2021

WTGNet-NCU

CNC 智能采集网关

无锡望天观科技有限公司

目录

1 前言

1.1 网关介绍

工业数据采集网关，用于采集数控机床的生产数据，如报警信息，生产件数，电机温度，刀具号等关键信息。也可以实现机床的远程管理和程序下载。适用的领域有航天工业，车船制造业，工程机械，电子配件等工业领域。所采集到的数据可以灵活的发送到工业互联网平台进行深度分析，如产量分析，告警分析，排班计划，机床维护，刀具磨损预测等工作，进一步实现企业的工业信息化程度，提升产能，降本增效。

本网关专业采集各种主流 CNC，PLC，机器人，目前支持 Fanuc(发那科)，Siemens(西门子)CNC，Mitsubishi(三菱)CNC，KND(凯恩蒂)，GSK(广数)CNC，Heidenhain(海德汉)，Citizen(西铁城)，Brother(兄弟)，Siemens(西门子)S7，Mitsubishi(三菱)MC，Omron(欧姆龙)Fins，Syntec(新代)，ModbusTCP/RTU 等协议。

本网关程序使用 C++开发，支持 ARM，X86 等各种 CPU 架构的 Linux 系统，可构建至中央服务器统一采集厂房所有联网设备，也可以构建于嵌入式主机，一对一采集单个设备，网关对外提供多种数据输出方式，包括 MQTT，HTTP，Socket，ModbusServer，OPCUA 等通讯协议，支持对接各大云平台，包括 Thingsboard，联通 Telit，移动 OneNet，百度天工，航天云网，紫光云，翰云等。

本网关内置 VPN 服务，搭配 VPN 服务，可以提供高效安全的 VPN 服务，可以实现远程上下载程序锁机等功能。

本文档基于本公司配套硬件设施，其他硬件仅网络配置界面不同。

2 硬件介绍

本网关提供两个有线网口，可选型号支持 WiFi 和 2G/3G/4G，同时提供一个 485 接口，电源为宽压电路，支持 9-30V 输入电压。

图 1：网关外观图

2.1 基本参数

表 1：基本参数

名称	标配参数	描述
尺寸	110.00mm*90.00mm*35.00mm	长* 宽* 高
CPU	AM3352BZCZD60@600MHz	ARM Cortex-A8 架构，工业级温宽
内存	256MB DDR3	工业级温宽：-40℃ +85℃
NandFlash	256MB SLC NandFlash	工业级温宽：-40℃ +85℃

2.2 有线网络

网关的两个网口，分别为 NET0 与 NET1 默认地址为：

表 2：有线网络

网卡	地址	备注
NET0	192.168.0.15/24	
NET0	192.168.253.254/24	不可修改， 永久有效
NET1	192.168.0.15/24	
NET1	192.168.254.254/24	不可修改， 永久有效

2.3 无线网络

无线网络仅支持 2. 4G 网络，不支持隐藏 SSID，无线网络需要登陆之后在页面配置，使用 WiFi 网络需要检查天线是否正常。

2.4 4G 网络

网关的两个网口，分别为 NET0 与 NET1 默认地址为：

表 3：4G 网络

G 灯高低电平状态	模块工作状态
慢闪（200ms 高/1800ms 低）	找网状态
慢闪（1800ms 高/200ms 低）	待机状态
快闪（125ms 高/125ms 低）	数据传输模式

2.5 RS485

485 芯片为 MAX485，最多支持接入 32 个 485 网络设备。

3 管理界面

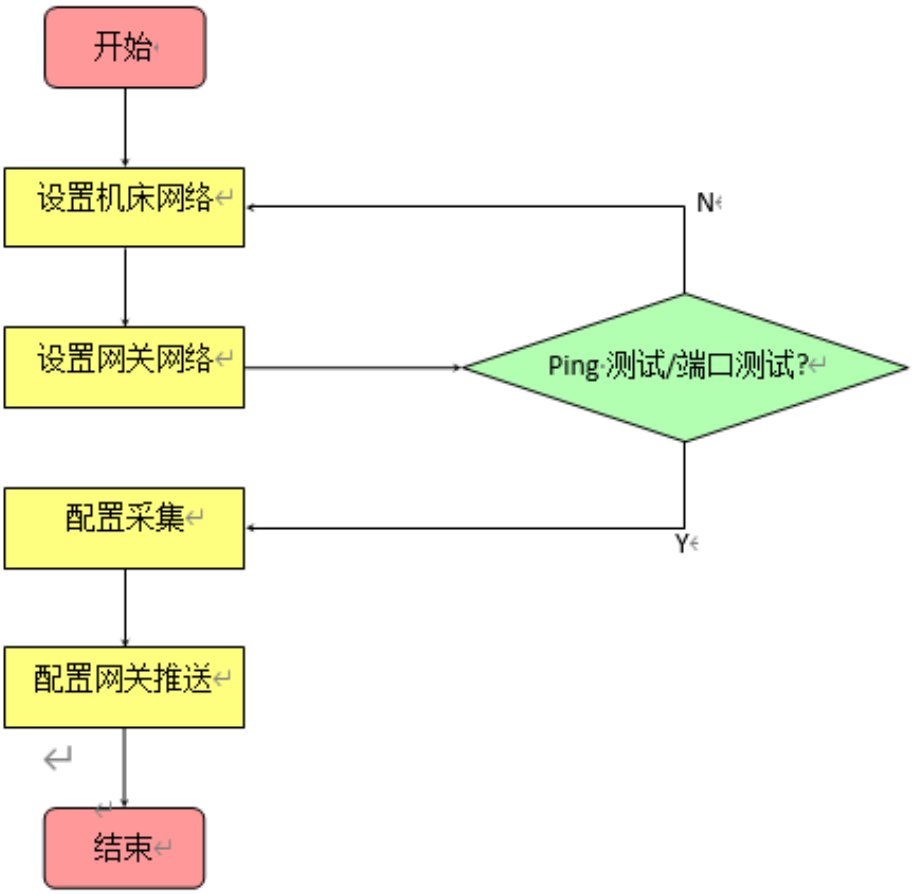
3.1 网关介绍

首次使用管理界面需要先确认电脑与网关的网络硬件连接正常，依据连接的网络接口不同和电脑设置的 IP，可选择以下地址登陆：

- NET0： http：//192.168.0.15：9080
- NET1： http：//10.10.80.15：9080
- NET0： http：//192.168.253.254：9080
- NET1： http：//192.168.254.254：9080

