.2.1.1 添加产品

[设备管理]->[产品] -> [创建产品]

12.2.1.2 添加设备

[设备管理]->[设备] -> [添加设备]


设备
全部产品
设备总数
1
激活设备
0
当前在线
0
设备列表
批次管理
添加设备
批量添加
DeviceName
请输入 DeviceName
请选择设备标签
DeviceName/备注名称
设备所属产品
节点类型
状态/启用状态
最后上线时间
操作
Device
GY-G300
网关
未激活
查看
删除
子设备(0)
删除
禁用
启用

12.2.1.3 功能定义


GY-G300
ProductKey
a1iNf0nbsNq 复制
ProductSecret
******** 查看
设备数
1 前往管理
产品信息
Topic 类列表
功能定义
数据解析
服务端订阅
设备开发
编辑草稿
物模型 TSL
生成设备端代码
功能类型
功能名称（全部）
标识符
数据类型
数据定义


编辑草稿
产品名称
GY-G300
ProductKey
a1iNf0nbsNq 复制
添加标准功能
添加自定义功能
快速导入
物模型 TSL
历史版本
您正在编辑的是草稿，需点击发布后，物模型才会正式生效。
功能类型
功能名称（全部）
标识符
数据类型
数据定义

添加自定义功能

* 功能类型

属性 | 服务 | 事件

* 功能名称

当前频率

* 标识符

current_frequency

* 数据类型

int32

* 取值范围

0 ~ 5000

* 步长

1

单位

赫兹 / Hz

* 读写类型

○ 读写 ◉ 只读

添加自定义功能

* 功能类型

属性 | 服务 | 事件

* 功能名称

设定频率

* 标识符

target_frequency

* 数据类型

int32

* 取值范围

0 ~ 5000

* 步长

1

单位

赫兹 / Hz

* 读写类型

读写 只读

添加自定义功能

* 功能类型

属性 | 服务 | 事件

* 功能名称

继电器

* 标识符

relay_1

* 数据类型

bool

* 布尔值

0 - 关

1 - 开

* 读写类型

读写 只读

描述

请输入描述

← 编辑草稿

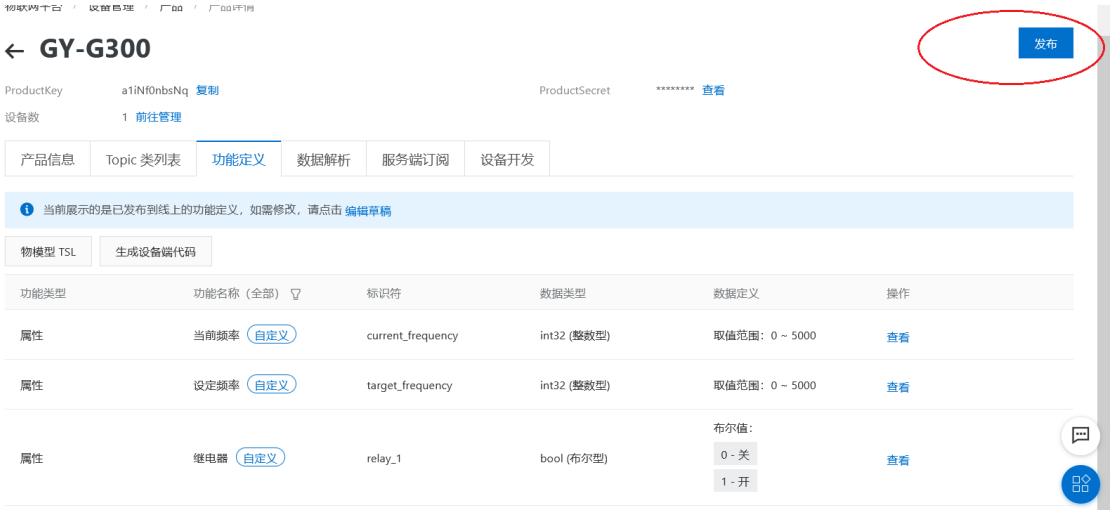
产品名称 GY-G300 ProductKey a1iNf0nbsNq 复制

添加标准功能 | 添加自定义功能 | 快速导入 | 物模型 TSL | 历史版本

您正在编辑的是草稿，需点击发布后，物模型才会正式生效。

功能类型	功能名称（全部）	标识符	数据类型	数据定义	操作
属性	当前频率 自定义	current_frequency	int32 (整数型)	取值范围：0 ~ 5000	编辑 删除
属性	设定频率 自定义	target_frequency	int32 (整数型)	取值范围：0 ~ 5000	编辑 删除
属性	继电器 自定义	relay_1	bool (布尔型)	布尔值：0 - 关 1 - 开	编辑 删除

发布上线 返回


GY-G300
发布
ProductKey a1iNf0nbsNq 复制
ProductSecret ******** 查看
设备数 1 前往管理
产品信息
Topic 类列表
功能定义
数据解析
服务端订阅
设备开发
当前展示的是已发布到线上的功能定义，如需修改，请点击 编辑草稿
物模型 TSL
生成设备端代码
功能类型
功能名称（全部）
标识符
数据类型
数据定义
操作
属性 当前频率 自定义 current_frequency int32 (整数型) 取值范围：0 ~ 5000 查看
属性 设定频率 自定义 target_frequency int32 (整数型) 取值范围：0 ~ 5000 查看
属性 继电器 自定义 relay_1 bool (布尔型) 布尔值： 0 - 关 1 - 开 查看


确认发布产品
您即将发布的产品为： GY-G300
产品发布后将从开发阶段进入到正式投产或使用阶段。
请勾选并确认该产品的信息和各项功能已具备发布条件：
第 1 步
请确认产品的各项基本信息准确无误，产品发布后将无法再做修改和删除。
已确认
第 2 步
请确认设备的各项功能已经完成开发和调试，发布后产品的功能改动请通过 OTA 升级。
已确认
第 3 步
请确认产品已经具备上线发布条件，开始进入规模化接入和部署。
已确认
发布
取消

12.2.2 网关配置

12.2.2.1 模拟变频器和继电器模块

参考案例 1 的《模拟变频器和继电器模块》

12.2.2.2 [主站模式]下配置变频器和继电器模块

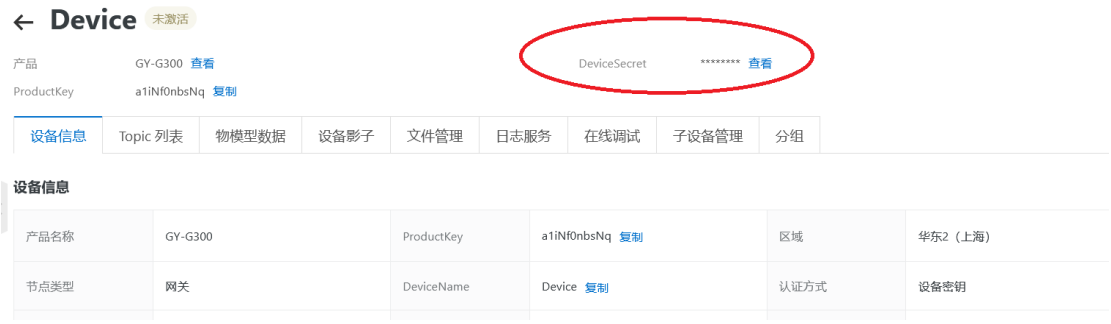
参考案例 1《[主站模式]下配置变频器和继电器模块》

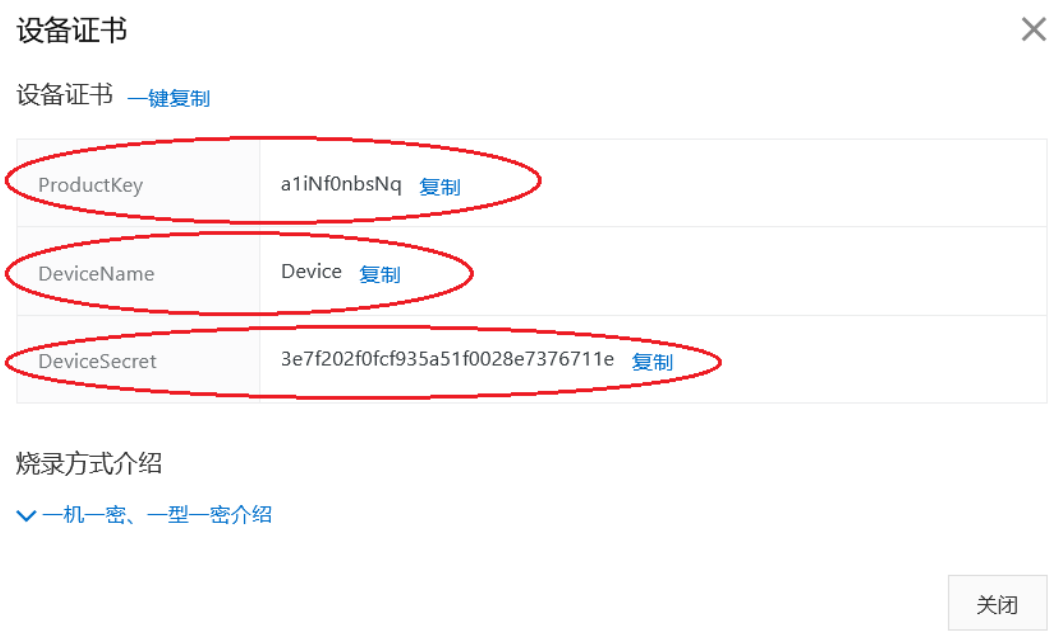
12.2.2.3 在阿里云物联网平台查阅账号

下面的链接是详细介绍阿里账号配置的文档，供参考。

https://help.aliyun.com/document_detail/86706.html

配置步骤如下：





密码生成工具链接：https://files.alicdn.com/tpsservice/88413c66e471bec826257781969d1bc7.zip?spm=a2c4g.11186623.2.25.3bb07908PepkPW&file=88413c66e471bec826257781969d1bc7.zip

填入设备信息：

productKey: a1iNf0nbsNq
deviceName: Device
deviceSecret: 3e7f202f0fcf935a51f0028e7376711e
timestamp:
clientId: 123456
method: hmacsha1

点击这里： Generate

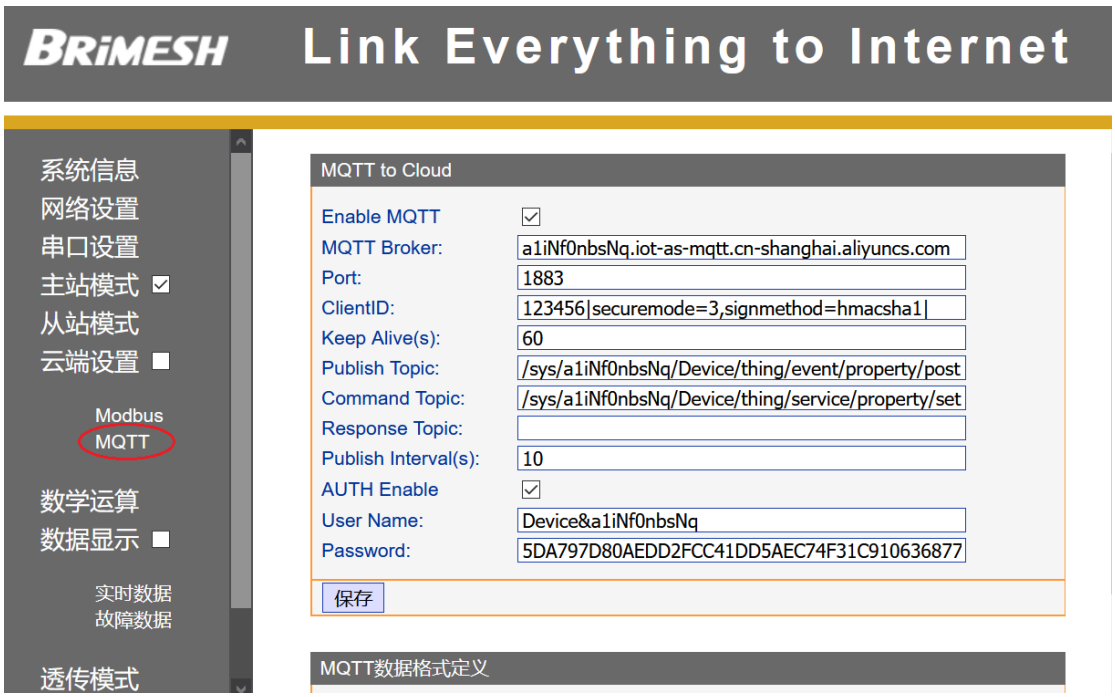
签名结果：

password: 5DA797D80AEDD2FCC41DD5AEC74F31C910636877

查看物联网云平台中 Topic 定义

其中，属性上报 Topic 就是网关发布数据的主题，属性设置 Topic 就是网关订阅并接收云端下发的配置命令主题。

12.2.2.4 在网关内配置阿里云物联网平台账号以及相关 TOPIC



以上的参数的来源请参考上一节。注意，这里的 Keep Alive 的时间如果小于 30 秒，阿里云会报错”Invalid ClientID”，这个并不是账号配置错误。

12.2.2.5 定义上报数据模板

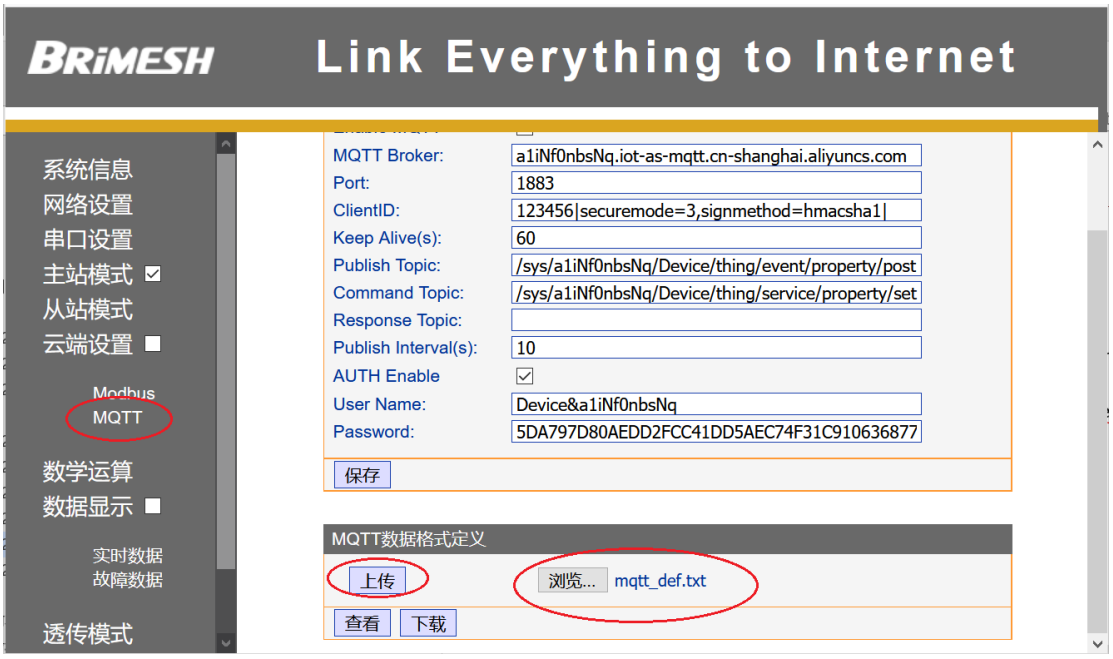
Alink 协议是阿里云基于 json 的协议规范，详细介绍如下：

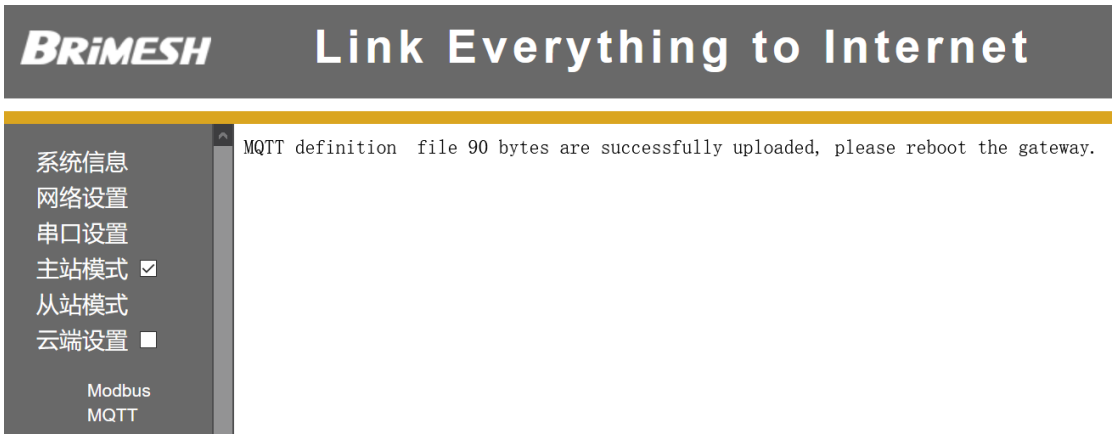
https://help.aliyun.com/document_detail/90459.html