此外，默认情况下 VPN 是连接的，如果插入 4G 卡使得网关拥有外网，也可以通过 VPN 管理后台的地址登陆。

3.2 网关流程



3.3 登陆

用户名为 admin，默认密码为 password，建议登陆后修改密码。

图 2：登录

3.4 系统信息

本页面可以显示网关的基本负载信息，包括 CPU，内存，磁盘，网络等，可以通过右上角的网关信息推送开关，选择是否将网关信息推送到云平台。

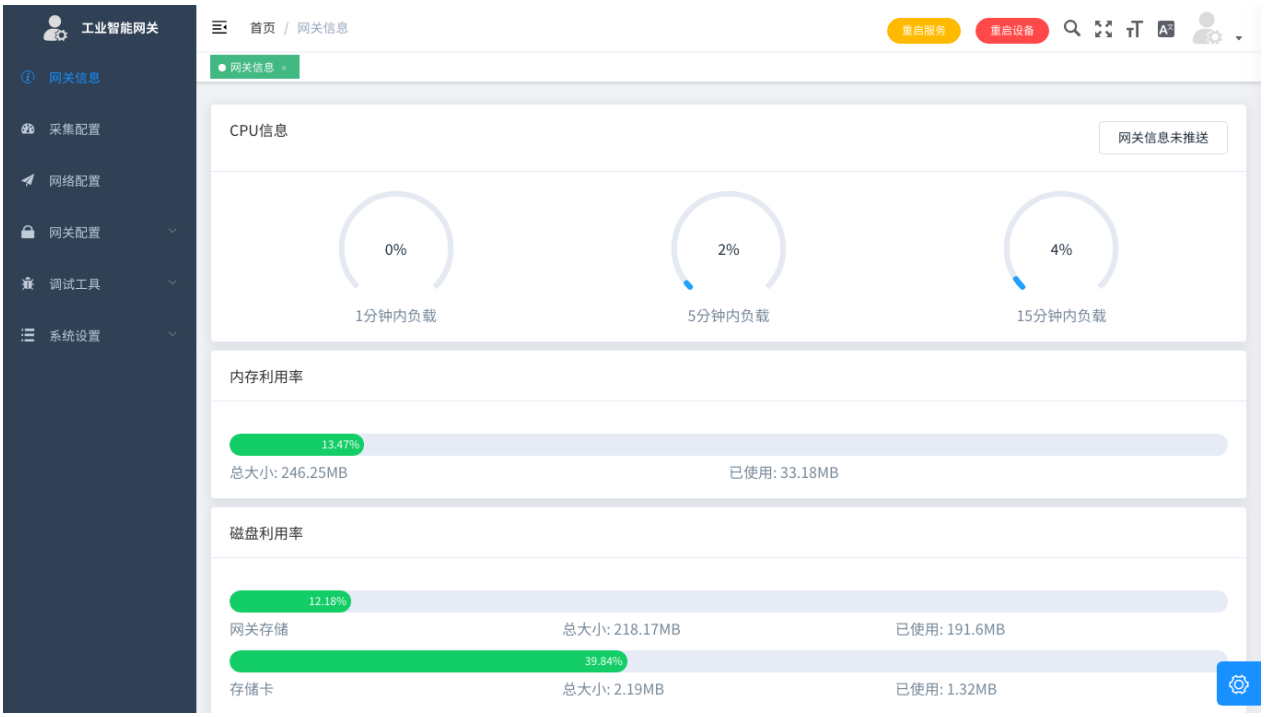


图 3：系统信息

3.5 网络配置

如图，根据不同硬件选配，网卡数量不同。

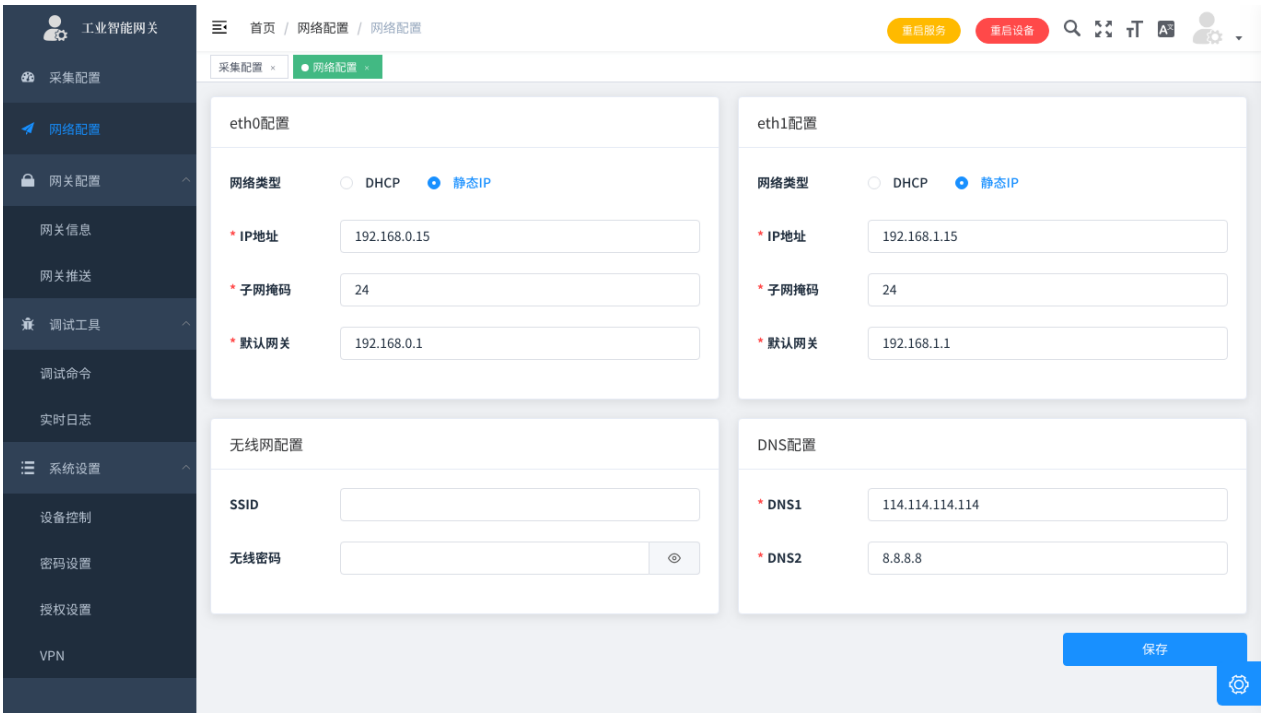


图 4：网络配置

- 有线网卡可以配置 DHCP(动态获取 IP 地址)或静态 IP 地址；
- 子网掩码表示方法为掩码长度，0-32 位的数字，可以用子网掩码计算器计算；
- 无线网络输入 SSID 和密码即可，如果需要禁用无线网络，清空输入即可；
- 所有网络修改完必须重启设备，否则不会生效；
- 所有网卡的网络配置必须保证 IP 地址不在同一网段，包括 DHCP 获取的 IP 和手动配置的静态 IP；
- eth0 为 net0，eth1 为 net1；
- 4G 网络插卡断电重启即可， 无需配置

3.6 网关配置

3.6.1 网关信息

此处可以配置网关 ID，网关 ID 应该是唯一的，很多协议内部会使用网关 ID 作为推送标识。

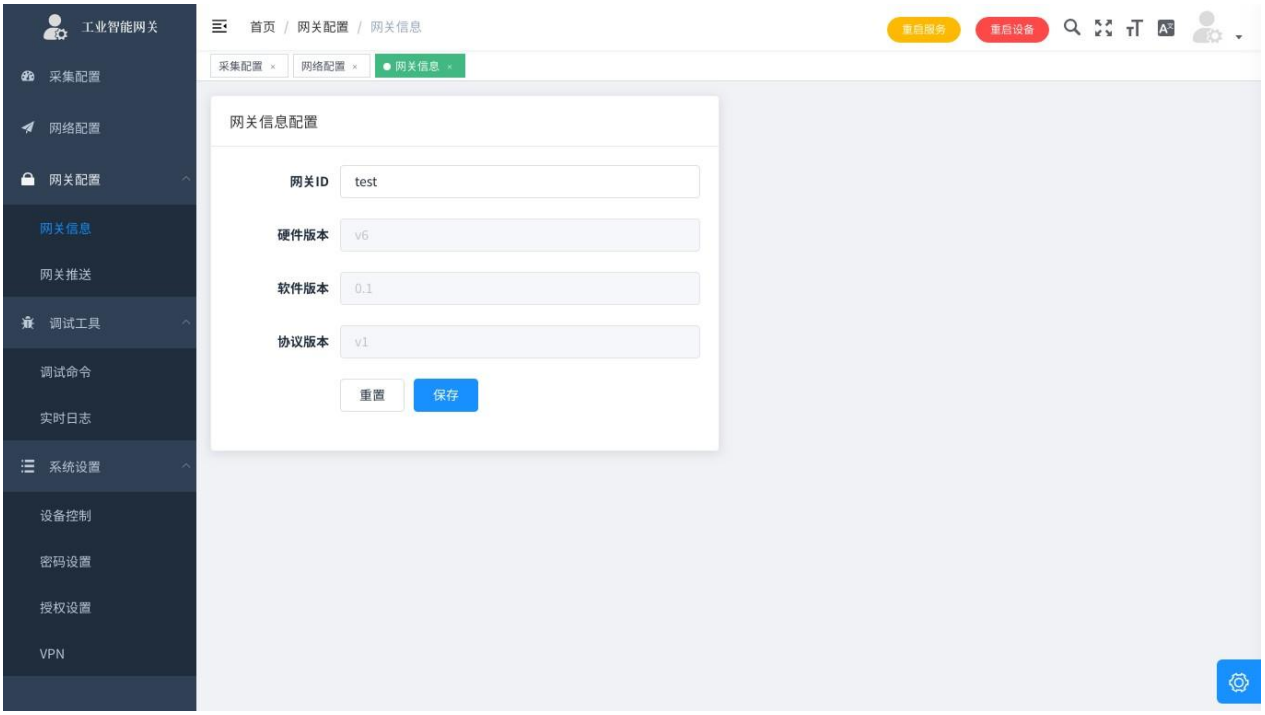


图 5：网关信息

3.6.2 网关推送

目前支持多种平台推送，其他协议如有需求，联系售后，可持续迭代添加，修改推送后，需要重启服务生效。

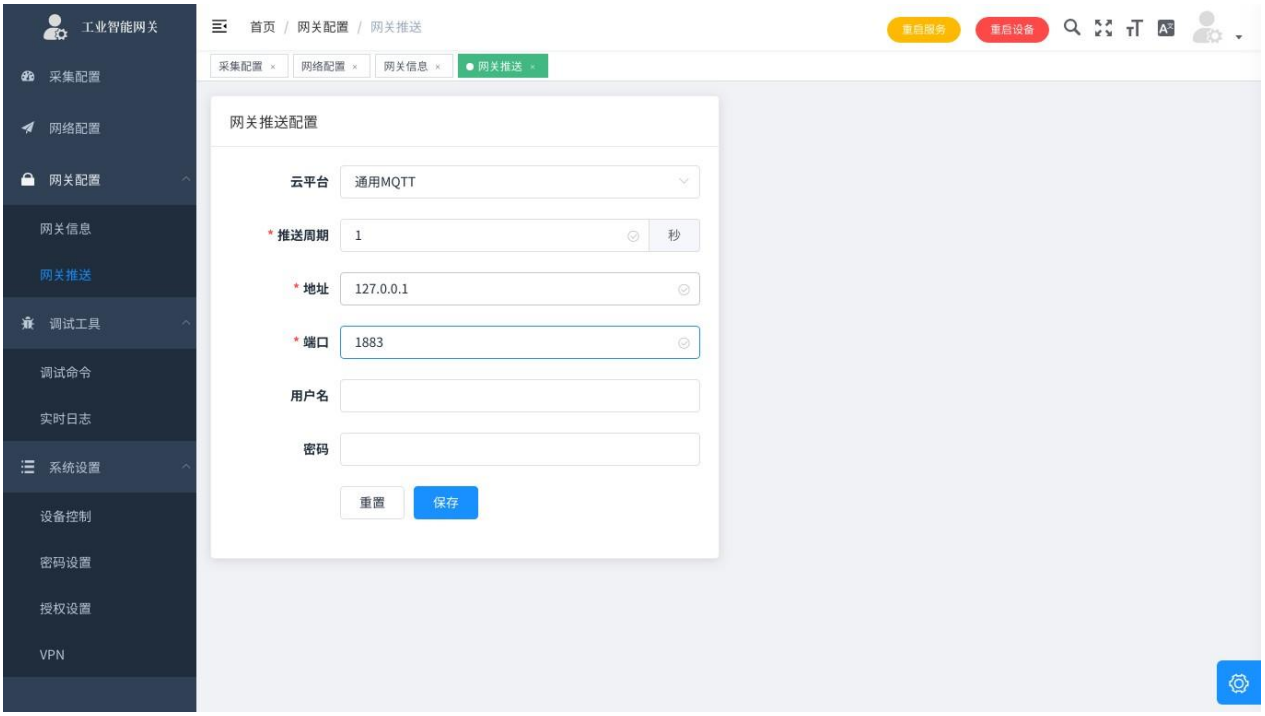


图 6：网关推送

协议文档　网关内置多种平台协议，通用 MQTT 为本公司定义的简易协议，文档参照通用 MQTT 协议其他协议文档参考平台官网。

自定义协议　采用 lua 脚本, 定义如下

```
-- publish.lua function init()
config = {};
config.address = 'tcp://127.0.0.1:1883' config.interval = 30
config.client_id = 'test'
global_config.topic = 'data/' .. global_config.gateway_id .. '/v1' return config
end

function gen_message(message) return message
```

```
end


function push_message(message) log(message)
push(cjson.encode(message), global_config.topic)
end


function init_push(devices) end
```

反向控制	通用 MQTT 支持反向控制, 具体控制方法参照 API 接口

3.7 调试工具

提供一些基本的调试工具，可以用来查看网关状态，以及判断失败原因等。

3.7.1 调试命令

获取 **IP** 地址 获取 IP 地址可以看到本机所以 IP 地址，其中 eth0 为 net0br0/eth0 为 net1wlan0 为 WiFi，eth2 为 4G，tap0 为 VPN。

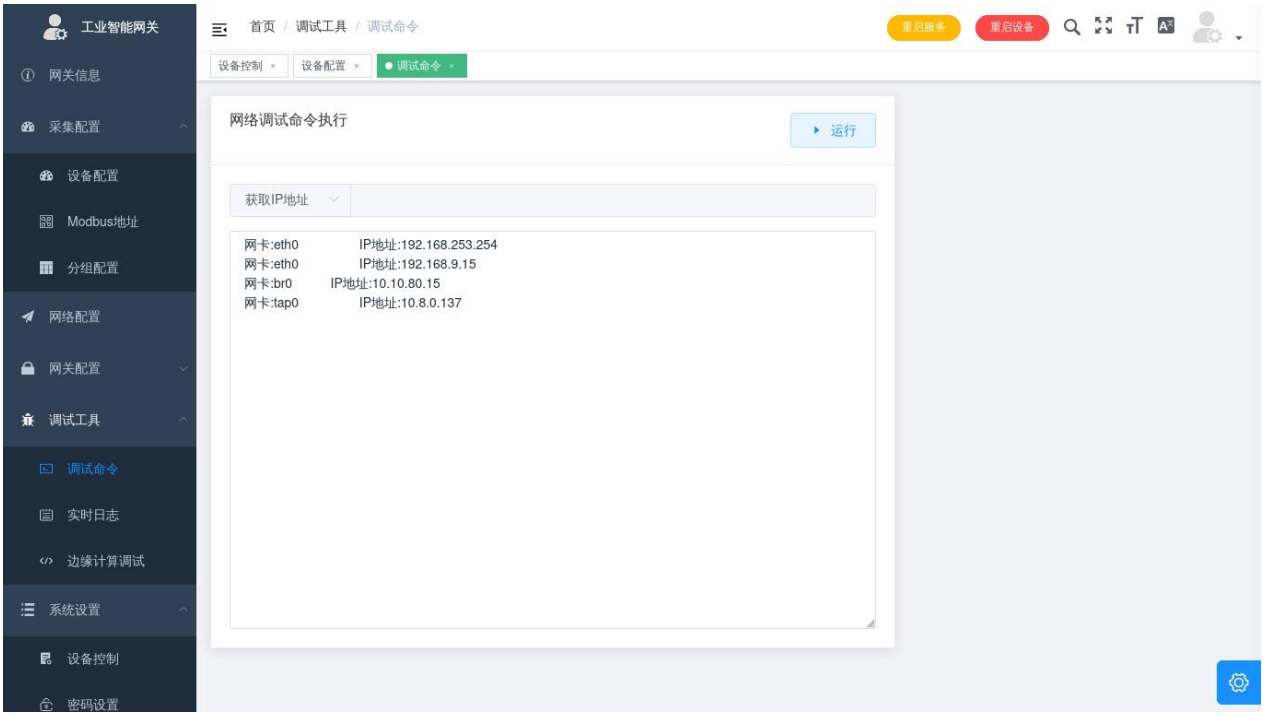


图 7：获取 IP 地址

Ping Ping 可以是 IP 或者域名，可以选择网口。

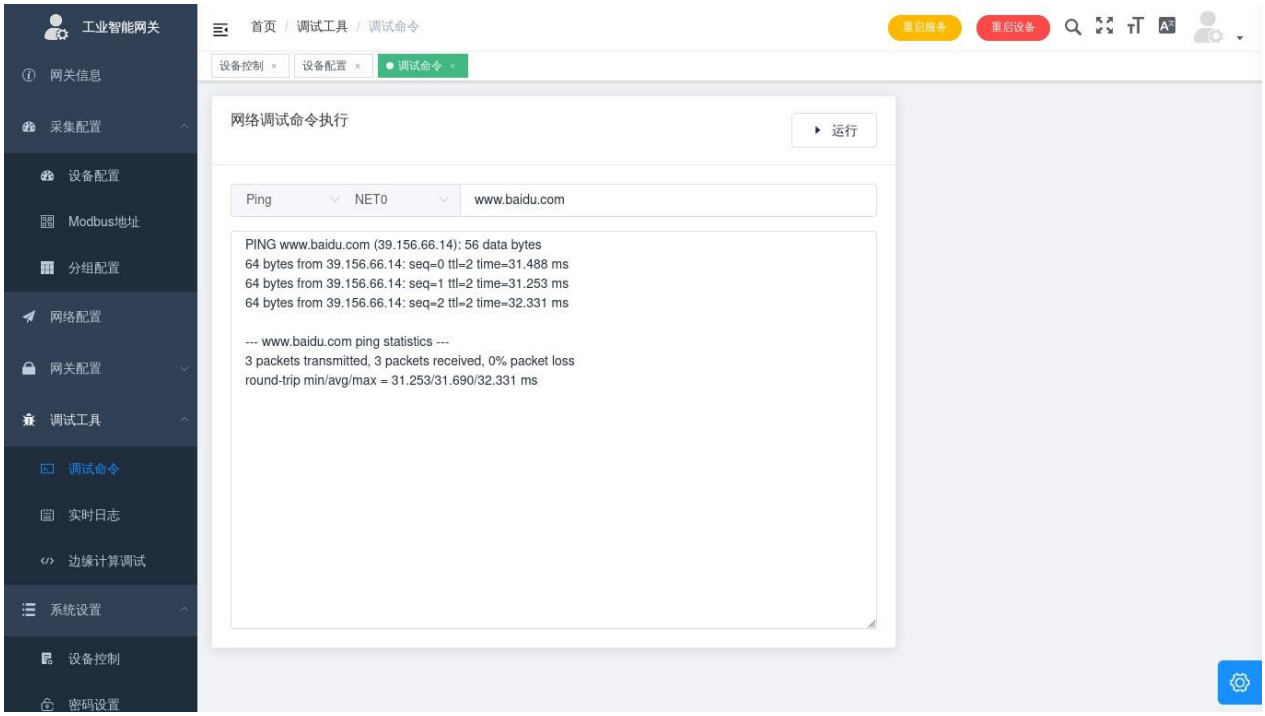


图 8：Ping

路由　路由为 linux 专业工具，需了解专业网络知识，可以查看当前网卡优先级别和路由状态。

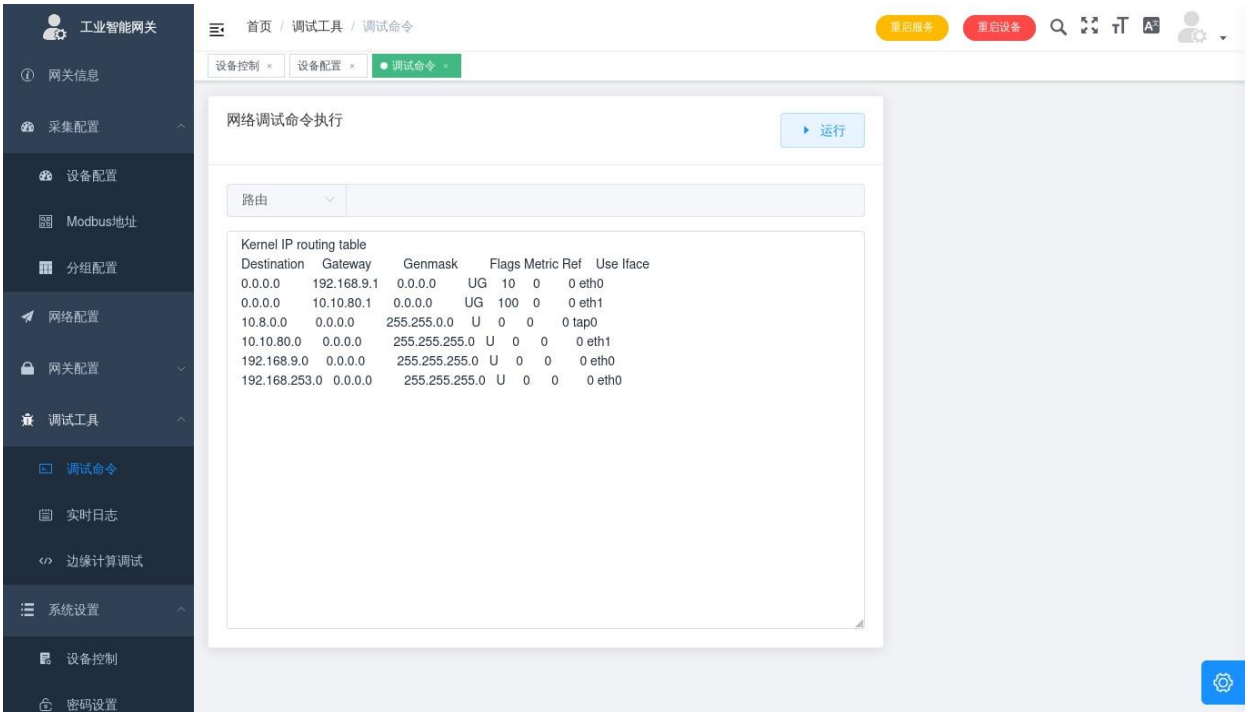