将如下数据保存到一个文本文件中（一定要在同一行），例如 mqttdef.txt

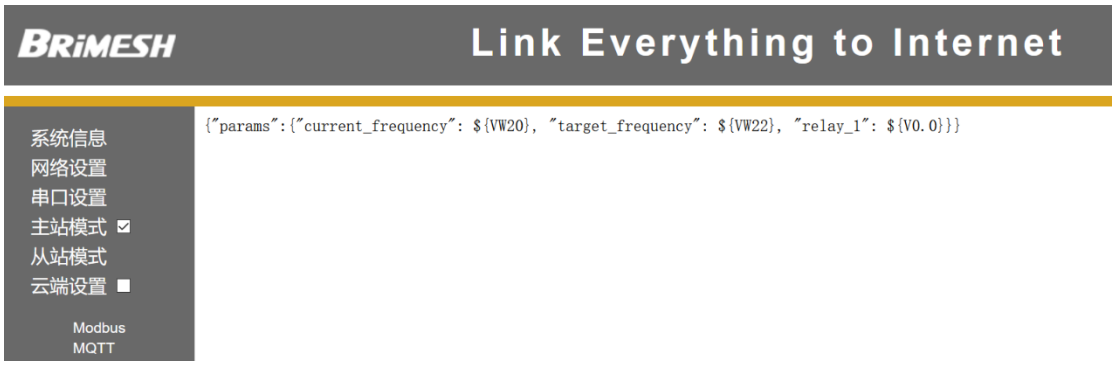
{"params":{"current_frequency": ${VW20}, "target_frequency": ${VW22}, "relay_1": ${V0.0}}}

然后在网关的 MQTT 数据格式定义页面中选择刚保存的文件，点击上传

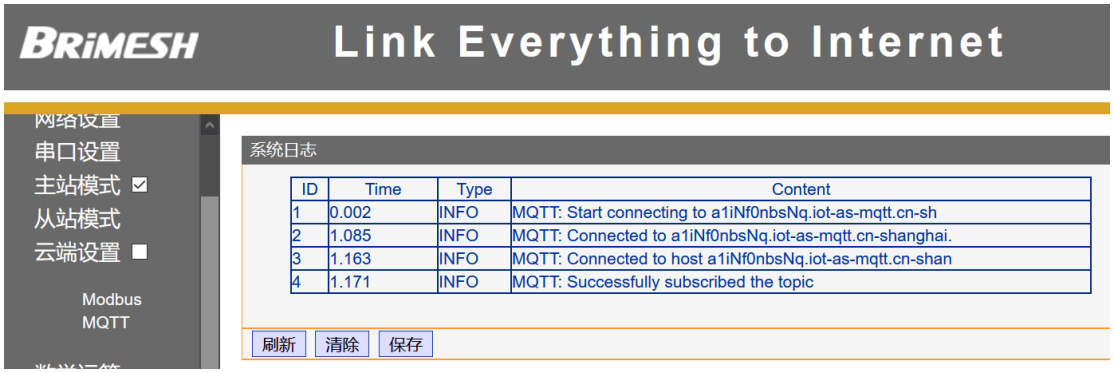




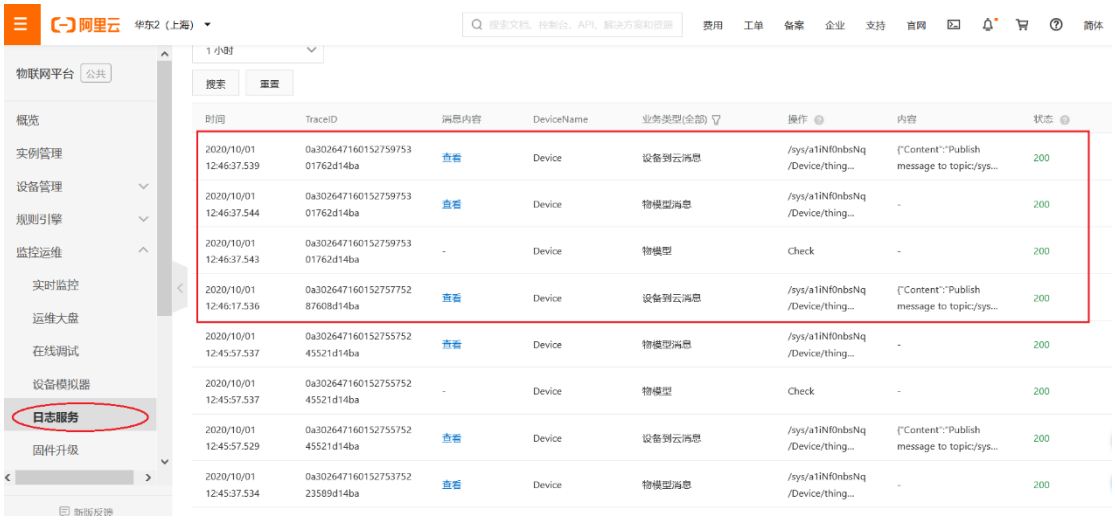
然后在 MQTT 数据格式定义页面点击【查看】按钮，就可以显示刚配置的数据模板。



重启网关之后，在系统日志可以看到连接成功的消息。



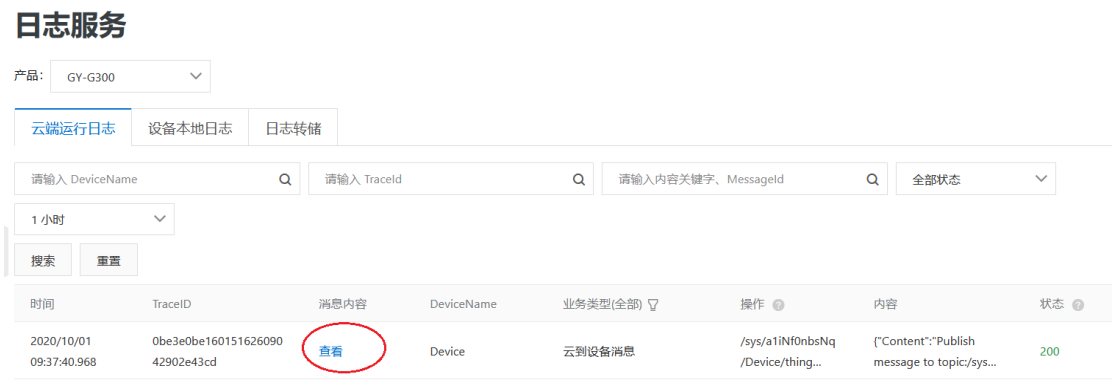
这时在云端的【日志服务】里面可以看到网关发布的数据日志

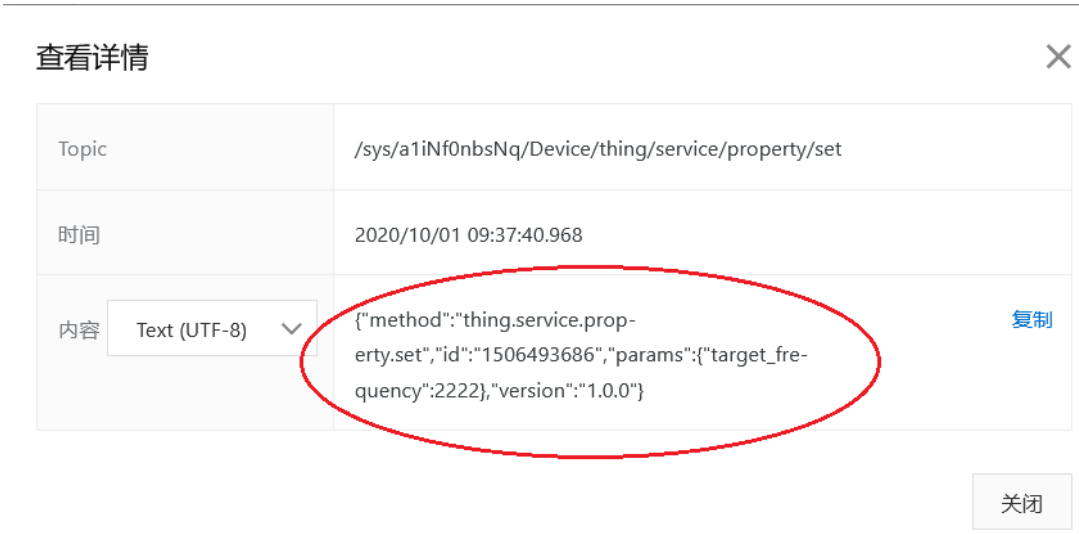


同时，可以在查看设备中的【物模型数据】中实时地看到当前数值

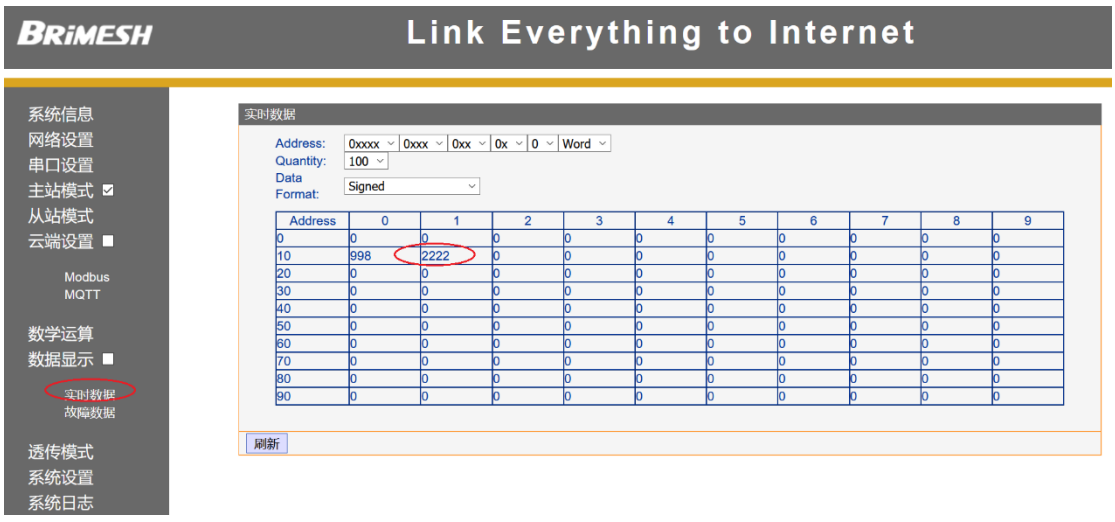
12.2.2.6 下发控制协议

通过日志服务查看刚下发的指令

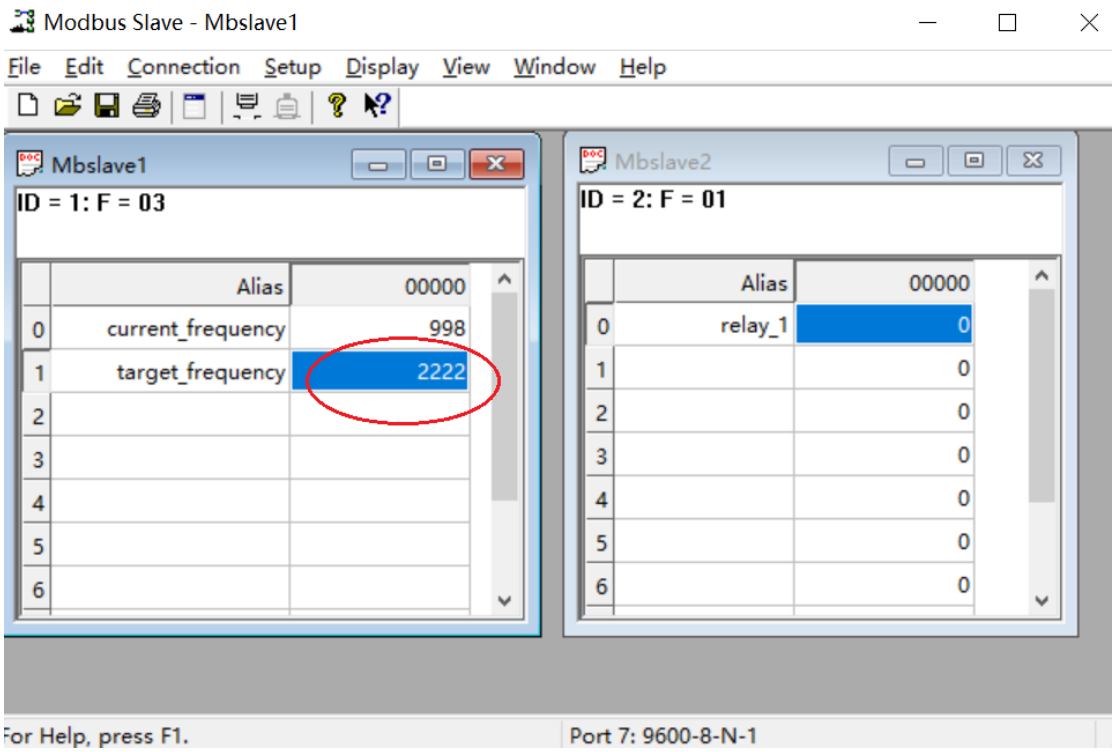




上图就是刚下发给网关的原始数据。网关收到云端发送的配置后，就会更改设定频率，如下图

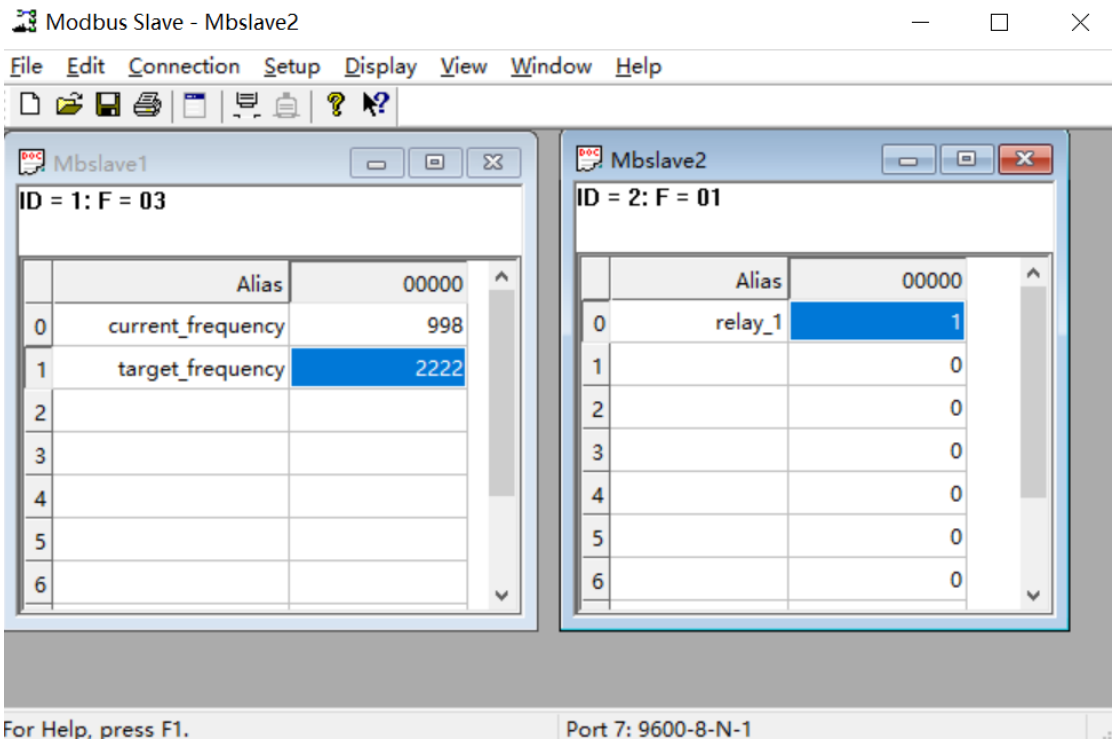


同时，我们的模拟器中的目标频率也被更改为 2222。如果是接实际的变频器，我们就可以看到变频器中的频率改变。


Modbus Slave - Mbslave1
File Edit Connection Setup Display View Window Help
Mbslave1
ID = 1: F = 03
Alias 00000
0 current_frequency 998
1 target_frequency 2222
2
3
4
5
6
Mbslave2
ID = 2: F = 01
Alias 00000
0 relay_1 0
1 0
2 0
3 0
4 0
5 0
6 0
For Help, press F1.
Port 7: 9600-8-N-1

类似，如果要控制继电器

点击发送指令，可以看到模拟器中的继电器状态变成 1。



12.3 通过 KEPSERVERMQTT 远程监控网关

KEPServer 的 mqtt 功能需要 mqtt broker 配合，因此先安装 mqtt broker 软件。

12.3.1 网关配置

12.3.1.1 模拟变频器和继电器模块

参考案例 1 的《模拟变频器和继电器模块》

12.3.1.2 [主站模式]下配置变频器和继电器模块

参考案例 1《[主站模式]下配置变频器和继电器模块》

12.3.1.3 MQTT 配置

MQTT to Cloud

Enable MQTT	☑
MQTT Broker:	www.brimesh.com
Port:	1883
ClientID:	2134safd
Keep Alive(s):	10
Publish Topic:	/mqtt_rx_data_channel/Device1
Command Topic:	/mqtt_rx_cmd_channel/Device1
Response Topic:	
Publish Interval(s):	10
AUTH Enable	☑
User Name:	test
Password:	test