图 9：路由

端口测试　端口测试用来确定 TCP 端口是否开放，格式为 192.168.0.15:22 或 www.Baidu.com:22。

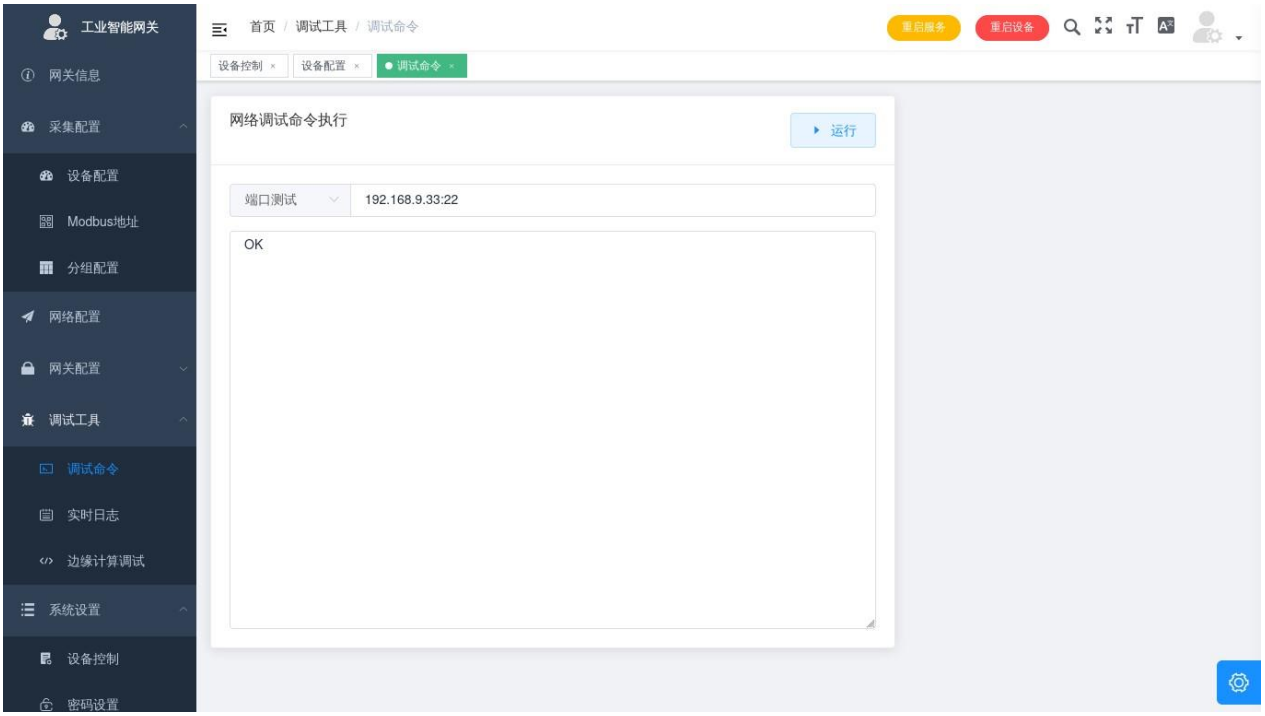


图 10：端口测试

3.7.2 实时日志

实时打印系统日志，注意，这个页面会无限获取日志，如果电脑性能不高，不要长时间待在这个页面。

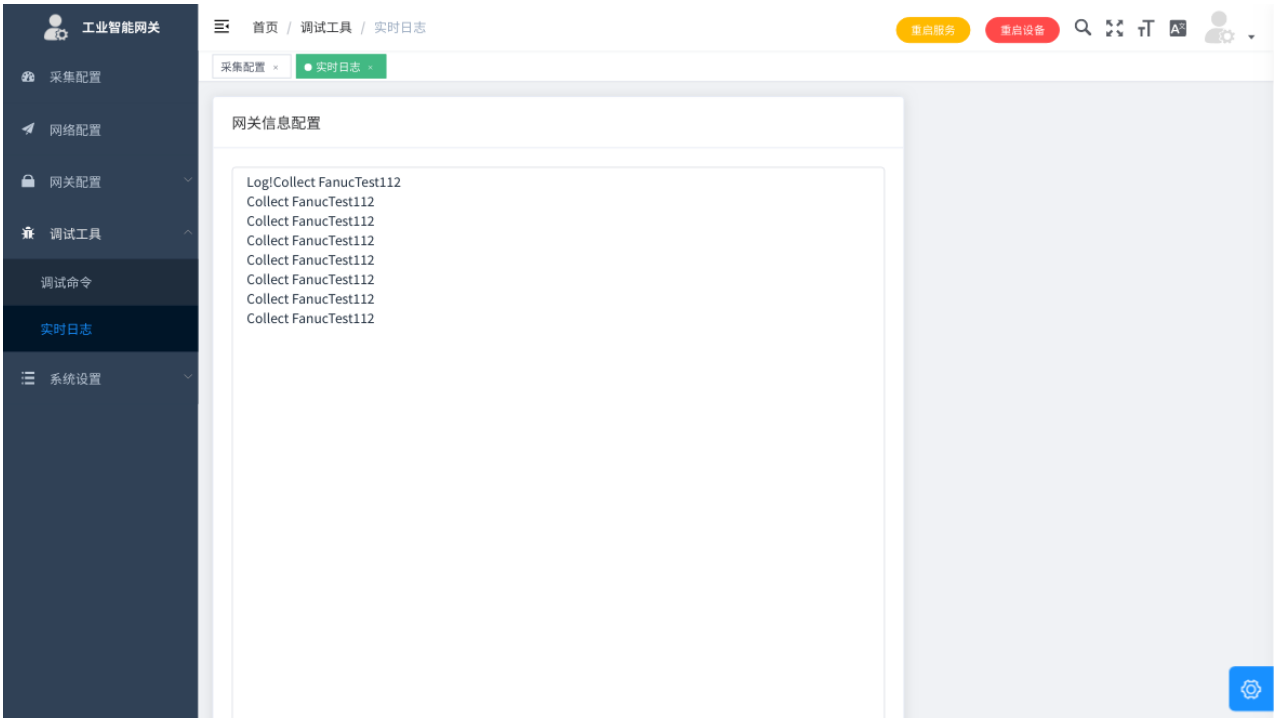


图 11：实时日志

3.8 系统设置

提供通用的系统设置功能。

3.8.1 设备控制

起停一些网关扩展功能包括：

- 路由器，模式将网关作为一个路由器，可以通过网关的网络上网；
- Modbus 服务将网关的数据以 ModbusTcp 输出；
- OPC UA 服务将网关数据以 OPC UA 方式输出；

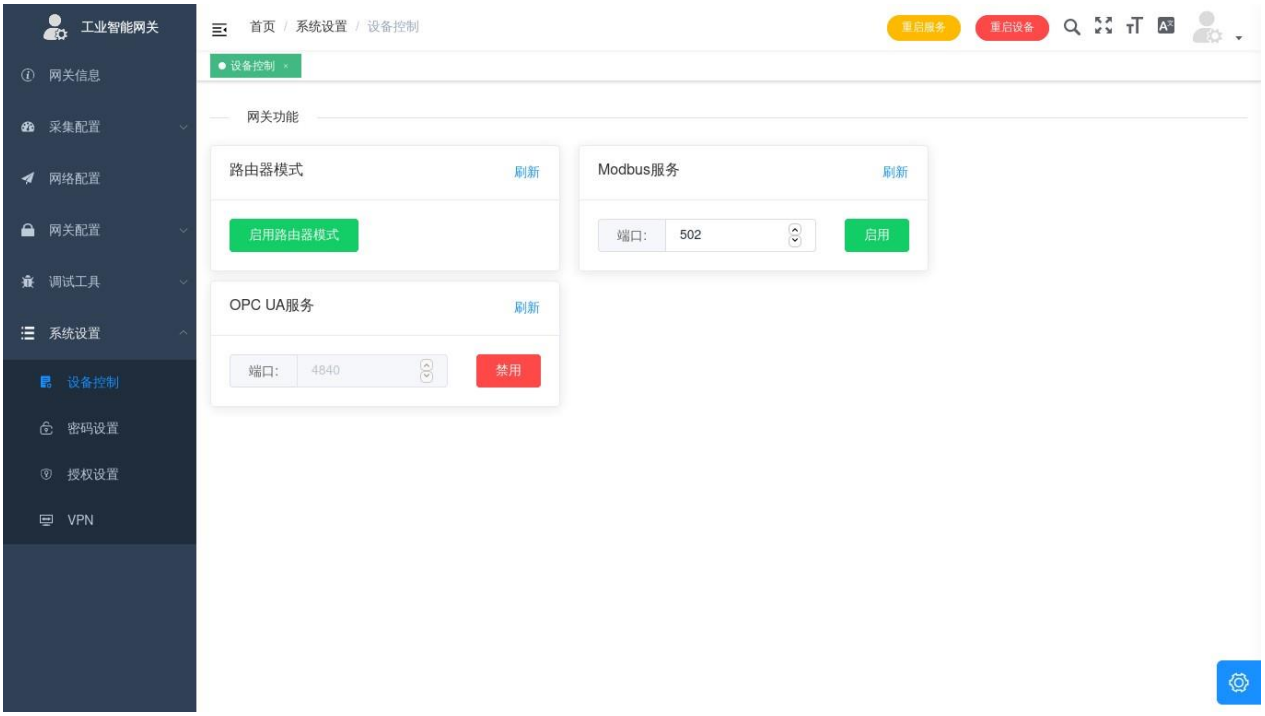


图 12：设备控制

3.8.2 密码设置

建议设置高强度密码，不要透漏给其他人。

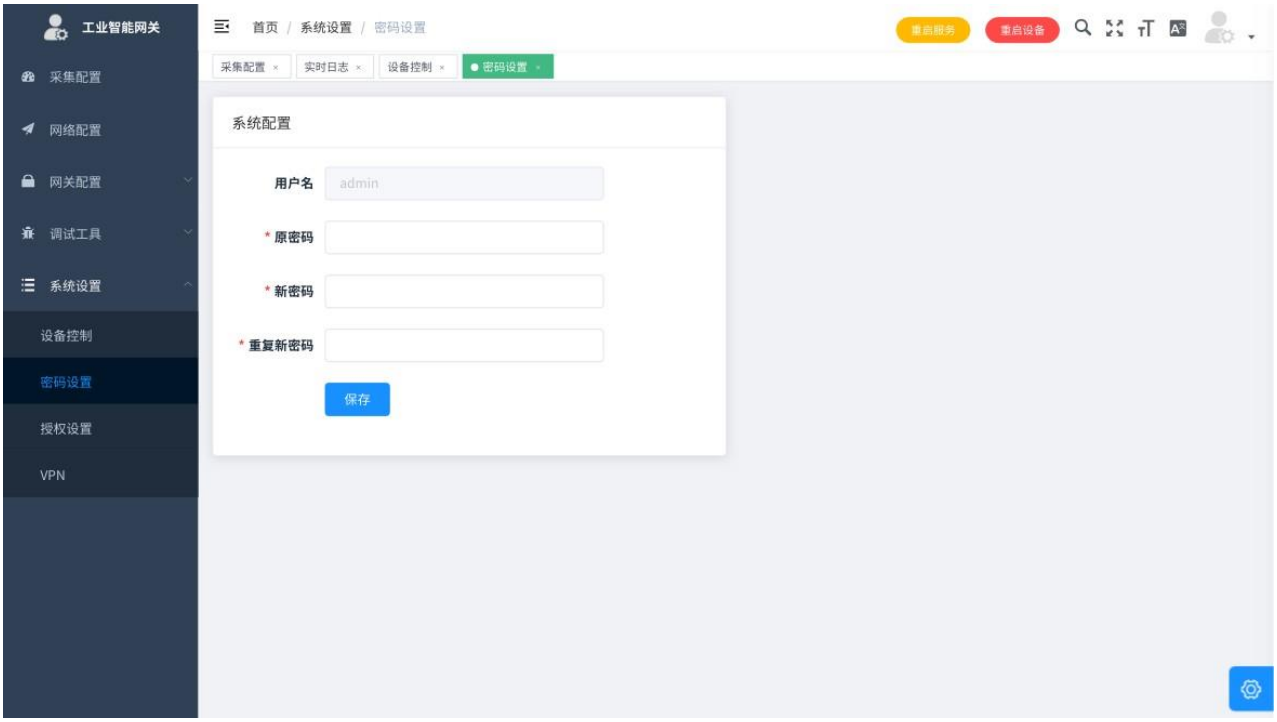


图 13：密码设置

3.8.3 授权设置

- 设备指纹为设备唯一编码，设备授权基于设备指纹，如需更新设备授权，请联系售后；
- 可采集数量当 all>0 时，为可以采集任意型号，总数不大于 all；
- 当 all=0 时，为可采集以下任意设备，且采集的型号的总数不大于对应授权的数量；

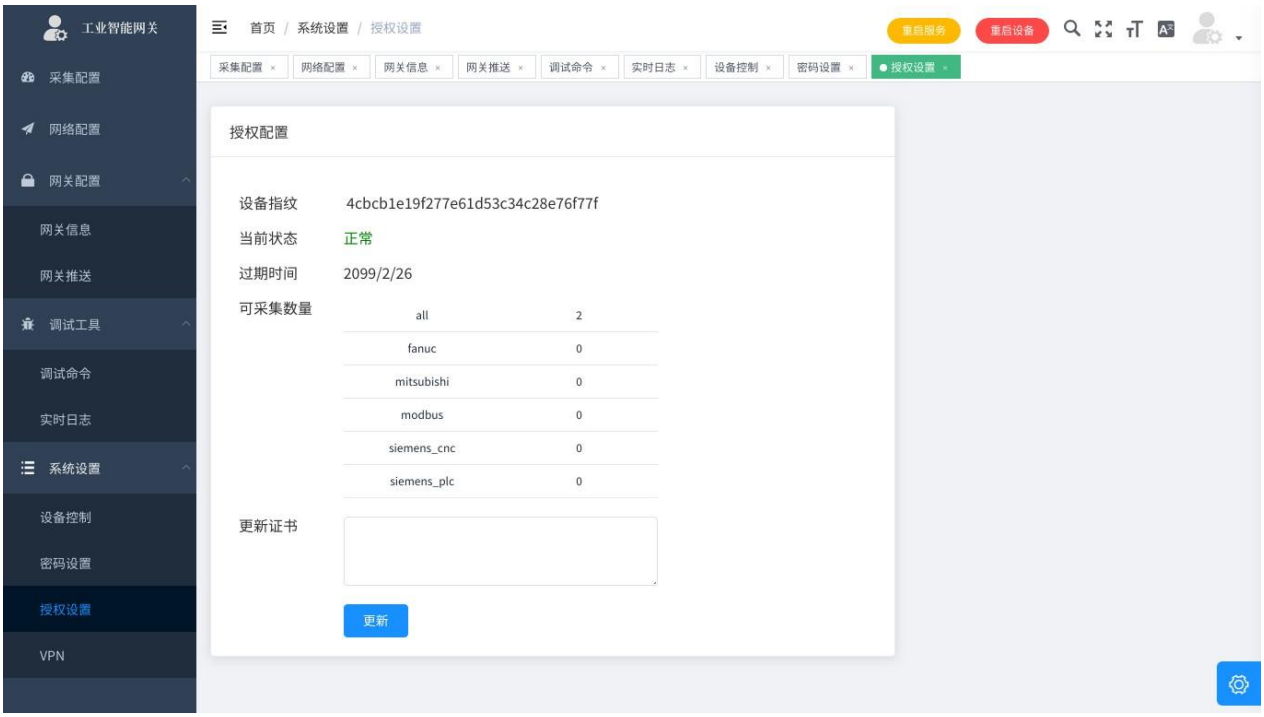


图 14：授权设置

3.8.4 VPN 设置

参见 VPN 章节。

4 通用 MQTT

4.1 推送格式

通用 MQTT 为网关默认推送协议，推送主题为 data/{gateway_id}/v1，gateway_id 为网关信息里的网关 ID，推送格式如下

```
[
  {
    "device_id": "Test", "ts":
    1561994655032,
    "device_type": "fanuc",
    "values": [
      {
        "name": "cnc_products",
        "value": 20559
      },
      {
        "name": "cnc_type",
        "value": "FANUC 0i MF"
      },
      {
        "name": "device_state",
        "value": 0
      },
      {
        "name": "cnc_mecpos",
        "value": [
          {
            "axis": "X",
            "value": -84.4
          },
          {
            "axis": "Y",
            "value": 2.986
          },
          {
            "axis": "Z",
            "value": -108.181
          }
        ]
      },
```

```
    {
      "name": "cnc_alarm",
      "value": [
        {
          "alarm_no": 1,
          "alarm_type": "T01",
          "alarm_msg": "待机中"
        },
        {
          "alarm_no": 204,
          "alarm_type": "XXX",
          "alarm_msg": "急停"
        }
      ]
    }
  ]
}
]
```