保存

12.3.1.4 定义上报数据模板

将如下数据保存到一个文本文件中（一定要在同一行），例如 mqttdef.txt

{"current_frequency": ${VW20}, "target_frequency": ${VW22}, "relay_1": ${V0.0}, “mqtt_rx_cmd_channel.Device1.target_frequency” :${VW22}, “mqtt_rx_cmd_channel.Device1.relay_1” :${V0.0}}

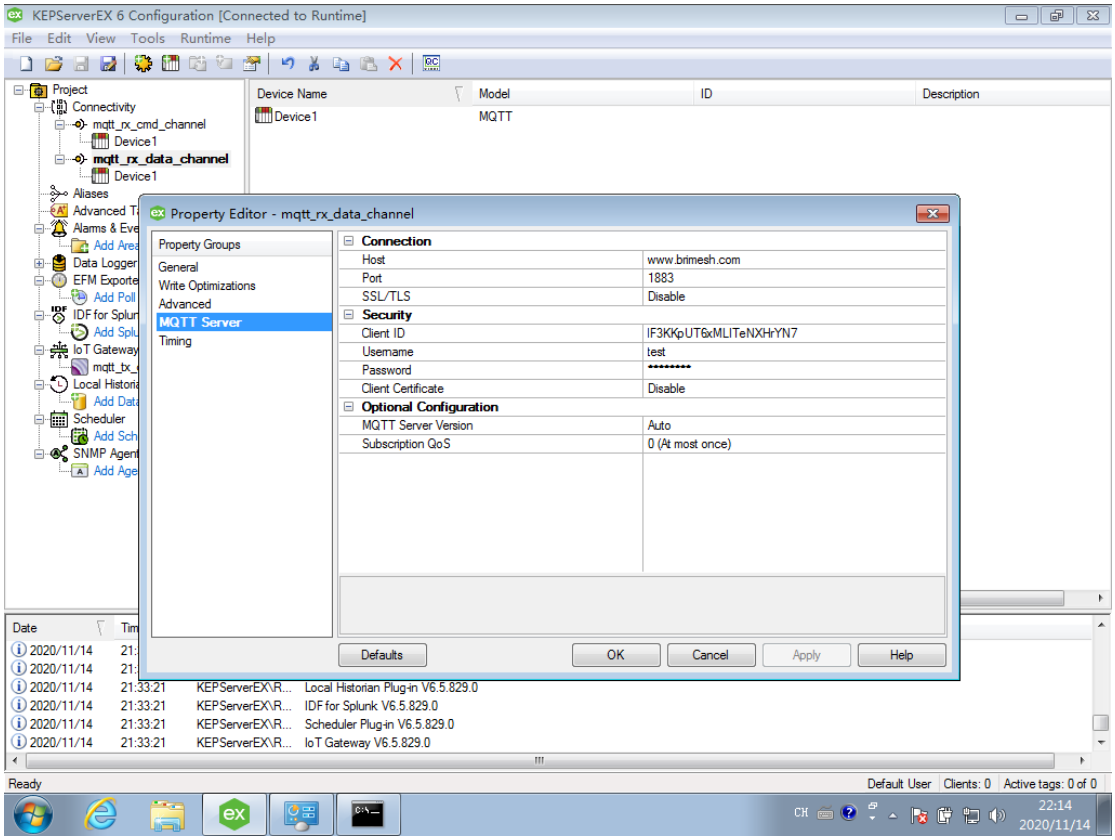
然后在网关的 MQTT 数据格式定义页面中选择刚保存的文件，点击上传。文本中的两个变量 mqtt_rx_cmd_channel.Device1.target_frequency 和 mqtt_rx_cmd_channel.Device1.relay_1 是 KEPServer 的下发控制客户配置变量。

12.3.2 KEPSERVER 配置

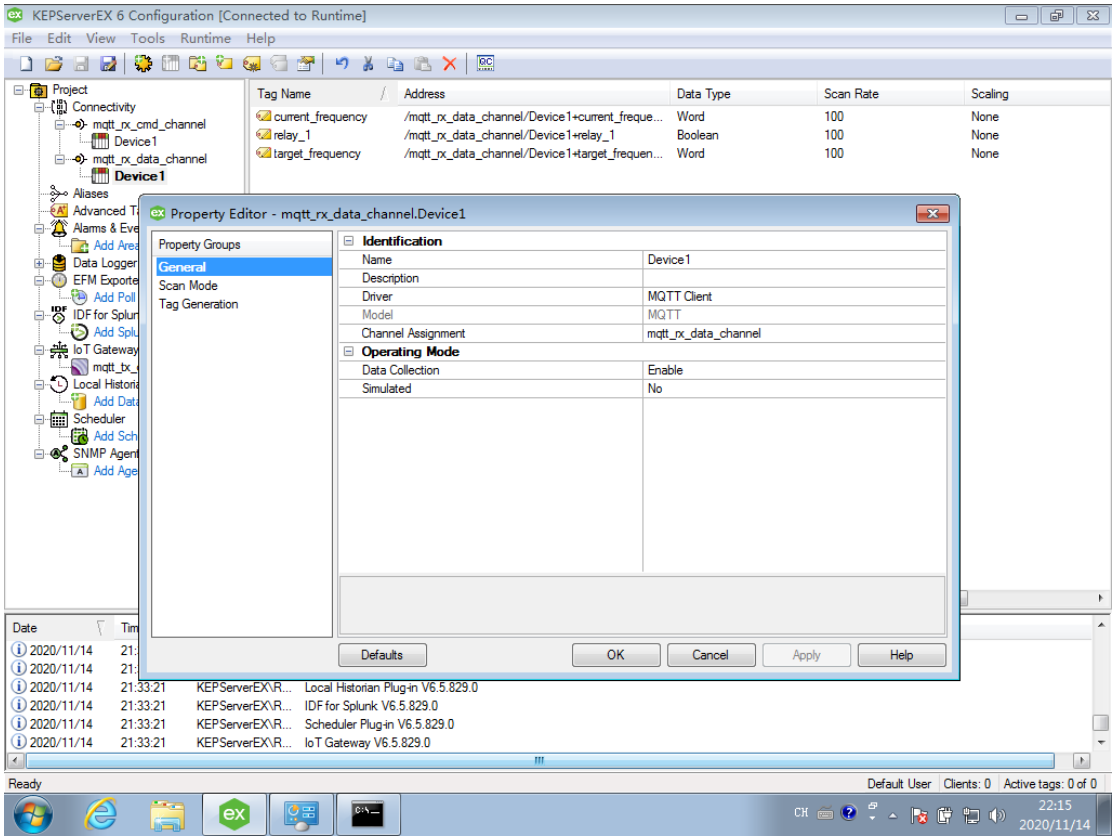
12.3.2.1 远程接收网关数据的配置

12.3.2.1.1创建 MQTT_RX_DATA_CHANNEL

创建基于 mqtt client driver 的 channel，该通道用于接收网关发送上来的数据。

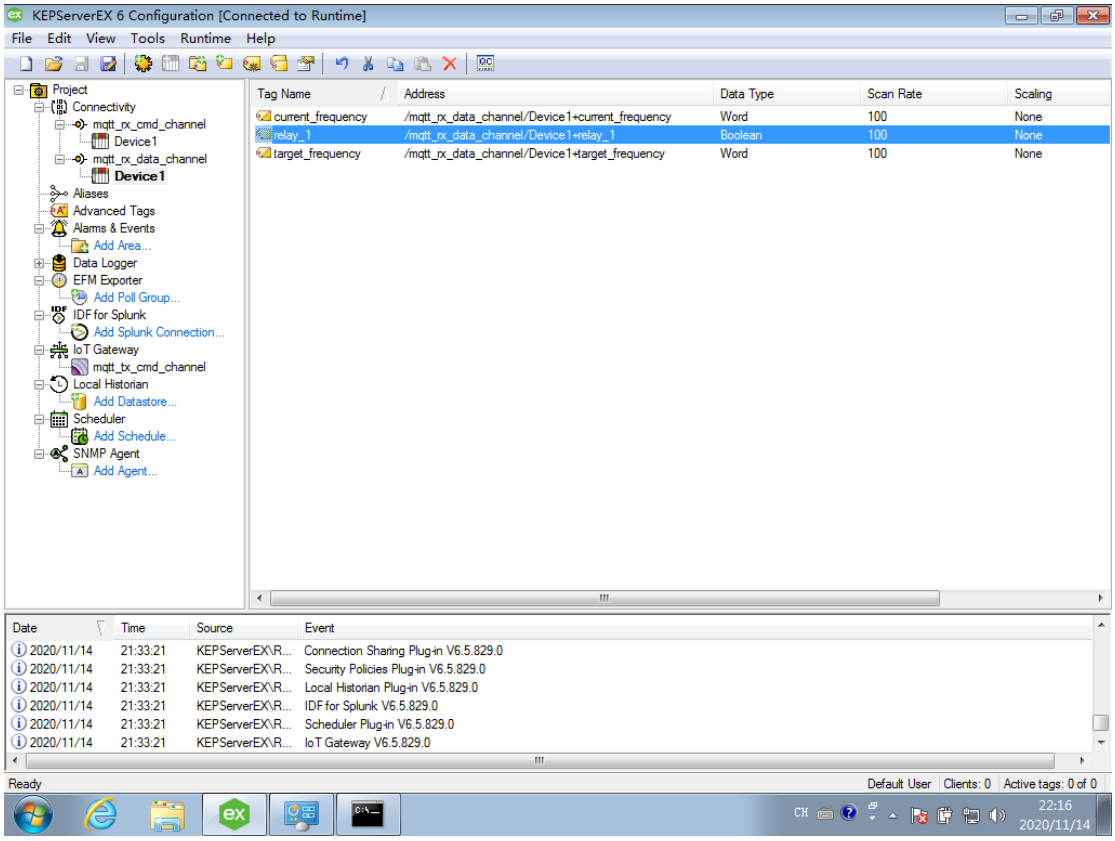

KEPServerEX 6 Configuration [Connected to Runtime]
File Edit View Tools Runtime Help
Project
Connectivity
mqtt_rx_cmd_channel
Device1
mqtt_rx_data_channel
Device1
Aliases
Device Name
Model
ID
Description
Device1
MQTT
Property Editor - mqtt_rx_data_channel
Property Groups
General
Write Optimizations
Advanced
MQTT Server
Timing
Connection
Host
www.brimesh.com
Port
1883
SSL/TLS
Disable
Security
Client ID
IF3KKpUT6xMLITeNXHrYN7
Username
test
Password
Client Certificate
Disable
Optional Configuration
MQTT Server Version
Auto
Subscription QoS
0 (At most once)
Defaults
OK
Cancel
Apply
Help
Date
2020/11/14 21:33:21 KEPServerEX\R... Local Historian Plug-in V6.5.829.0
2020/11/14 21:33:21 KEPServerEX\R... IDF for Splunk V6.5.829.0
2020/11/14 21:33:21 KEPServerEX\R... Scheduler Plug-in V6.5.829.0
2020/11/14 21:33:21 KEPServerEX\R... IoT Gateway V6.5.829.0
Ready
Default User
Clients: 0
Active tags: 0 of 0
22:14
2020/11/14

12.3.2.1.2创建 DEVICE1



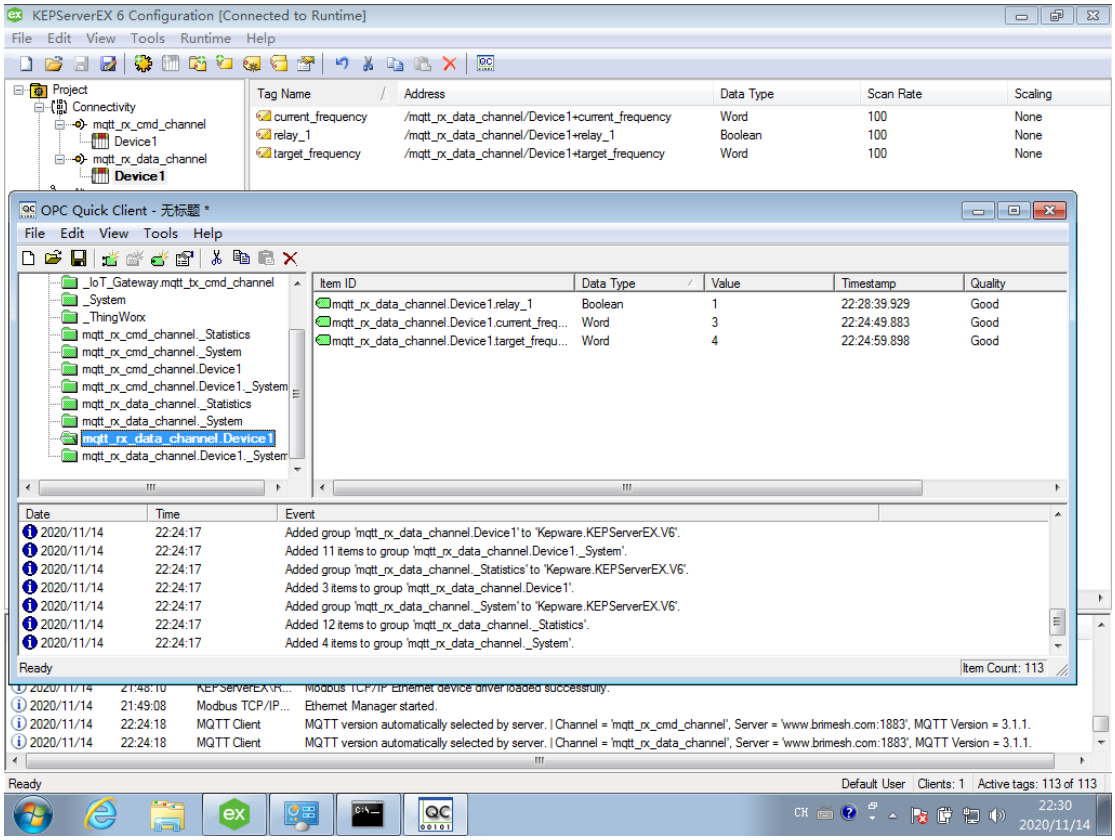
12.3.2.1.3创建变量

分别创建 current_frequency, target_frequency, relay_1，注意这几个 tag 的名字必须跟网关中的 mqtt 上报格式中的名称要一致。另外，图中的 Address 是通过 topic+tag 的格式。



12.3.2.1.4运行 OPC QUICK CLIENT 查看接收到的数据

在网关运行并上传数据后，可以看到三个变量都有数据上来。



12.3.2.2 控制网关的配置

KEPServer 往网关下发控制数据是通过 IoT Gateway 来实现的，官方有说明文档，可以参考：

https://www.kepware.com/getattachment/96bdb7bb-4f9a-4cfe-be30-ef048d16dd83/iot-gateway-manual.pdf

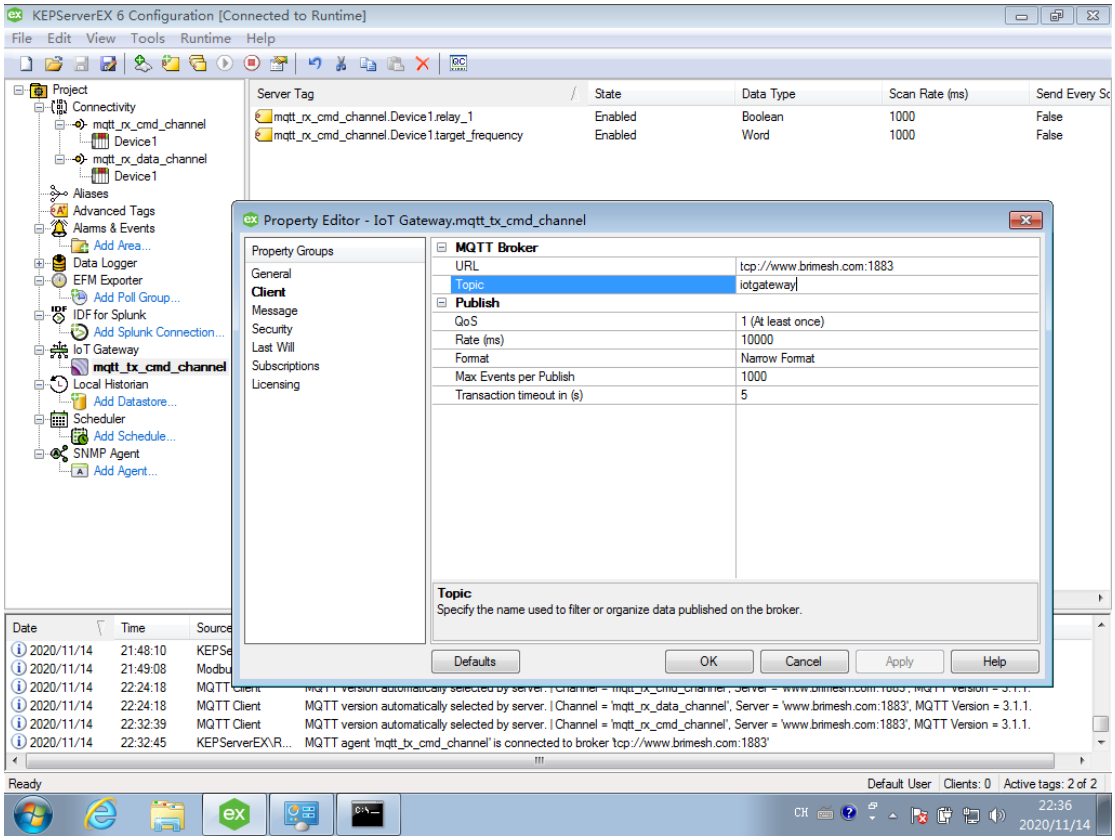
参考步骤如下：

12.3.2.2.1创建 MQTT_RX_CMD_CHANNEL

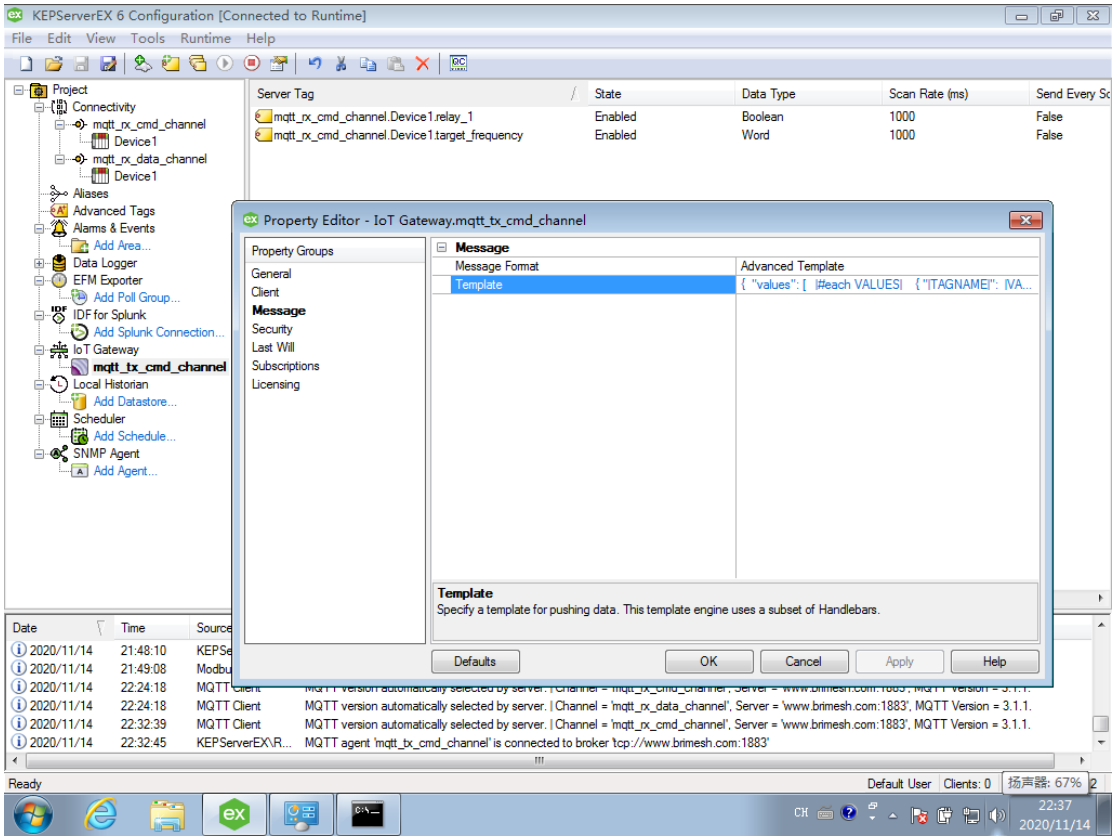
创建用于接收控制命令的通道，用户也可以选择 KEPServer 其它的任何通道创建数据，这里仅仅是为了演示，然后在 KEPServer 选择周期性的或者数据有变化时发送控制指令。

12.3.2.2.2创建 MQTT_TX_CMD_CHANNEL

在 IoT Gateway 下面创建发送指令的通道，特别留意发布的 Topic，网关会订阅这个 Topic，用于接收控制命令（在 cmd topic 填入）：



修改 message 的模板格式，**记得 Message Format 要选 Advanced Template**



Template 的模板如下：

```
{
    |#each VALUES|
    "|TAGNAME|":   |VALUE|,
    |/each|
}
```

该模板会把所有下发控制的变量遍历一遍生成 json 并 publish 给 iotgateway topic。网关通过订阅 iotgateway 接收控制指令。关于模板的具体介绍，请看 KEPServer 官方文档 iot-gateway-manual.pdf 中《Advanced Template Format》介绍，摘取如下

Advanced Template Data Format

The IoT Gateway pushes data in a standard JSON format via the REST and MQTT Clients. This format may then be consumed by the third party endpoint and broken down in an appropriate way. The REST Client agent and MQTT agent have the ability to use an advanced template for pushing data. This template engine uses a subset of Handlebars. The Advanced Template allows more complete control over the payload format. In addition to JSON, formats like XML and CSV can be generated using this template. The Advanced Template is a drop down selection in the Message Format field.

Variables

Like the Standard Template, variables can be inserted into the template representing publish time and data changes. Top-level variables like SERVERDATE and SERVERTIMESTAMP can be inserted anywhere in the tem plate.

|SERVERTIMESTAMP|	Time of publish, represented as the number of milliseconds since January 1st, 1970, midnight
|SERVERDATE|	Date and time of publish as human-readable string

The VALUES variable represents a list of every data change in the publish. Each data change contains vari-ables for the tag name, value, quality, and timestamp.

|TAGNAME|	The name of the tag ('Channel.Device.Tag')
|VALUE|	The value of the tag
|QUALITY|	"true" if the tag was read successfully, "false" if the tag could not be read (e.g. a con-nection issue)
|TIMESTAMP|	The time at which the tag was read, represented as the number of milliseconds since January 1st, 1970, midnight.

Syntax

The 'each' keyword allows text to be generated for each item update. The template inside the 'each' block is evaluated once for every item update in the publish. Depending on the format, this can be used to make each update a JSON object or a CSV row for example.

```
|#each VALUES|
  |TAGNAME|, |VALUE|, |QUALITY|, |TIMESTAMP|,
|/each|
```

Additional Syntax

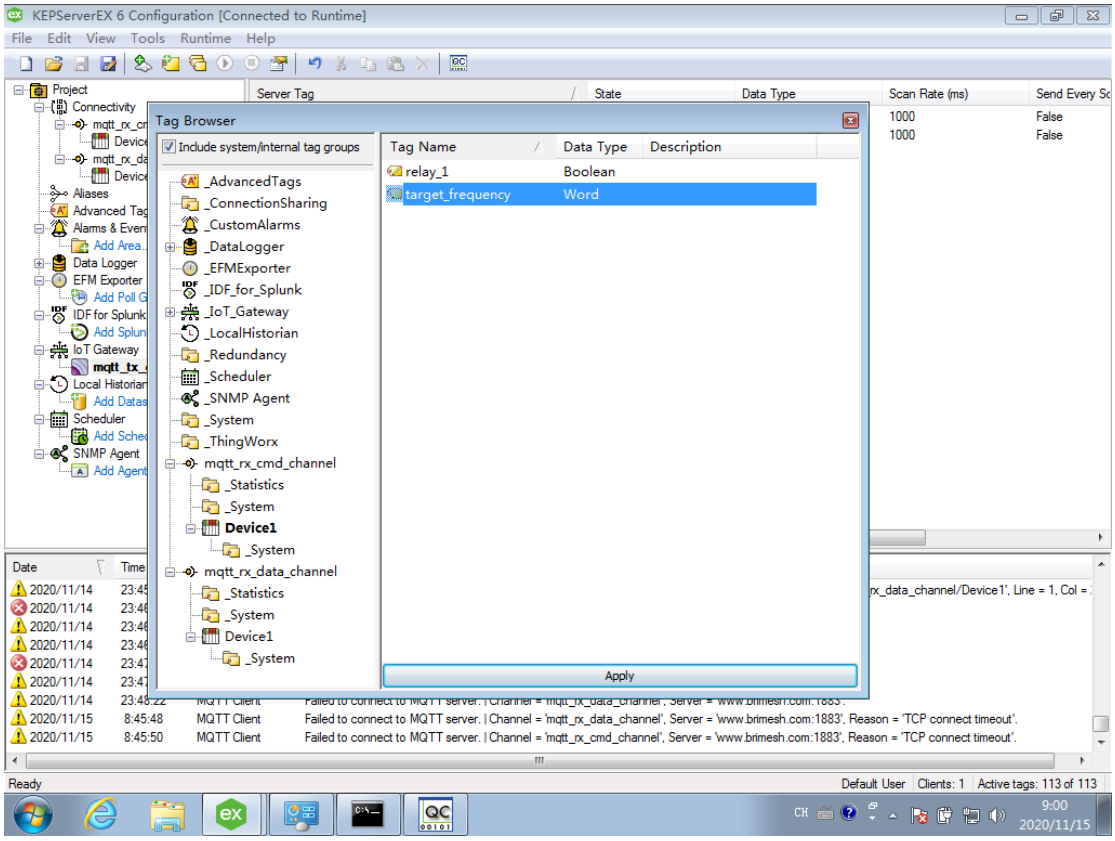
The default advanced template contains the following syntax:

```
|#unless @last|,|/unless|
```

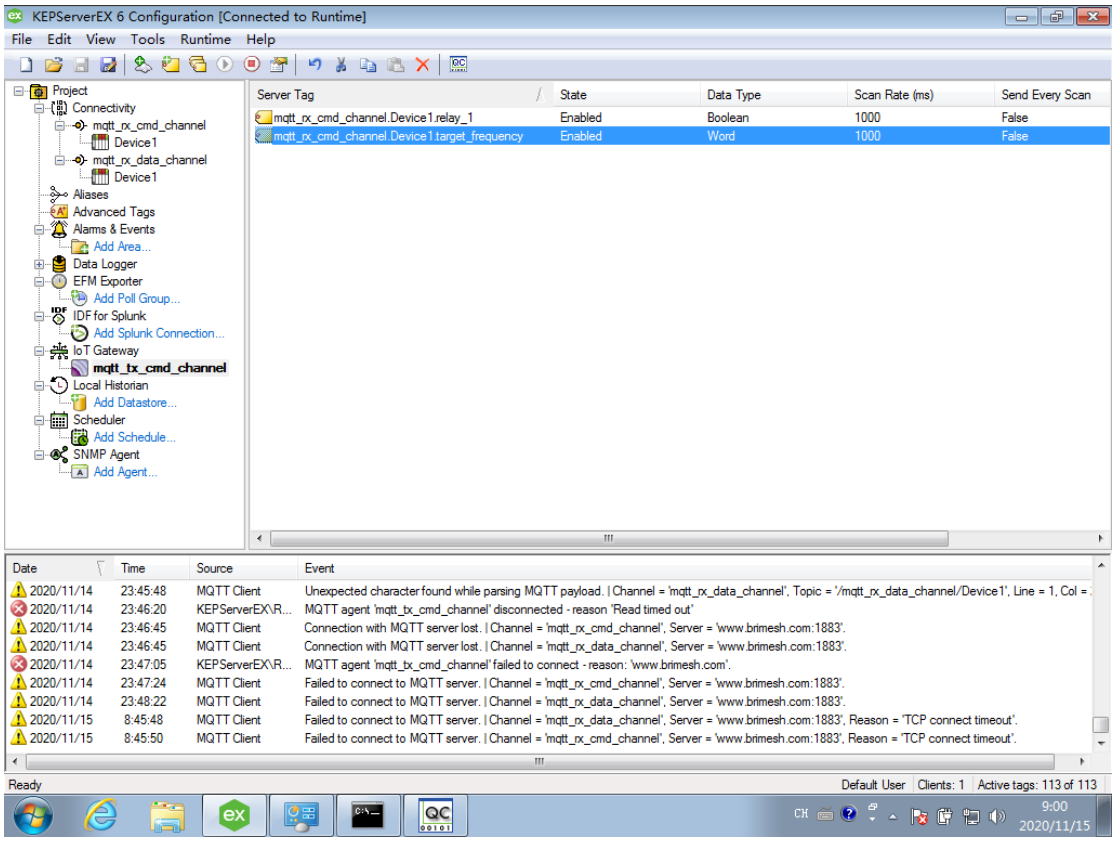
This can be read as: "Unless this is the last item in the list, insert a comma."
This effectively eliminates the trailing comma for the last item in a list. While this can be omitted in most cases, it is necessary if the consumer of the payload treats trailing commas as a syntax error.

12.3.2.2.3创建 SERVER TAG

Server tag 是在已经创建的 tag 里面选择，然后 KEPServer 的 IoT Gateway 模块会根据设置选择周期性的或者数据有变化时把控制数据发布给网关。

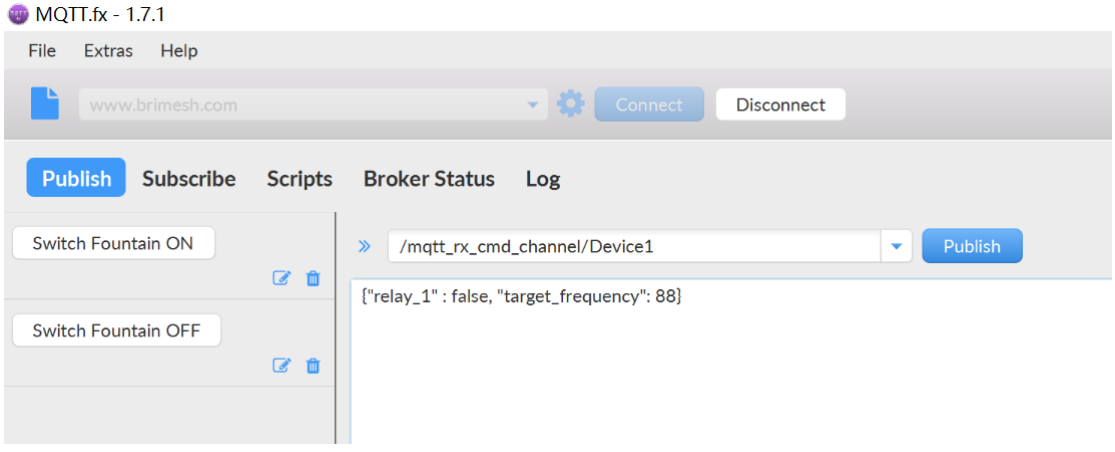


在 mqtt_rx_cmd_channel 通道里面选择需要控制的变量，如下图:

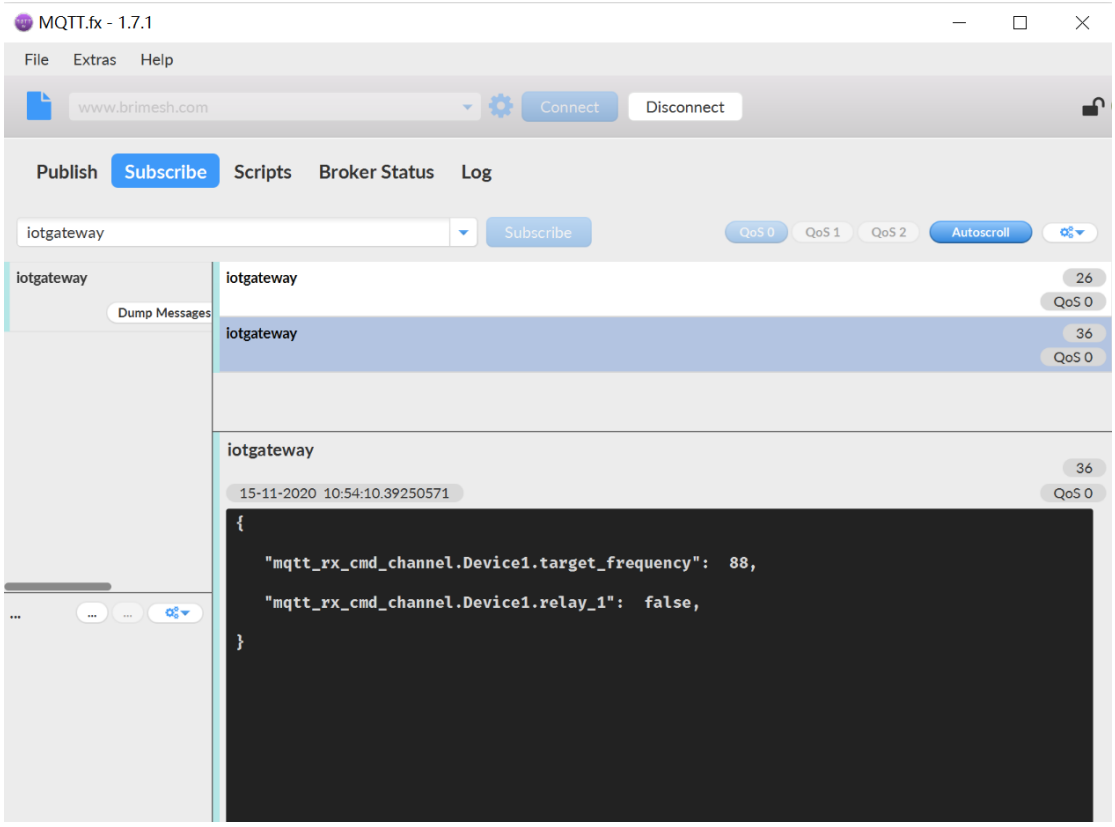


12.3.2.2.4发送控制指令

使用 mqttfx 往 mqtt_rx_cmd_channel 主题发送控制命令，如下图



同时订阅 iotgateway 主题，就可以看到 KEPServer 发送下来的控制指令。



在网关的系统日志里面可以看到下发的数据，如下图：

系统日志

ID	Time	Type	Content
1	0.002	INFO	MQTT: Start connecting to www.brimesh.com:1883
2	4.004	ERROR	MQTT: Failed to connected to www.brimesh.com:1883
3	9.005	INFO	MQTT: Start connecting to www.brimesh.com:1883
4	9.146	INFO	MQTT: Connected to www.brimesh.com:1883
5	9.169	INFO	MQTT: Connected to host www.brimesh.com
6	9.191	INFO	MQTT: Successfully subscribed the topic
7	474.977	INFO	MQTT RECV: message len = 245
8	474.977	INFO	MQTT: VW22=600
9	474.977	INFO	MQTT: V0.0=1
10	474.978	INFO	MQTT: VW22=600
11	474.978	INFO	MQTT: V0.0=1
12	604.839	INFO	MQTT RECV: message len = 127
13	604.839	INFO	MQTT: VW22=88
14	604.839	INFO	MQTT: V0.0=0