4.2 Java Demo

Java 程序的解析 Demo

```
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken; import
org.fusesource.mqtt.client.*;

import java.lang.reflect.Type; import
java.util.ArrayList;
import java.util.concurrent.TimeUnit;

class Alarm {
    public String alarm_type; public
    String alarm_msg; public int
    alarm_no;
}

class Position { public String
    axis;
    public Double value;
}

class Value {
    public String name; public
    Object value;
}

class Root {
    public String device_id; public
    long ts;
    public String device_type; public
    Value[] values;
}

public class JavaDemo {
    final static String TOPIC_NAME = "data/+/v1"; final static
    String IP = "127.0.0.1";
    final static int PORT = 1883;

    public static void run() throws Exception { MQTT mqtt =
        new MQTT();
```

```
mqtt.setHost(IP, PORT);
BlockingConnection connection = mqtt.blockingConnection();
connection.connect();
System.out.println("Connected to Broker!");
Topic[] topics = {new Topic(TOPIC_NAME, QoS.EXACTLY_ONCE)};
connection.subscribe(topics);
Type AlarmType = new TypeToken<ArrayList<Alarm>>() {
}.getType();
Type PositionType = new TypeToken<ArrayList<Position>>() {
}.getType(); while
(true) {
    Message message = connection.receive(10, TimeUnit.SECONDS); if
    (message != null) {
        Type RooList = new TypeToken<ArrayList<Root>>() {
        }.getType();
        ArrayList<Root> table = new Gson().fromJson(new
            String(message.getPayload()), RooList);
        for (Root tt : table)
            { System.out.println(tt.device_id); for
            (Value value : tt.values) {
                System.out.print(value.name + "\t");
                if (value.value instanceof java.util.ArrayList) { if
                    (value.name.equals("cnc_alarm")) {
                        ArrayList<Alarm> alarms = new
                            Gson().fromJson(value.value.toString(), AlarmType);
                        for (Alarm alarm : alarms) { System.out.print("\nalarm_no:" +
                            alarm.alarm_no +
                                "\talarm_type:" + alarm.alarm_type +
                                "\talarm_msg:" + alarm.alarm_msg);
                        }
                    } else {
                        ArrayList<Position> position = new Gson().fromJson(value.value.toString(), PositionType);
                        for (Position axis : position) { System.out.print(axis.axis + ":" + axis.value +
                            "\t");
                        }
                    }
                } else {
                    System.out.print(value.value);
                }
                System.out.println();
```

```
                    }
                }
                message.ack();
            }
        }
    }

    public static void main(String[] args) throws Exception { JavaDemo.run();
    }
}
```

4.3 Net core Demo

```
using System;
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Client.Options; using
System.Text;
using System.Collections.Generic;
namespace dotnet_demo
{
    public class Position
    {
        public string axis { set; get; } public string
        value { set; get; }
    }

    public class Alarm
    {
        public int alarm_no { set; get; } public string
        alarm_type { set; get; } public string alarm_msg
        { set; get; }
    }

    public class Values
    {
        public string name { get; set; } public
        object value { get; set; }
    }

    public class Root
    {
        public string device_id { get; set; } public long ts
        { get; set; }
        public string device_type { get; set; } public
        List<Values> values { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
```

```
string topicSubscribe = "data/+/v1";
MqttClient mqttClient = new MqttFactory().CreateMqttClient() as MqttClient;
mqttClient.UseConnectedHandler(async handle =>
    {
        var result = await mqttClient.SubscribeAsync(new MqttTopicFilter()
        {
            Topic = topicSubscribe,
            QualityOfServiceLevel =
                MQTTnet.Protocol.MqttQualityOfServiceLevel.AtLeastOnce
        });
     });

mqttClient.UseApplicationMessageReceivedHandler(handle =>
    {
        var payload = Encoding.Default.GetString(handle.ApplicationMessage.Payload);
        List<Root> table = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Root>>(payload);
        foreach (var m in table)
        {
            foreach (var s in m.values)
            {
                Console.Write(s.name + "\t");
                if (typeof(Newtonsoft.Json.Linq.JArray) == s.value.GetType())
                {
                    if (s.name == "cnc_alarm")
                    {
                        List<Alarm> alarms =
                            Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alarm>>(s.va foreach (var
                        alarm in alarms)
                        {
                            Console.WriteLine("\nalarm_no:" + alarm.alarm_no.ToString()
                                + "\talarm_type:" + alarm.alarm_type + "\talarm_msg:" +
                                alarm.alarm_msg);
                        }
                    }
                    else
                    {
                        List<Position> pos =
                            Newtonsoft.Json.JsonConvert.DeserializeObject<List<Position>>(s foreach (var
                        axis in pos)
```

```
                                {
                                    Console.Write(axis.axis + ":" +
                                        axis.value.ToString() + "\t");
                                }
                            }
                        }
                        else
                        {
                            Console.Write(s.value);
                        }
                        Console.WriteLine();
                    }
                }
            });
        var options = new MqttClientOptionsBuilder()
            .WithProtocolVersion(MQTTnet.Formatter.MqttProtocolVersion.V311)
            .WithClientId(Guid.NewGuid().ToString().Substring(0, 5))
            .WithTcpServer("127.0.0.1", 1883)
            .WithCleanSession()
            .Build();
        mqttClient.ConnectAsync(options);
        Console.ReadKey();
    }
}
    }
```

4.4 Python Demo

```
import json

import paho.mqtt.client as mqtt
def on_connect(mqtt_client, userdata, flags, rc):
    mqtt_client.subscribe('data/+/v1')
def on_message(mqtt_client, userdata, msg): try:
        table = json.loads(msg.payload) for tt in
        table:
            print(tt.get('device_id'))
            for value in tt.get('values'):
                print(value.get('name'), end='\t') data =
                value.get('value')
                if isinstance(data, list):
                    if value.get('name') == 'cnc_alarm': for alarm
                        in data:
                            print(
                                f'\nalarm_no:{alarm.get("alarm_no")}\talarm_type:{alarm.get("alarm_typ end='')
                    else:
                        for axis in data: print(f'{axis.get("axis")}:{axis.get("value")}', end='\t')
                else:
                    print(data, end='')
                print('')
    except Exception as e:
        print(e)
mqtt_client = mqtt.Client() mqtt_client.on_connect =
on_connect mqtt_client.on_message = on_message
mqtt_client.connect('127.0.0.1', 1883, 60)

if__name____== '_main_':
    mqtt_client.loop_forever()
```

4.5 Golang Demo

```
package main

import (
    "encoding/json"
    "fmt"
    mqtt "github.com/eclipse/paho.mqtt.golang" uuid
    "github.com/satori/go.uuid"
    "time"
)
type Alarm struct {
    AlarmType string `json:"alarm_type,omitempty"`
    AlarmMsg string `json:"alarm_msg,omitempty"` AlarmNo
    int64 `json:"alarm_no,omitempty"`
}
type Position struct {
    Axis string `json:"axis,omitempty"` Value float64
    `json:"value,omitempty"`
}
type Value struct {
    Name string          `json:"name,omitempty"` Value
    interface{} `json:"value,omitempty"`
}
type Root struct {
    DeviceId string `json:"device_id,omitempty"` DeviceType
    string `json:"device_type,omitempty"` Ts     int64
                `json:"ts,omitempty"`
    Values       []Value `json:"values,omitempty"`
}
func reciveHandler(client mqtt.Client, m mqtt.Message) { var messages
    []Root
    _ = json.Unmarshal(m.Payload(), &messages) for _,
    message := range messages {
        fmt.Println(message.DeviceId)
        for _, value := range message.Values
            {fmt.Print(value.Name + "\t") switch
            value.Value.(type) {
            case []interface{}:
                if value.Name == "cnc_alarm" {
                    _alarm, _ := json.Marshal(value.Value)
```

```
                    var alarms []Alarm
                    _ = json.Unmarshal(_alarm, &alarms) for _,
                    alarm := range alarms {
                        fmt.Printf("\nalarm_no: d\talarm_type: s\talarm_msg: s", alarm.AlarmNo,
                            alarm.AlarmType, alarm.AlarmMsg)
                    }
                } else {
                    _position, _ := json.Marshal(value.Value) var position
                    []Position
                    _ = json.Unmarshal(_position, &position) for _,
                    axis := range position {
                        fmt.Printf(" s: f\t", axis.Axis, axis.Value)
                    }
                }
            default:
                fmt.Print(value.Value)
            }
            fmt.Println()
        }
    }
}
func main() {
    topic := "data/+/v1"
    address := "tcp://wuyun.pro:1883" u1,
    _ := uuid.NewV4()
    opts := mqtt.NewClientOptions().AddBroker(address).SetClientID(u1.String()) c :=
    mqtt.NewClient(opts)
    if token := c.Connect(); token.Wait() && token.Error() != nil { panic(token.Error())
    }
    if token := c.Subscribe(topic, 0, reciveHandler); token.Wait() && token.Error()
        != nil { fmt.Println(token.Error())
    }
    for {
        time.Sleep(1 * 1000 * 1000)
    }
}
```

---

5 API

5.1 HTTP

- BASEURL： /api

登陆

请求

- Method： POST
- URL： /user/login
- Headers：Content-Type：application/json
- Body：

```
{
"username": "admin",
"password": "password"
}
```

返回

- 200

```
{
"username": "admin",
"password": "password"
}
```

- 400 请求参数错误
- 410 密码错误

详细 API 列表参考附录

5.2 MQTT

MQTTAPI 是对 HTTP 的一层封装，但是基于 MQTT 的特性，需要将请求与返回分开

请求 topic 为 control/gateway_id/v1gateway_id 为网关 ID 请求报文为：

```
{
  "cmd_id": "uuid",
  "method": "PUT",
  "url": "/api/device/{device_name}/control_by_name", "data": {
    "key": "M100",
    "value": 1,
    "value_type": 1
  }
}
```

其中

- cmd_id 为指令唯一编号， 建议用 uuid， 返回主题的 payload 会与之匹配
- url 参考 httpapi
- method 参考 httpapi
- data 参考 http api 的 request body， 若无请求 body， 请用”data”：null

返回 topic 为 echo/gateway_id/v1gateway_id 为网关 ID 每次发完 control 主题后，都会有 echo 请求报文为：

```
{
  "cmd_id": "uuid",
  "msg": "OK",
  "ret": 0
}
```

其中

- cmd_id 为指令唯一编号，与 control 中的 cmd_id 匹配；
- msg，请求返回值；
- ret 若请求成功，值为 0， 若失败，参考 http api 的 http_statuscode。

6 VPN

6.1 简介

6.1.1 未授权

未授权情况下用户只可选择是否启用 VPN，启用 VPN 后可远程配置本网关，仅可以搭配本公司网关管理平台，关闭后，任何人将不能远程控制网关，对安全性有要求的公司，可以关闭 VPN 功能。

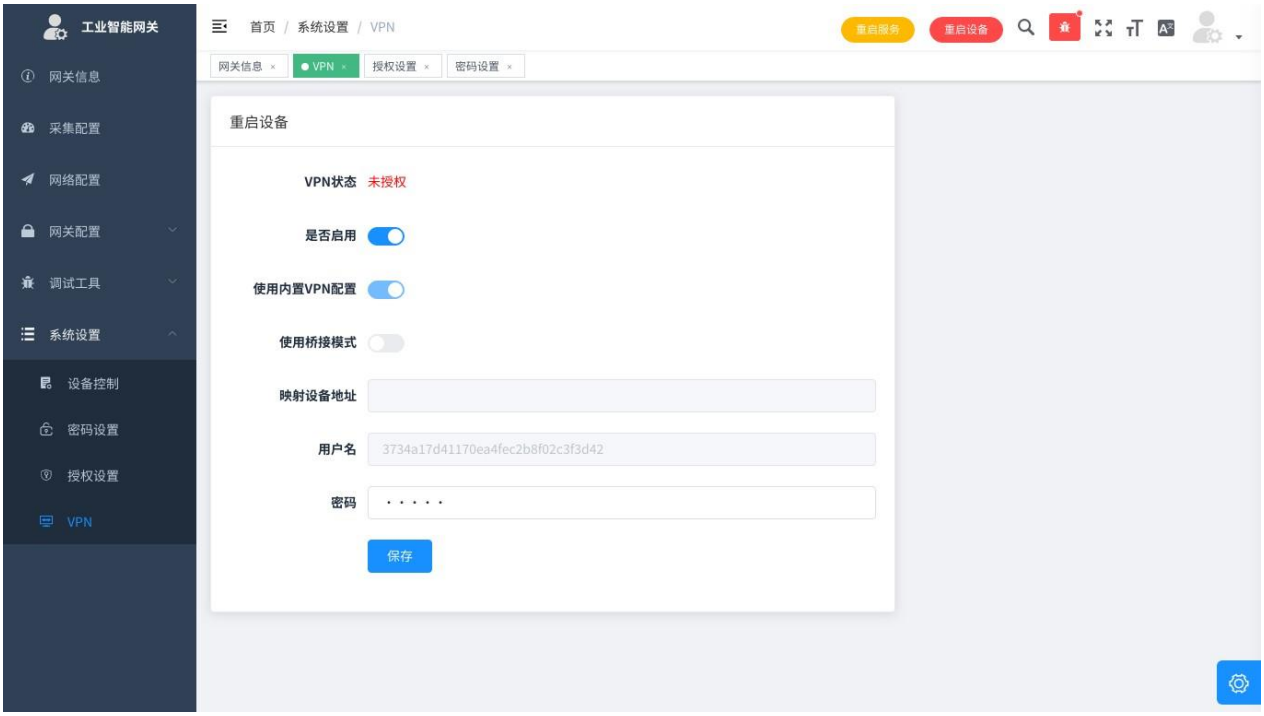


图 15：未授权 VPN

6.1.2 已授权

- VPN 已授权后可以映射采集设备， 使用自定义配置文件，以及使用桥接模式；
- 观云配置为本公司默认配置，可搭配本公司平台使用(推荐)；
- 映射设备：地址为欲通过 VPN 控制的机床设备的 IP 地址，请确保在调试页面可以 ping 通，支持 TCP 和 UDP 点对点协议，不支持 UDP 广播，如需映射设备，需要将设备的默认网关(Default Gateway，Fanuc 中为路由器地址)设置为本网关的 IP 地址；
- 桥接模式：桥接模式启用后，VPN 会转为交换机模式，所有连接在 net1 上的机床设备，与采集网关网关本身，与客户电脑的 VPN 网卡将处于同一虚拟交换机上.本方案适用于 VPN 批量机床设备，或者机床/PLC 的控制协议非 tcp 协议的情况；
- 观云配置的密码为平台配置密码，需要与平台密码一致。

6.2 使用方法

如需使用授权 VPN 功能，首先联系客服获取平台账号，打开智能网关管理系统，首页有配置文件和软件，请根据自己的平台选择使用。

图 16：平台首页

设备列表页面可以看到自己公司所有设备的状态，列表所示的 IP 地址即为采集网关的 VPNIP 地址，可以用这个地址管理网。


望天观CNC采集网关
首页
设备管理
首页 / 设备管理 / 设备管理
首页
设备管理
标题
搜索
添加
导出
序号 设备ID 公司名称 设备名称 设备指纹 IP地址 更新时间 授权许可 是否在线 操作
1 111 望天观科技 833f5bb932d27fb629ecd8a67c616359 833f5bb932d27fb629ecd8a67c616359 10.8.0.120 2019-12-18T18:10:57 已授权 机床 编辑 删除
2 129 望天观科技 3734a17d41170ea4fec2b8f02c3f3d42 3734a17d41170ea4fec2b8f02c3f3d42 10.8.0.142 2019-12-14T14:39:48 已授权 机床 编辑 删除
3 63 望天观科技 雪大师 66c268d9d36d7d4fe148aba6188c9bbf 10.8.0.39 2019-12-07T16:57:42 未授权 机床 编辑 删除
4 1 望天观科技 办公室测试001 3580803b9efafb1ef2cea886c8d2dec 10.8.0.88 2019-06-25T22:43:50 已授权 机床 编辑 删除

图 17：设备列表

6.2.1 映射设备使用方法

1. 采集网关的 VPN 页面将需要映射的设备的 IP 地址填入映射 IP 地址；
2. 需要被映射的设备的网络配置界面需要把 Gateway 设置为采集网关的 IP 地址；

然后就可实现采集网关到设备的映射，此时在连接了 VPN 的电脑上可直接对采集网关的 VPNIP 做机床/PLC 上下载功能，即相当于对机床/PLC 本身的操作。

图 18：映射配置图

6.2.2 桥接模式使用方法

在采集网关的 VPN 页面，将桥接模式置于打开模式，即可使用 VPN 桥接模式，此方案比较复杂，适用于映射方案无法解决问题的时候。

本方案的核心为，将机床/PLC 与用户置于同一交换机，所以需要用户手动配置 IP 地址，使网络可达，网络配置方法如下(如果不需要配置 IP 地址，可以跳过，直接用相关软件进行广播扫描)

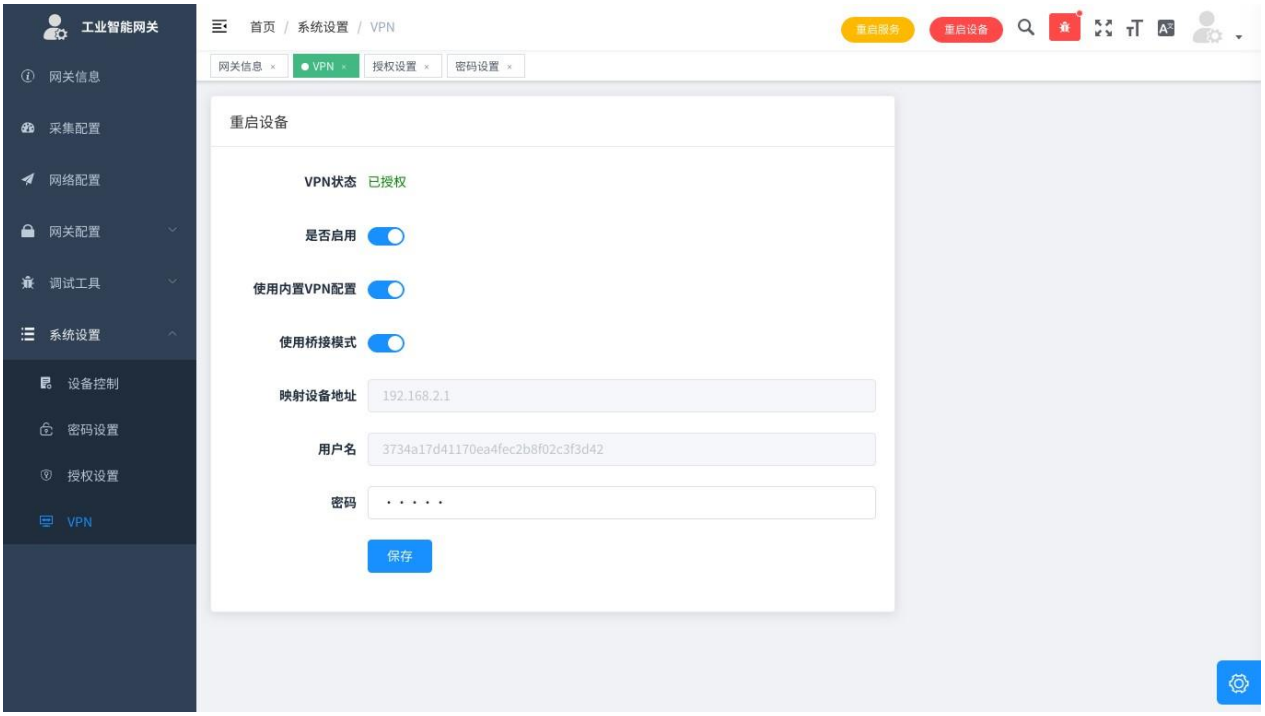


图 19：桥接配置图

- 桥接模式的 IP 地址设置有两种方式：

1) 修改机床 机床修改 IP 地址(推荐)

a) 在设备列表页面， 点击本次配置的网关的操作列，机床按钮，进入本网关的机床列表；

b) 填加机床后，会给机床分配一个机床 IP，这个 IP 在全网内都是不冲突的，可以安全使用；

c) 点击 IP 地址，会出现完整的 IP 配置参数；

d) 在机床的网络界面将本页面展现的网络参数填入，最主要的两个参数为 IP 地址和子网掩码，其他参数可以不设置(如果不需要配置 IP 地址，本条可以跳过)；

e) 至此网络配置完成，如果配置了 IP 地址，可以直接操作机床；

2) 修改本机-如果机床已有自己的网络地址，不方便修改，可以通过修改电脑 IP 达成一个子网的目的

a) 在 windows 网络中心找到 VPN 的网卡， 驱动包含 Tap 字段。

图 20：机床 IP

b) 打开 cmd(也许需要管理员权限) 输入以下命令

```
netsh interface ip add address "以太网卡名" 192.168.x.x 255.255.255.0 #将以太网卡名替换为上一步查到的名称,IP 地址按照自己的规则修改
```