刷新 清除 保存

同时，在【实时数据】页面可以查看到写下来的数值

实时数据

Address: 0xxxx 0xxx 0xx 0x 0 Word

Quantity: 100

Data Format: Signed

Address	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
10	0	88	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0

刷新

12.4 通过 THINGSBOARD 远程监控网关

ThingsBoard 是一个开源的物联网平台，用于数据收集、处理、可视化展示以及设备管理。ThingsBoard 使用行业标准物联网协议（MQTT，CoAP 和 HTTP）实现设备连接，并支持云和本地部署。

ThingsBoard 结合网关可以做什么？Thingsboard 配合网关可以实现将现场采集的的设备数据储存到云端数据库中，并且提供多种部件库，用于展现当前以及历史数据，以及远程控制设备的界面。同时对于高级用户，可以调用 ThingsBoard 提供的 restful 接口来自己开发前端界面。

本例子使用 ThingsBoard，通过 mqtt 协议收集网关上报的数据，并存储到 pgsql 数据库中，同时通过调试工具 curl 远程调用 ThingsBoard 的 restful 接口，以此达到远程控制网关下的 Modbus RTU 设备。

12.4.1 THINGSBOARD 配置

12.4.1.1 THINGSBOARD 安装

Thingsboard 官方有很详细的安装说明文档，这里以 Windows 平台作为测试，具体文档请参考如下链接，一步步跟着操作就可以完成安装

https://thingsboard.io/docs/user-guide/install/windows/

需要注意，安装 pgsql 的时候，语言选中文可能会有安装问题，如果是这样请选择 English 语言。

Thinsboard 的 github 库链接：

https://github.com/thingsboard/thingsboard

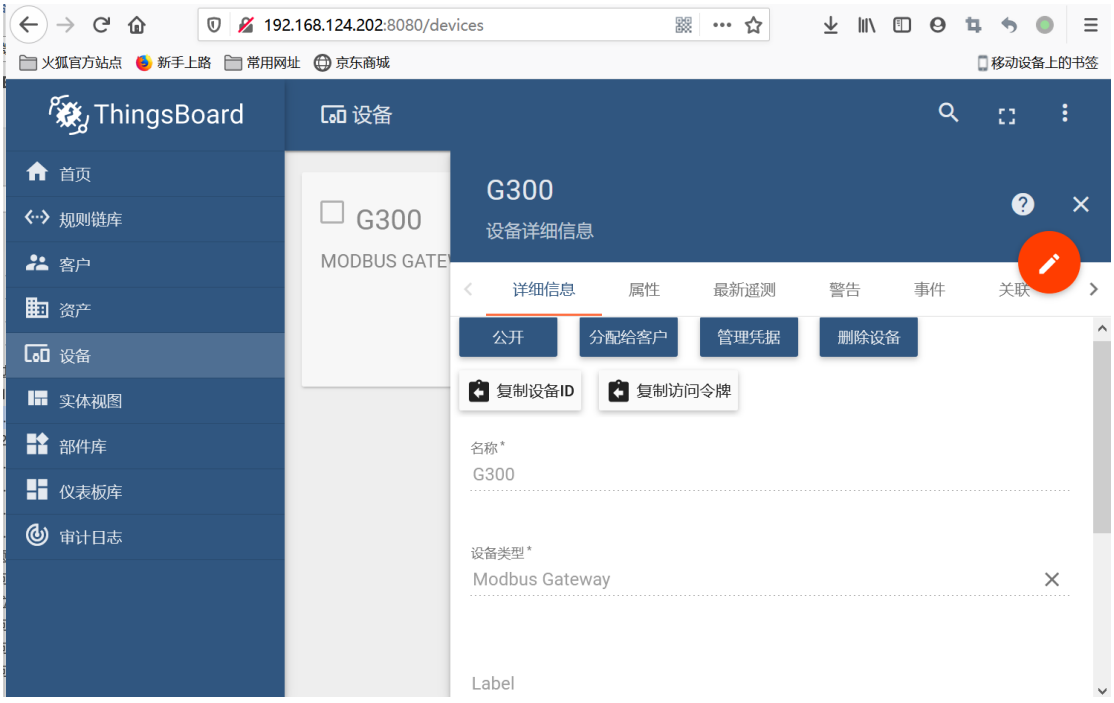
Thinsboard 的正式发布版本链接（注意，v3.1.1 接收 mqtt 有问题，请使用 v2.5.4）：

https://github.com/thingsboard/thingsboard/releases

在安装完之后，登录 http://localhost:8080

以默认管理员账户 sysadmin@thingsboard.org / sysadmin 登录并创建一个租户账号，用于创建设备。

12.4.1.2 创建设备



12.4.1.3 THINGSBOARD 的 MQTT 通讯

ThingsBoard 的 Mqtt 通讯在官网中有很详细的说明，链接如下：

https://thingsboard.io/docs/reference/mqtt-api/

网关通过 v1/devices/me/telemetry 这个 Topic，以 json 的格式来上报数据，通过 v1/devices/me/rpc/request/+这个 Topic 来下发控制指令。也就是说，网关订阅 v1/devices/me/rpc/request/+来接收控制指令，这里符号+是一个通配符，网关实际收到的数据来自于由 v1/devices/me/rpc/request/和一个 id 号组成的 Topic，例如，第一个指令来自 Topic v1/devices/me/rpc/request/1，第二个指令就是来自 v1/devices/me/rpc/request/2，以此类推。而指令回复是通过 v1/devices/me/rpc/response/$request_id 这个 Topic 发送的，也就是这个 Topic 由 v1/devices/me/rpc/response/和指令 Topic 后面的 id 组成，成双成对的，例如：

指令 Topic： v1/devices/me/rpc/request/888

回复 Topic：v1/devices/me/rpc/response/888

12.4.2 网关配置

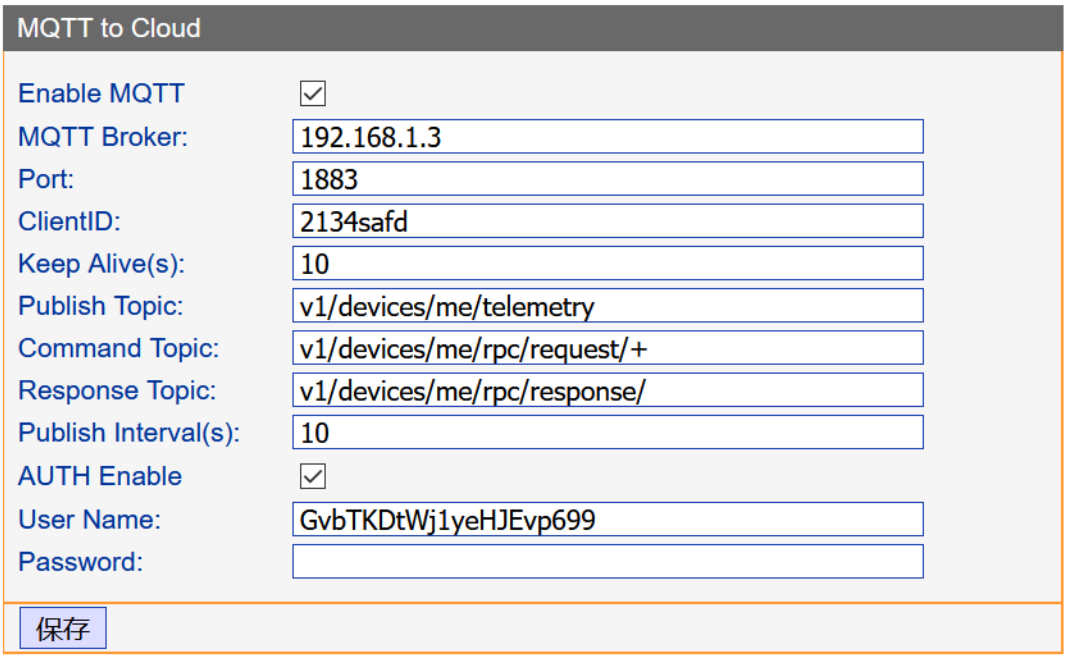
12.4.2.1 模拟变频器和继电器模块

参考案例 1 的《模拟变频器和继电器模块》

12.4.2.2 [主站模式]下配置变频器和继电器模块

参考案例 1《[主站模式]下配置变频器和继电器模块》

12.4.2.3 MQTT 配置



图中的用户名是通过 ThingsBoard 页面中设备的【复制访问令牌】得到，唯一对应于创建的每个设备，密码为空。ClientID 用户自己设置，可以用序列号，这样保证不重复。

12.4.2.4 定义上报数据模板

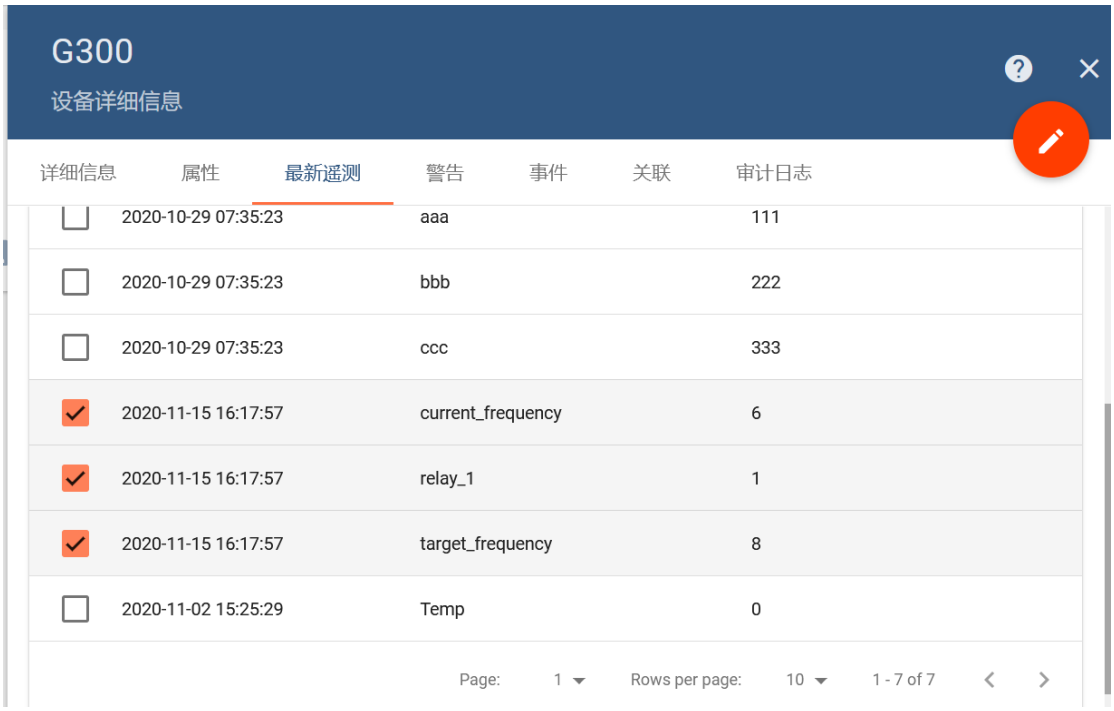
将如下数据保存到一个文本文件中（一定要在同一行），例如 mqttdef.txt

{"current_frequency": ${VW20}, "target_frequency": ${VW22}, "relay_1": ${V0.0}}

然后在网关的 MQTT 数据格式定义页面中选择刚保存的文件，点击上传。

12.4.3 查看变频器当前频率的历史曲线

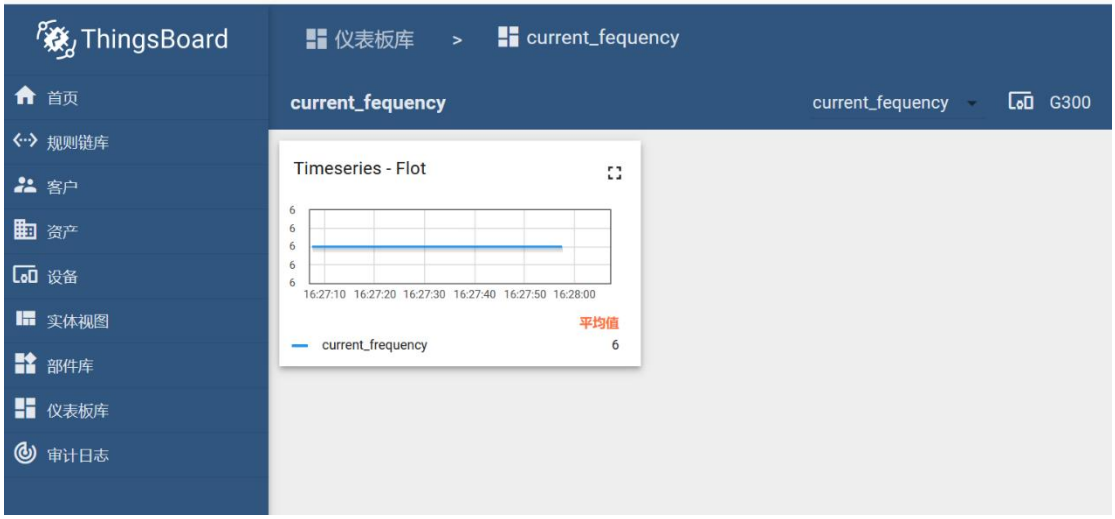
在设备的【最新遥测】页面可以看到当前设备的最新数据，如下图：



同时，也可以对当前数据进行组态，查看历史曲线，例如选 current_frequency，

点击右上角【在部件上显示】，显示 Charts 部件，

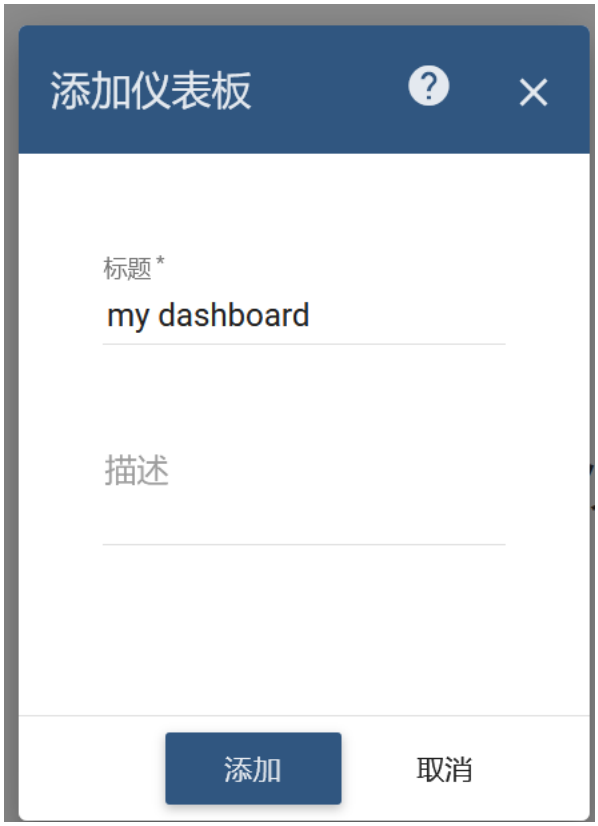
然后在仪表盘库里面就可以看到当前频率的历史曲线。

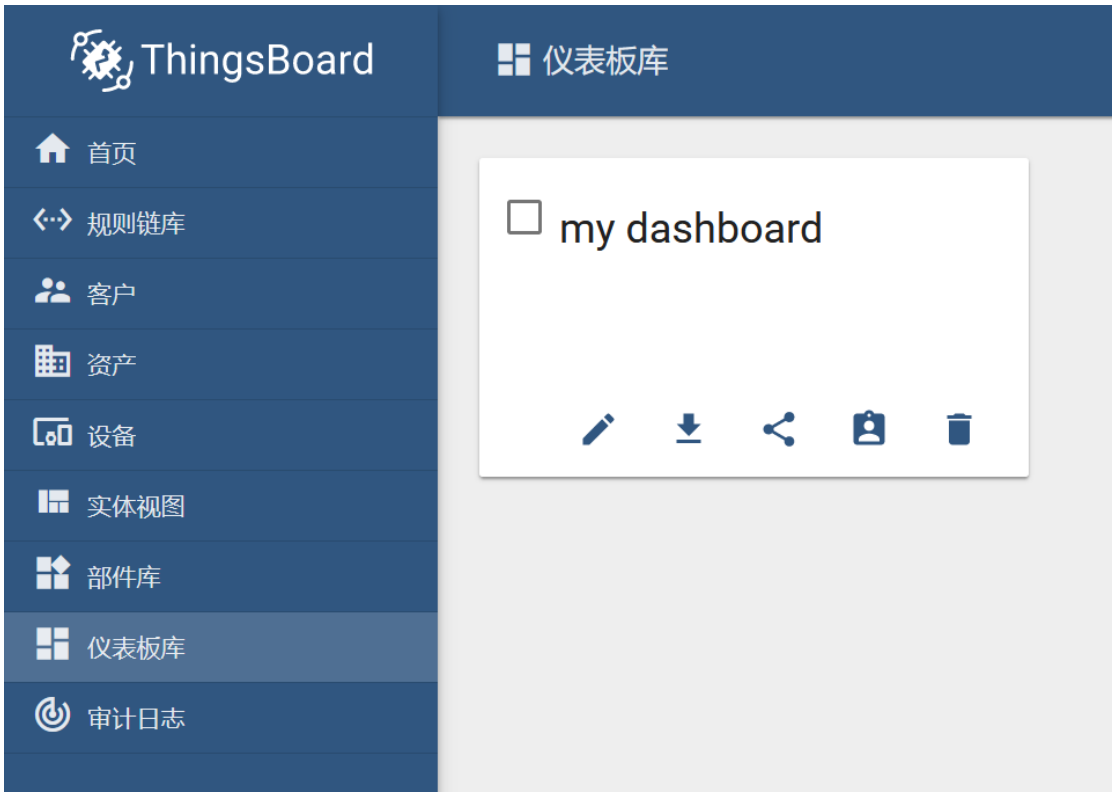


通过手机也可以查看历史曲线。如果提供给用户使用，需要由租户创建 custom 账户，然后分享给 custom，客户就可以登录查看。

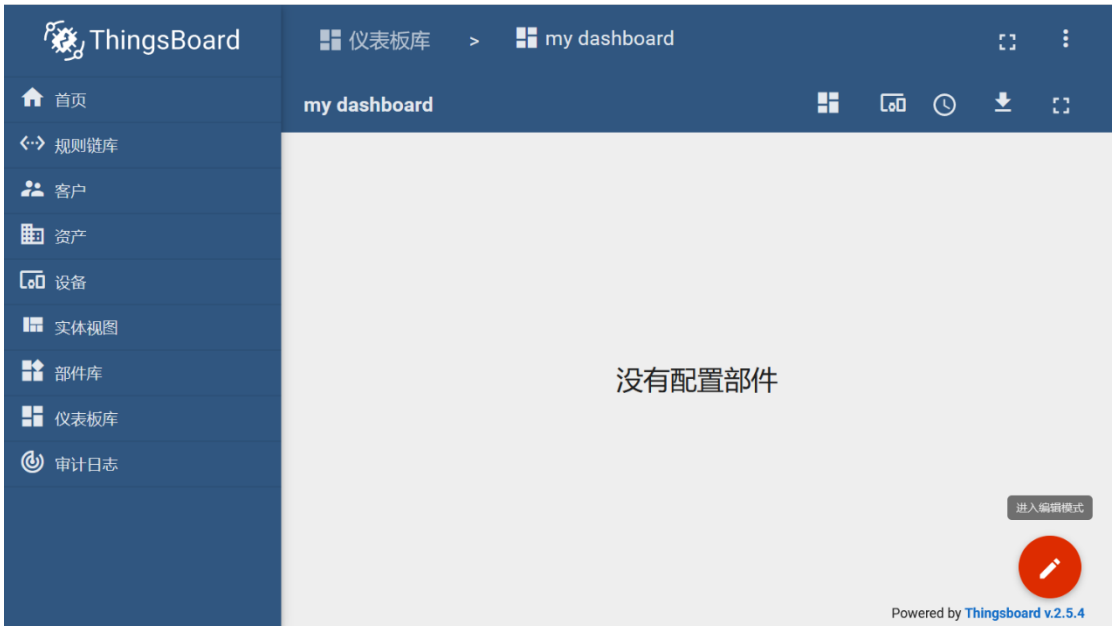
12.4.4 远程控制继电器，设置频率并查看频率曲线

在仪表盘库页面选择【添加新的仪表盘】按钮（右下角红色圆圈），

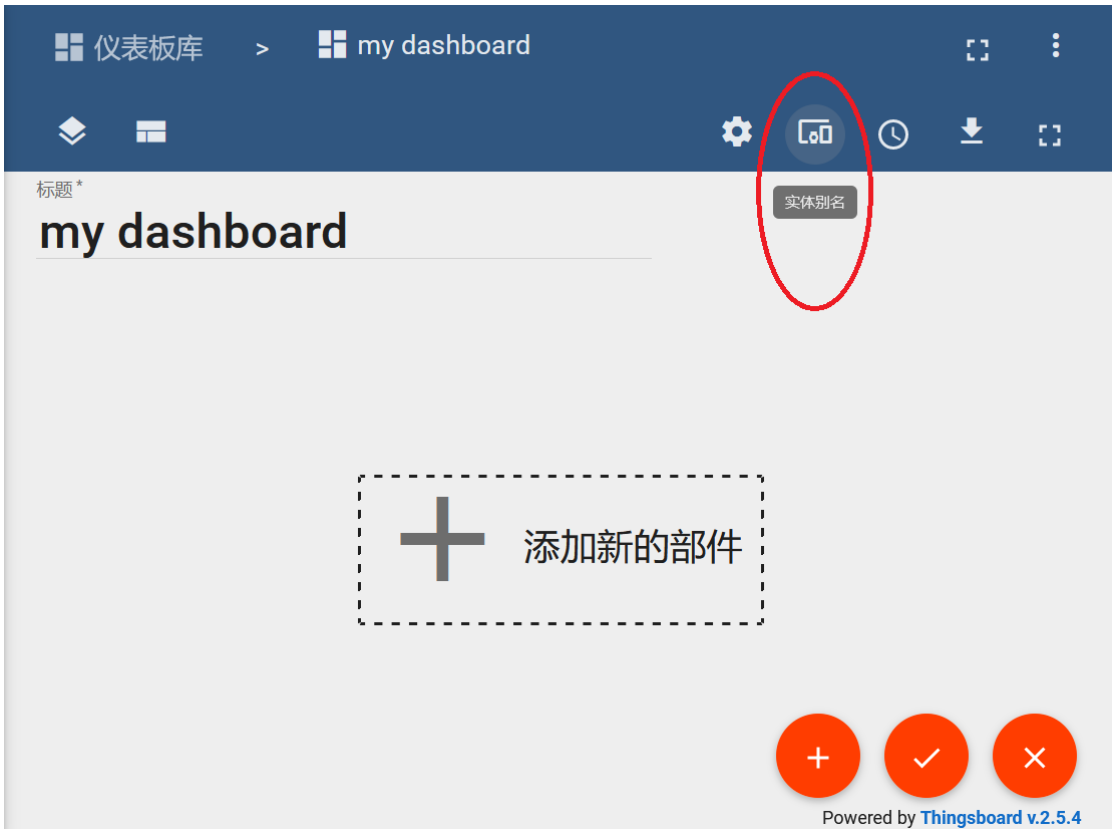

添加仪表板
标题 *
my dashboard
描述
添加
取消


ThingsBoard
仪表板库
首页
规则链库
客户
资产
设备
实体视图
部件库
仪表板库
审计日志
my dashboard

进入编辑模式



设置实体别名



添加实体别名


实体别名
别名
实体过滤
解决为多实体
添加别名
保存
取消

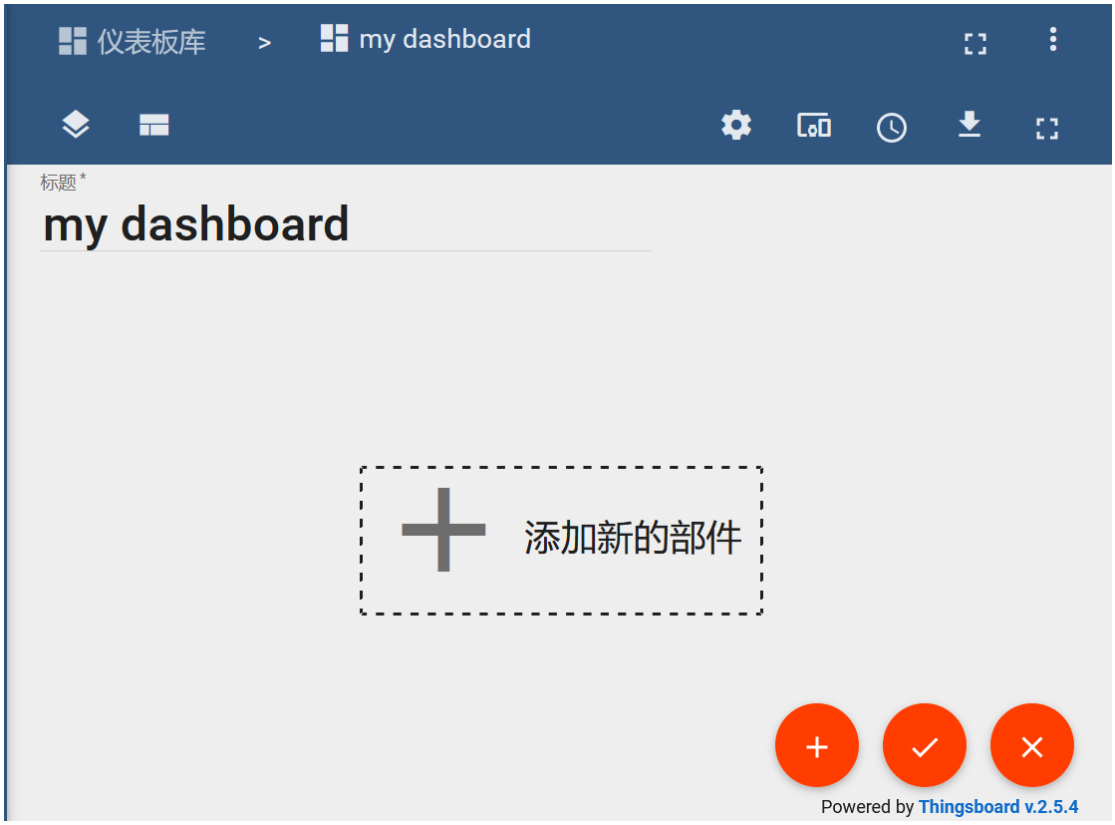

添加别名
别名 *
device1
解决为多实体
过滤类型 *
设备类型
设备类型 *
Modbus Gateway
名称前缀
添加
取消

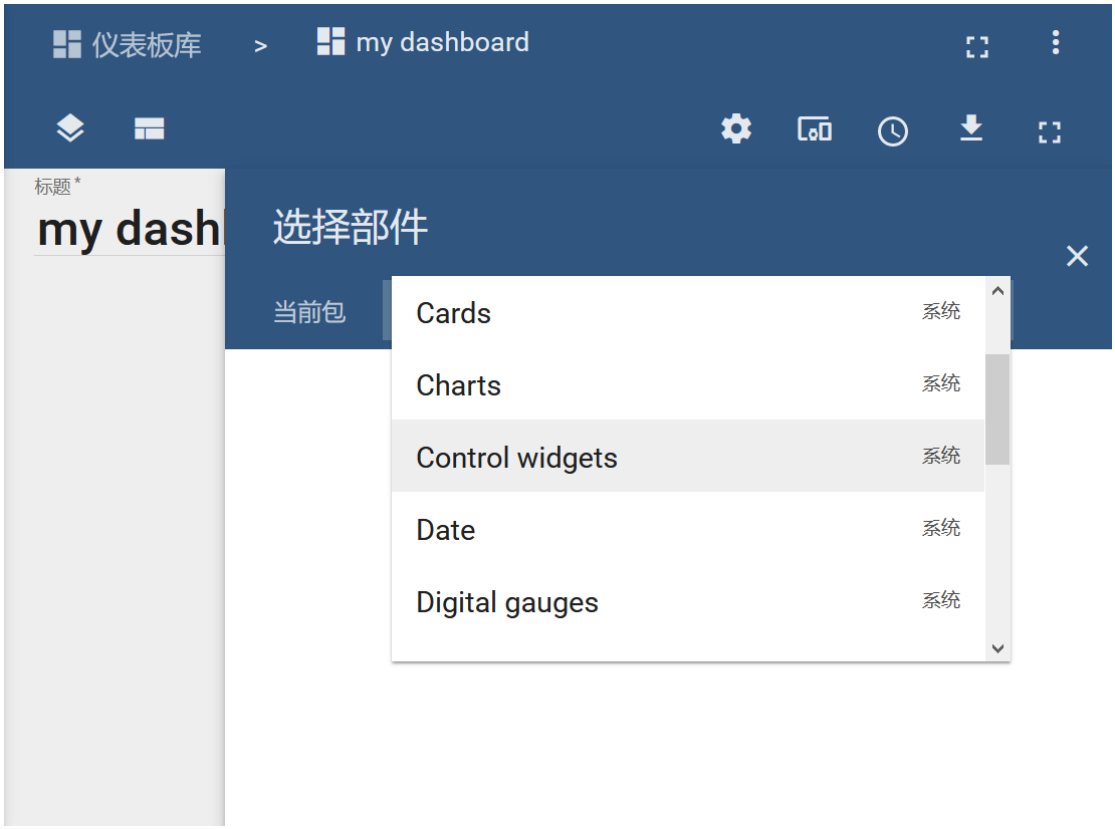
添加完并保存


实体别名
别名
实体过滤
解决为多实体
1.
device1
类型为 'Modbus Gateway' 的设备
添加别名
保存
取消

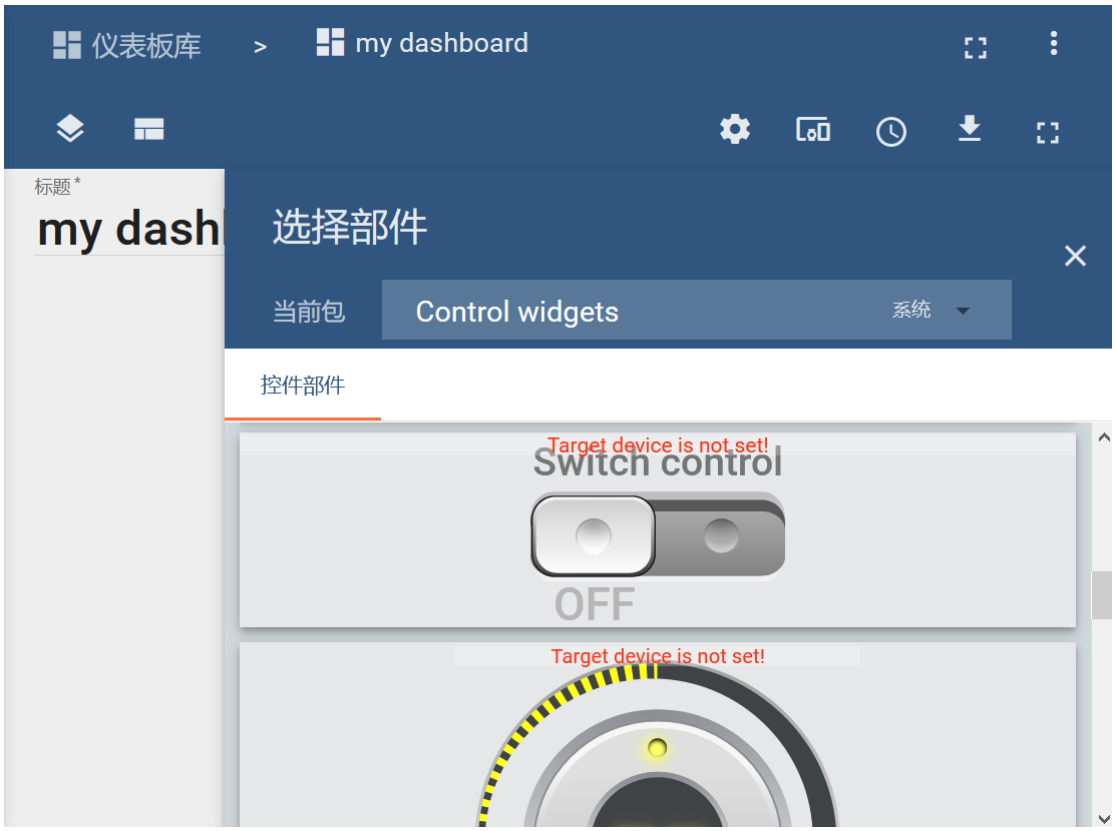
下面开始添加每个部件（Widget）

12.4.4.1 添加开关按钮





选择 Switch Control



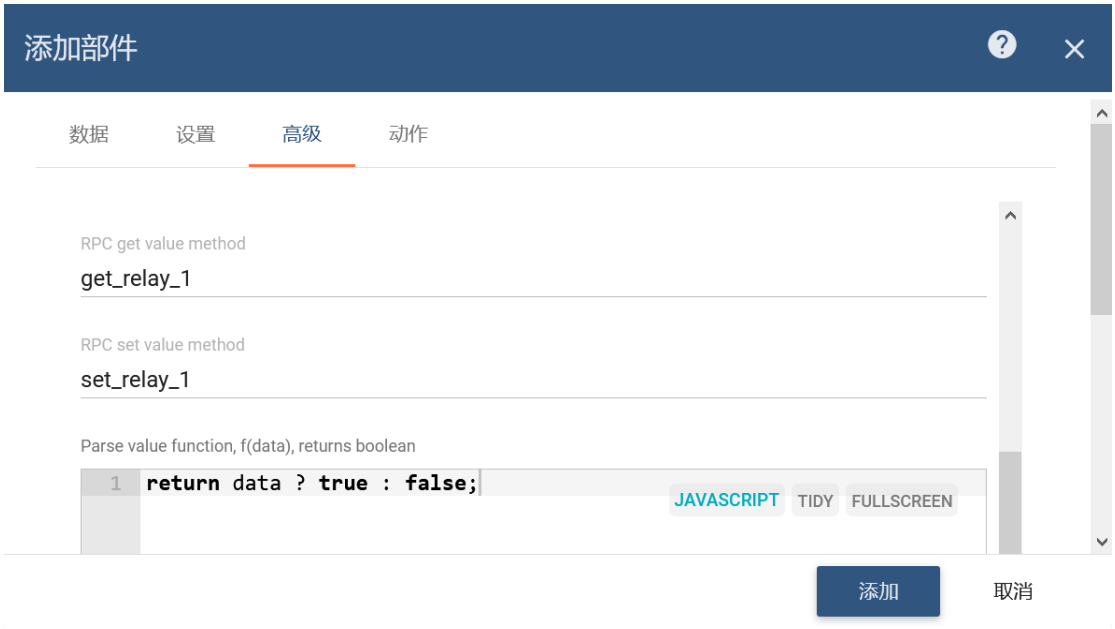
在高级设置里面，这两个参数 RPC get value method 和 RPC set value method 很重要，涉及到跟网关通讯。其中，RPC get value method 是 ThingsBoard 用来获取网关某个参数数值的通讯方式，该通讯方式基于 mqtt json，网关收到问询后会回复当前数值。同样，RPC set value method 是 ThingsBoard 设置网关某个参数的通讯方式。在跟网关通讯时，这两个参数必须遵循如下格式：

RPC get value method: get_$TAGNAME

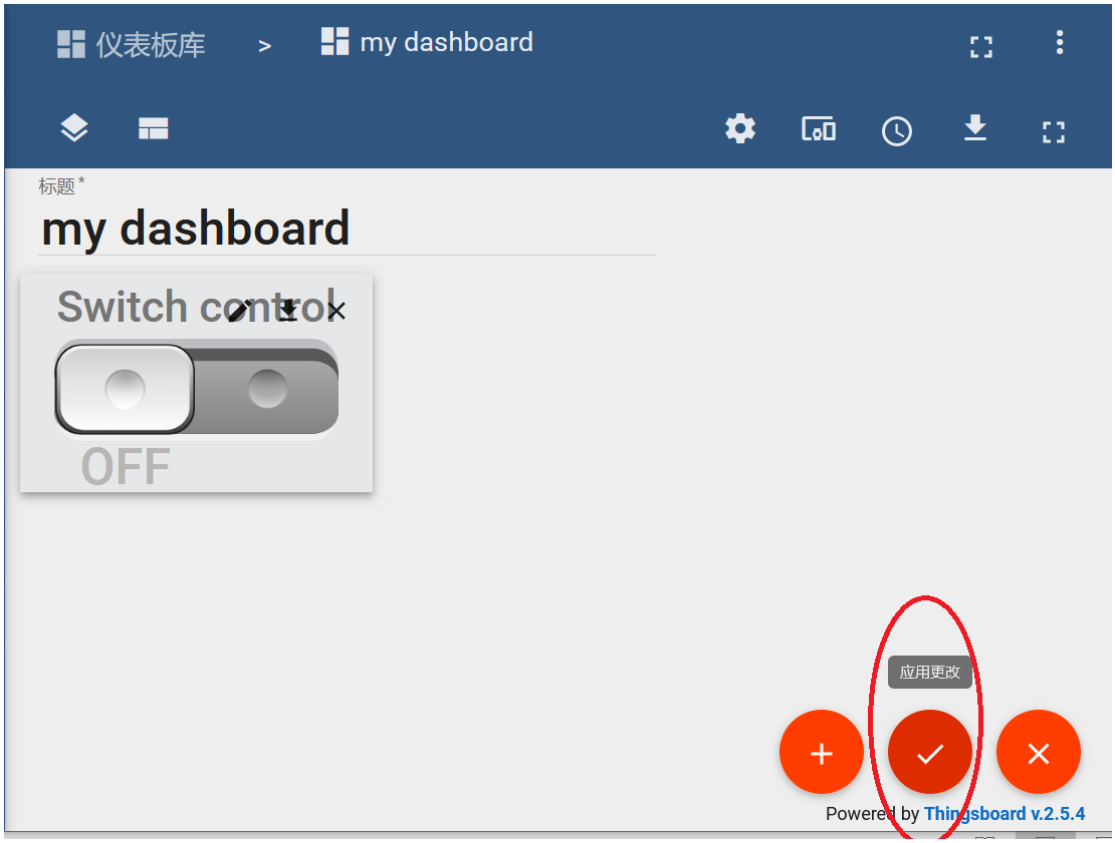
RPC set value method: set_$TAGNAME

其中，$TAGNAME 就是网关上报数据模板中的某个参数，这里继电器用的是 relay_1，所以这两个参数用的是 set_relay_1 和 get_relay_1。如果用其它控制部件，比如设置目标频率，那么对应这两个参数就是 set_target_frequency 和 get_target_frequency。我们再看一下网关的 mqtt 上传模板：

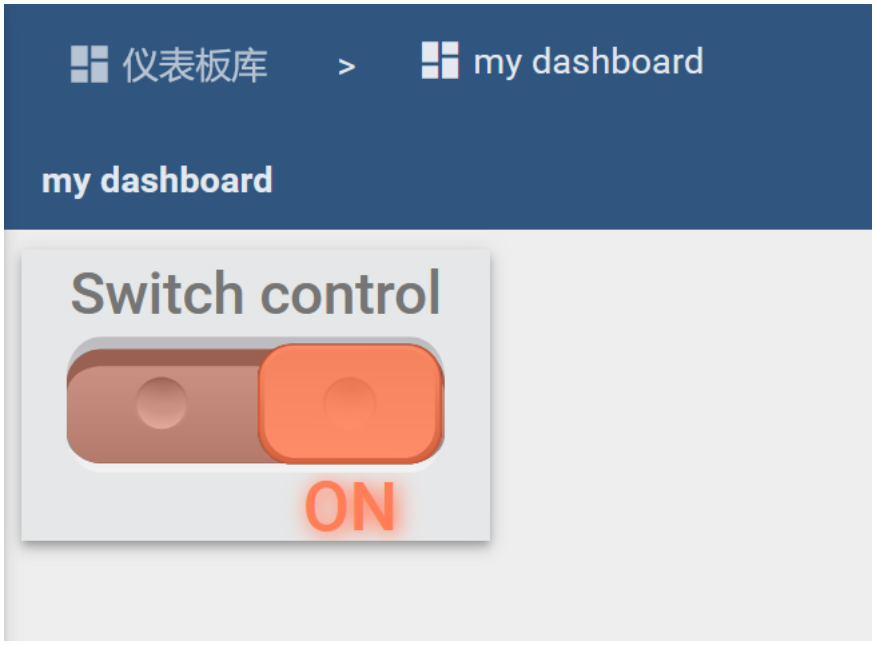
{"current_frequency": ${VW20}, "target_frequency": ${VW22}, "relay_1": ${V0.0}}


添加部件
数据
设置
高级
动作
RPC get value method
get_relay_1
RPC set value method
set_relay_1
Parse value function, f(data), returns boolean
1
return data ? true : false;
JAVASCRIPT
TIDY
FULLSCREEN
添加
取消

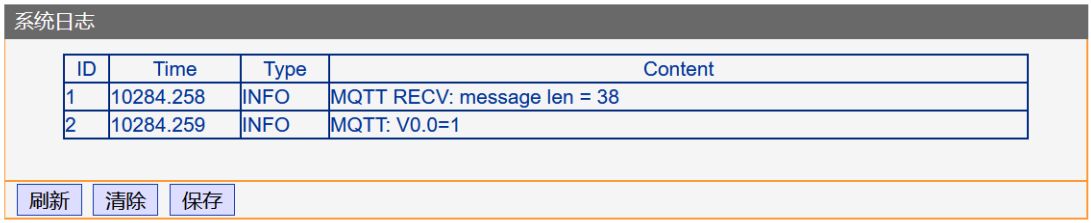
添加完注意应用更改，


仪表板库
>
my dashboard
标题*
my dashboard
Switch control
OFF
应用更改
Powered by Thingsboard v.2.5.4

这时按下按钮就可以看到网关下的继电器会有打开



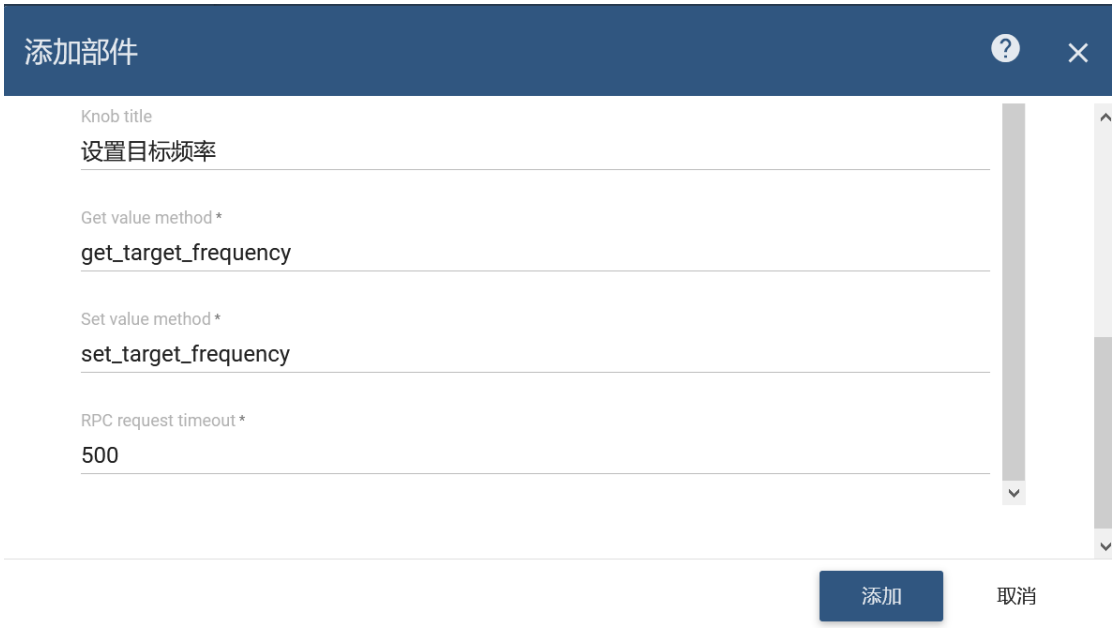
另外，在网关的【系统日志】里面也可以看到 ThingsBoard 下发的信息，并打开继电器的记录，方便调试查问题。



12.4.4.2 添加频率设置按钮


标题 *
my dash
Switch c
选择部件
当前包
Control widgets
系统
控件部件
Target device is not set!
50.00
min
max
Knob control

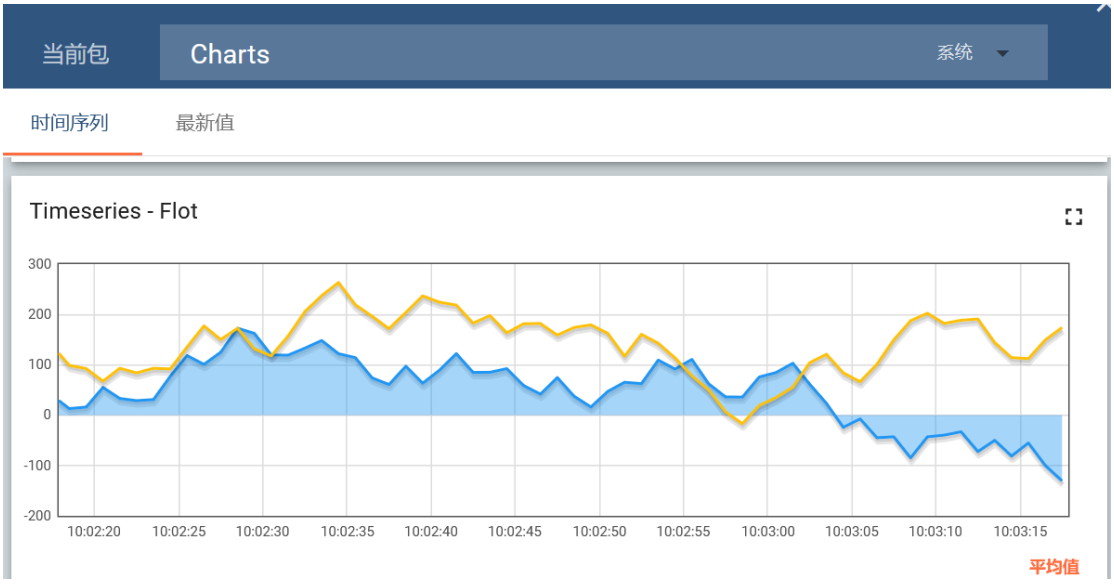

添加部件
数据
设置
高级
动作
目标设备
device1
添加
取消

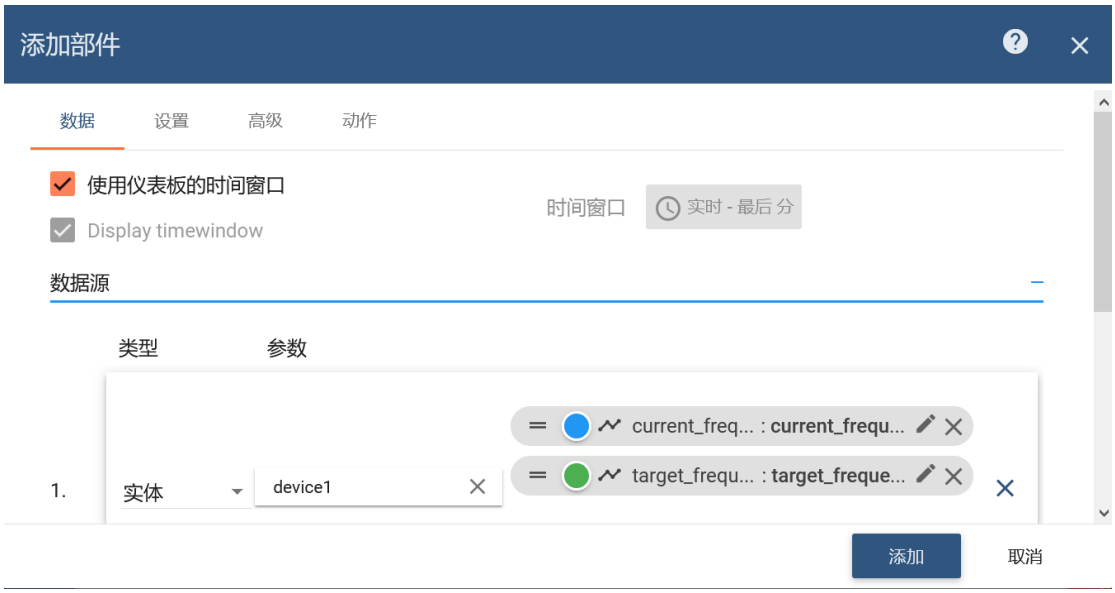


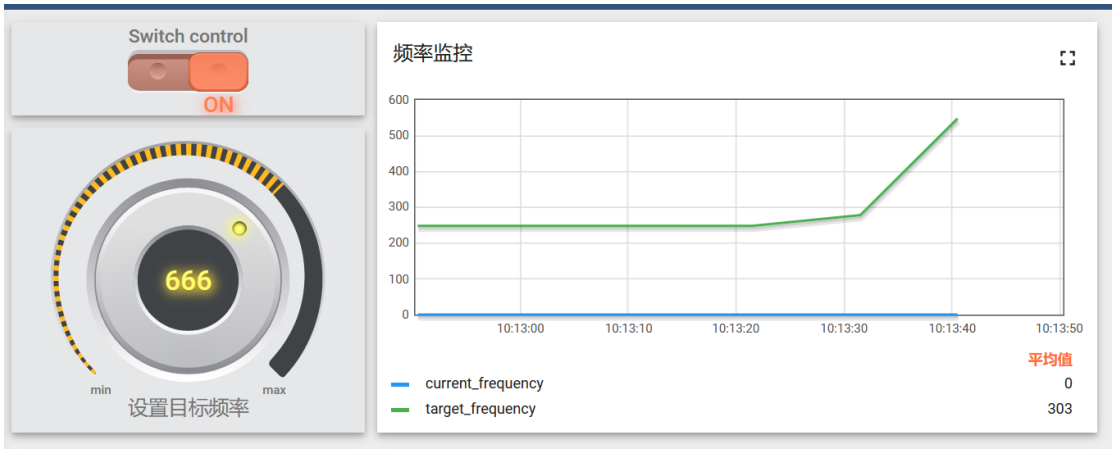
设置浮点位数，若是整数就是 0

添加完就可以看到下图的频率设置旋钮，可随意设置频率

12.4.4.3 添加当前频率和设置频率的曲线







12.4.4.4 手机查看和控制

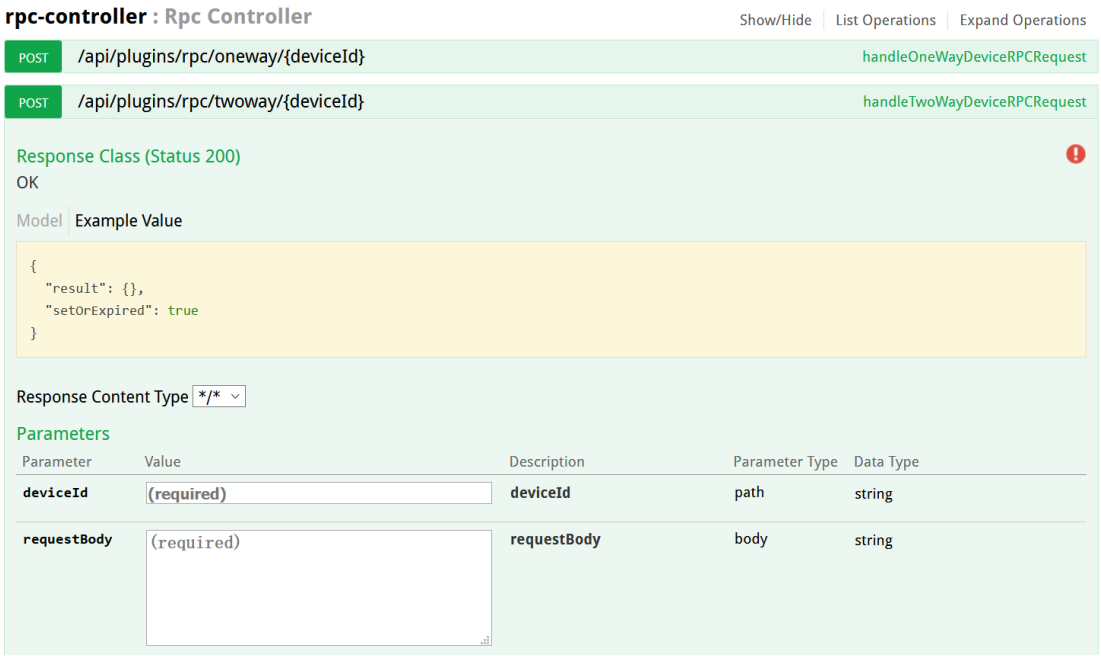
tenant 账户（可以编辑仪表盘）可以创建 custom 账户（只有查看仪表盘权限），然后通过手机浏览器就可以远程查看控制仪表盘。


10:00
192.168.124.202:808
14
my dashboard
my dashboard
Switch control
OFF
248
min
设置目标频率
Powered by Thingsboard v.2.5.4

12.4.5 RESTFUL API

ThingsBoard 提供丰富的 Restful 接口，方便客户做二次开发，通过如下链接，就可以查看具体的 api 含义 http://YOUR_HOST:PORT/swagger-ui.html，例如

http://192.168.124.202:8080/swagger-ui.html



详细资料，请查看官网资料。

https://thingsboard.io/docs/api/

12.4.5.1 通过调试工具 CURL 实现远程控制

将下面的文本复制到一个文本文件，并以 bat 结尾，例如 get_token.bat，其中替代里面的用户名和密码，修改 IP 地址

```
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" -d "{\"username\":\"user@email\", \"password\":\"12345678\"}" http://192.168.124.202:8080/api/auth/login
```

运行后收到如下回复，

{"token":"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ3dWp1bkBicmltZXNoLmNvbSIsInNjb3BlcyI6WyJURU5BTlRfQURNSU4iXSwidXNlcklkIjoiYTNiYzhmNDAtMTk2Zi0xMWViLWE2NWItNTFlMDk5YTA4ZmNhIiwiZW5hYmxlZCI6dHJ1ZSwiaXNQdWJsaWMiOmZhbHNlLCJ0ZW5hbnRJZCI6IjZjYWFjZmQwLTE5NmYtMTFlYi1hNjViLTUxZTA5OWEwOGZjYSIsImN1c3RvbWVySWQiOiIxMzgxNDAwMC0xZGQyLTExYjItODA4MC04MDgwODA4MDgwODAiLCJpc3MiOiJ0aGluZ3Nib2FyZC5pbyIsImlhdCI6MTYwNTQ3OTU0NywiZXhwIjoxNjA1NDg4NTQ3fQ.XHW8yFkoodCTrY4Lh-A-K7GZhflOOFB2L_mHqXzaEh4qBbXYv2iPvqlgmZrA2sg1BY8_u5p-T98ht_AUvCWx9w","refreshToken":"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ3dWp1bkBicmltZXNoLmNvbSIsInNjb3BlcyI6WyJSRUZSRVNIX1RPS0VOIl0sInVzZXJJZCI6ImEzYmM4ZjQwLTE5NmYtMTFlYi1hNjViLTUxZTA5OWEwOGZjYSIsImlzUHVibGljIjpmYWxzZSwiaXNzIjoidGhpbmdzYm9hcmQuaW8iLCJqdGkiOiI0MzE0ZjM4NS1hYjU2LTQ4YzQtYTA0ZS04YzVhNjFlMDlmMGEiLCJpYXQiOjE2MDU0Nzk1NDcsImV4cCI6MTYwNjA4NDM0N30.uKepaiBBegohZLvbRI99tG2sQrCo7o7-fUO455PmnxhTE2y1jbN0ISLMjO38GSb7NOgSzFG4qzmVvRsGolgYDg"}

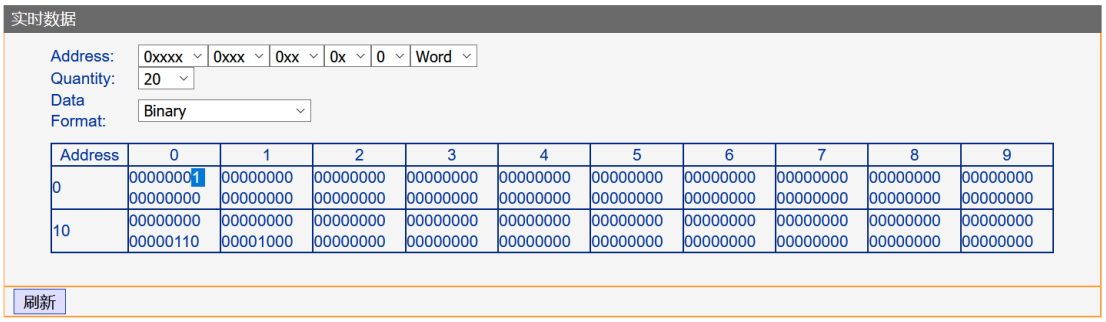
把上面下划线部分替代掉下图框中的 token，下图中的粗体部分需要在 thingsboard 网页中【设备】页面选择对应的控制的设备，并选择【复制设备 id】，并替代掉下框中粗体部分，并以文件 twoway.bat 保存，并运行

```
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" --header "X-Authorization:BearereyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ3dWp1bkBicmltZXNoLmNvbSIsInNjb3BlcyI6WyJURU5BTlRfQURNSU4iXSwidXNlcklkIjoiYTNiYzhmNDAtMTk2Zi0xMWViLWE2NWItNTFlMDk5YTA4ZmNhIiwiZW5hYmxlZCI6dHJ1ZSwiaXNQdWJsaWMiOmZhbHNlLCJ0ZW5hbnRJZCI6IjZjYWFjZmQwLTE5NmYtMTFlYi1hNjViLTUxZTA5OWEwOGZjYSIsImN1c3RvbWVySWQiOiIxMzgxNDAwMC0xZGQyLTExYjItODA4MC04MDgwODA4MDgwODAiLCJpc3MiOiJ0aGluZ3Nib2FyZC5pbyIsImlhdCI6MTYwNTQ3OTU0NywiZXhwIjoxNjA1NDg4NTQ3fQ.XHW8yFkoodCTrY4Lh-A-K7GZhflOOFB2L_mHqXzaEh4qBbXYv2iPvqlgmZrA2sg1BY8_u5p-T98ht_AUvCWx9w" -d "{\"method\":\"rpc\",\"params\":{\"relay_1\":1}}" "http://192.168.124.202:8080/api/plugins/rpc/twoway/c9eae1d0-196f-11eb-a65b-51e099a08fca"
```

运行结果如下：

```
C:\software\thingsboard_tools>curl -X POST --header "Content-Type: application/j
son" --header "Accept: application/json" --header "X-Authorization:Bearer eyJhbG
ciOiJIUzUxMiJ9.eyJzdWIiOiJ3dWp1bkBicmltZXNoLmNvbSIsInNjb3BlcyI6WyJURU5BTlRfQURNS
U4iXSwidXN1ck1kIjoiYTNiYzhmNDAtMTk2Zi0xMWViLWE2NWItNTFlMDk5YTA4ZmNhIiwiZW5hYmx1Z
CI6dHJ1ZSwiaXNQdWJsaWMiOmZhbHNlLCJ0ZW5hbnRJZCI6IjZjYWFjZmQwLTE5NmYtMTFlYi1hNjViL
TUxZTA5OWEwOGZjYSIsImN1c3RvbWVySWQiOiIxMzgxNDAwMC0xZGQyLTExYjItODA4MC04MDgwODA4M
DgwODAiLCJpc3MiOiJ0aG1uZ3Nib2FyZC5pbyIsImlhdCI6MTYwNTQ3OTU0NywiZXhwIjoxNjA1NDg4N
TQ3fQ.XHW8yFkoodCTrY4Lh-A-K7GZhf1OOFB2L_mHqXzaEh4qBbXYv2iPvq1gmZrA2sg1BY8_u5p-T9
8ht_AUvCWx9w" -d "{\"method\":\"rpc\",\"params\":{\"relay_1\":1}}" "http://192.1
68.124.202:8080/api/plugins/rpc/twoway/c9eae1d0-196f-11eb-a65b-51e099a08fca"
{"method":"rpc","params":{"relay_1":1}}
C:\software\thingsboard_tools>
```

这是可以看到继电器被打开，同时在网关的实时数据里面也可以看到对应的 Bit 被设置成 1，如下图：

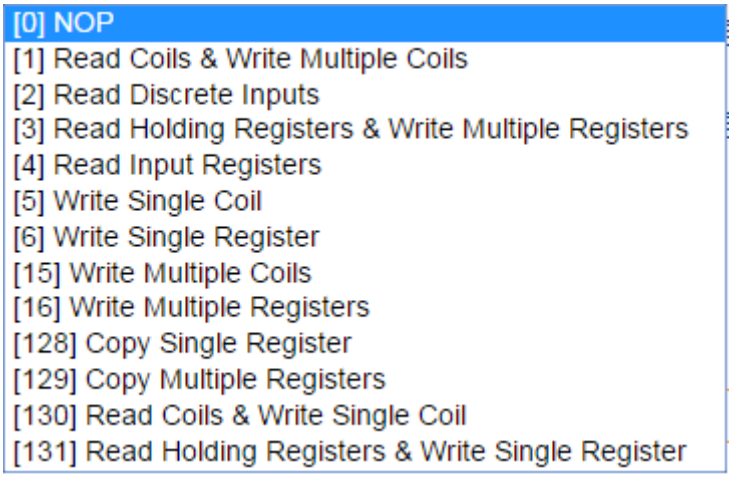


13 常见问题

13.1 如何判断从站数据是否已经采集上来

通讯成功后，可以在【实时数据】页面显示被采集上来的数据，如果有通讯失败的，在【故障数据】页面里显示出故障的记录，并且可在【系统日志】页面查看具体的原因。

13.2 上位机如何通过网关把数据写入从站



在上图这些功能码中，5，6，15，16 是以设定的扫描周期把源虚拟地址中的数据写入目标从站，这里，源虚拟地址中的数据来源于某个从站，也可以来源于上位机主站。网关相当于不停地把虚拟地址中的数据搬运从站地址空间。

而功能码 1，3，130，131 是以设定的扫描周期间隔把从站数据读取到虚拟地址空间。当上位机往虚拟地址写数据的时候，功能码 1 就会通过写多线圈的命令 15 把改动的线圈写入到从站,功能码 3 通过写多个寄存器命令 16 把改动的数据写入从站,功能码 130 通过写单个线圈的命令 5 把改动的线圈写入到从站，功能码 131 通过写单个寄存器的命令 6 把改动的单个寄存器写入到从站。这些写从站的动作只有在收到上位机写数据请求时才会触发，区别于 5，6，15，16 功能码（不停地循环写），功能码 1，3，130，131 在往从站写完数据后，下个周期会继续读数据。

13.3 如何对从站只写不读

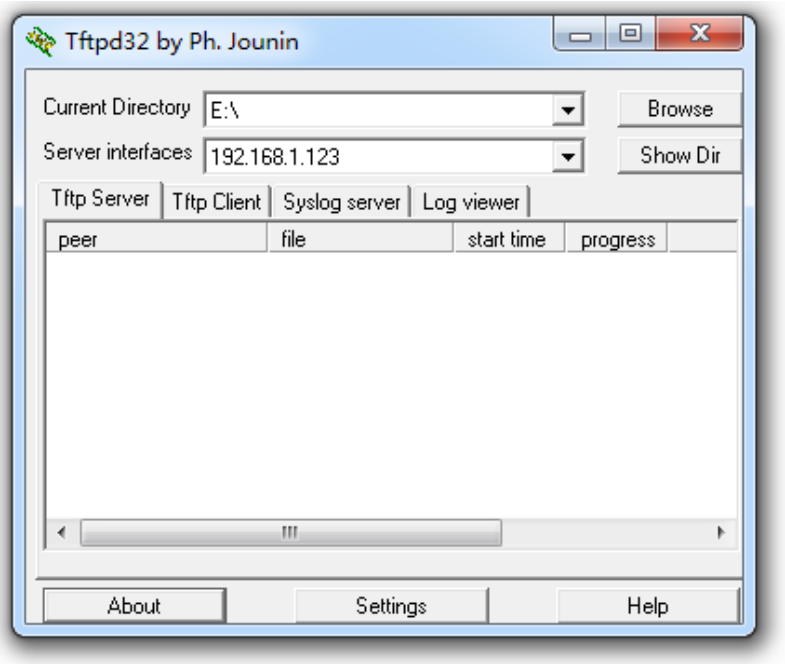
在功能码 1，2，3，4，130，131 中选择满足要求的选项，并且把扫描周期设置成 0，这样网关就不会去读，但在收到上位机写数据请求的时候，网关会把数据写入到从站。

13.4 如何实现 2 个从站间的数据搬运（通讯）

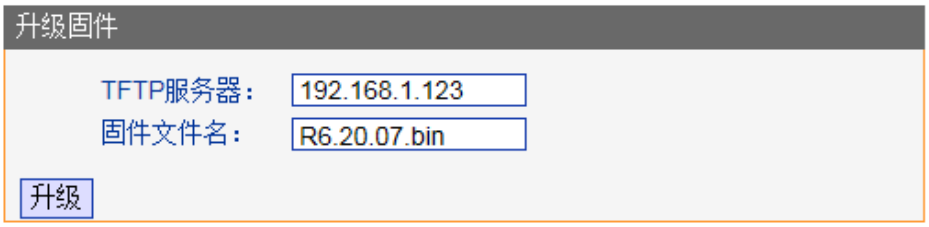
首先，在网关设置读命令，把从站数据读取到指定虚拟地址，然后设置写命令，网关会把虚拟地址内的数据写入设定的从站。注意，读命令要在前，写命令在后。

13.5 如何升级固件

通过网络升级固件的功能极大地提高了设备在现场的可维护性。在升级之前，请先下载要升级的新的固件，并保存到某个目录，假设名字是 R6.20.07.bin，保存于 E 盘。同时请下载安装 tftp Server。推荐使用 tftpd32 软件，非常易用和方便。详细下载地址是：http://tftpd32.jounin.net/，（在我们官网上链接 http://www.brimesh.com/download/TFTPD.rar）打开显示如下界面，在”Current Directory”选择固件所保存固件的文件夹，这里选择 E 盘。这样 TFTP 服务器就安装好了。



之后打开浏览器访问网关，在【系统设置】菜单内可看到【升级固件】对话框，输入 tftp 服务器 IP 地址（也就是计算机的 IP，Windows 系统通过 ipconfig 命令查看）和固件全称（注意，这里一定要全称，包括文件后缀.bin），如下图：



点击【升级】。整个升级过程约 20 秒左右，然后手动刷新网页。升级成功后可以通过系统信息的页面查看新的版本号。

开始升级的提示：

- Start upgrading firmware, please wait 20s and manually refresh the page.

看到网页显示这条消息后，网关会开始升级并且灯会闪烁，整个过程大约 20 秒，然后请手动刷新网页。

升级失败的提示：

- ERROR: Firmware R6.20.07.bin is not found at tftp server!!!

错误原因：固件没有找到，请确认固件是否放置到 tftp 服务软件的当前目录(Current Directory)下，并且名称填写正确。

- ERROR: TFTP service at 192.168.1.123 is unavailable!!!

错误原因：无法访问服务器，或者 tftp 端口（UDP 端口 69）没有打开。确认 IP 是可访问的，并且 tftp 服务端口没有被防火墙屏蔽掉。