可能的问题：

- 如果有多台网关同时开启了桥接模式，有可能会导致机床 IP 冲突
- 本方案填加的 IP 地址，会在 VPN 重连后丢失，每次重连需要重新配置

6.2.3 映射端口模式

1) 保证网关能 ping 通需要映射的设备，并且可以访问到需要转发的端口；

2) 采集网关的 VPN 页面填加需要映射的 IP 和端口；

3) 如下图所示，内部端口为自动生成，可以通过本网关的 vpn 地址加内部端口访问被映射的服务；

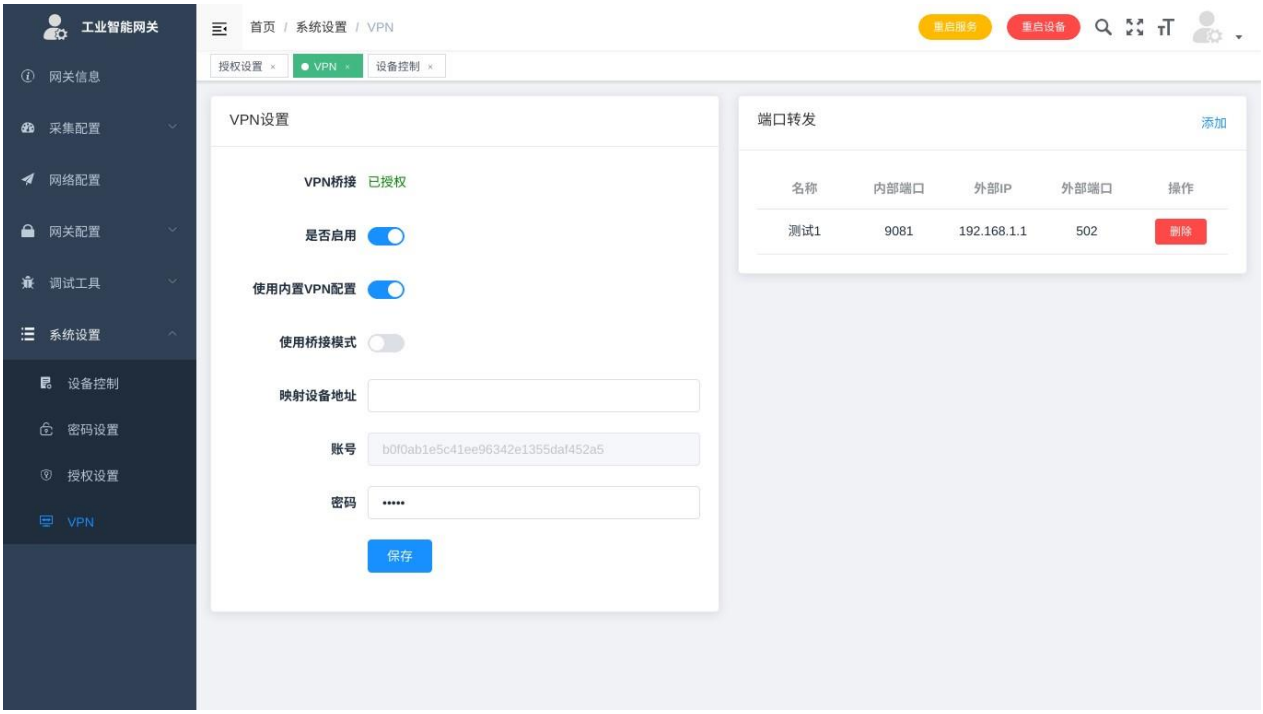


图 21：映射端口模式

6.2.4 路由模式

1) 如下图打开设备控制界面

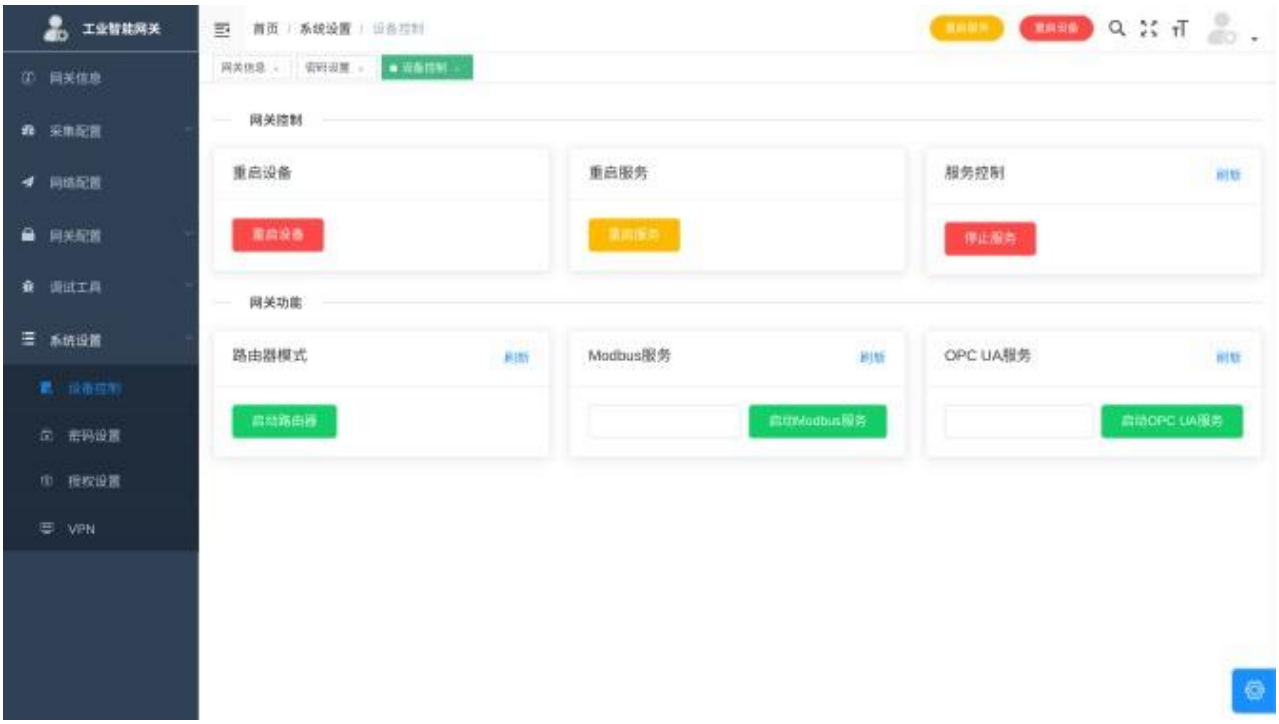


图 22：路由模式

2) 启用路由器模式

3) 保证网关能 ping 通需要访问的设备

4) 打开 Windows 的 cmd 命令框， 需要管理员模式输入以下命令

```
route add 192.168.1.0 mask 255.255.255.0 10.8.0.22

:: 192.168.1.0 为需要路由的网段,也可以为单个 IP

:: 255.255.255.0 为需要路由的网段掩码,单个 IP 用 255.255.255.255

:: 10.8.0.22 为网关的 VPN IP
```

5) 设置完成即可直接访问设备的内部 IP， 如以上配置下的 192.168.1.x。

6.2.5 特殊网络拓扑

联系售后人员指导配置。

6.3 注意事项

1) VPN 可直接打通企业内网，如果对安全性要求很高，需谨慎操作；

2) 如需私有部署网关管理平台，请联系商务；

3) 由于 VPN 服务的数据流量会过服务器，必定会导致延迟过高，对于低延迟场景，避免使用 VPN；

7 边缘计算

本网关提供基础的边缘计算功能，边缘计算暂时支持每个变量的独立计算与单个设备所有变量同时参与计算两种。

关于 Lua 语法，参见 Lua 教程。

7.1 单变量计算

规则 单个变量脚本支持两个输入参数，函数名必须为 calc，参数变量名可以自行决定

```
function calc(current,last)
    --[[
    current:
        本次采集到的值,注意,此处的值有可能为数字,字符串,array,table 等 Lua 数据类型,需要根据不同变量做判
    last:           上次采集到的值,如果为第一次采集则为空
    return:         计算完返回的变量,可以为数字,array,table,字符串等如果
    脚本错误,上报的数据为错误内容
    ]]
    current=current+1
    return current
end
```

测试 如图，当前值输入脚本的第一个参数，上次值输入脚本的第二个参数，然后输入脚本，执行后，输出里即可得到 return 的值；

注意，如果输入值为数字，则直接输入数字，如果输入值为字符串，则输入带引号的字符串，如果输入值为 table，则输入 json 格式；

配置 在采集配置编辑与添加页面，可以针对变量配置脚本，如下图，选择对应的变量，在脚本框输入经过测试的脚本即可。

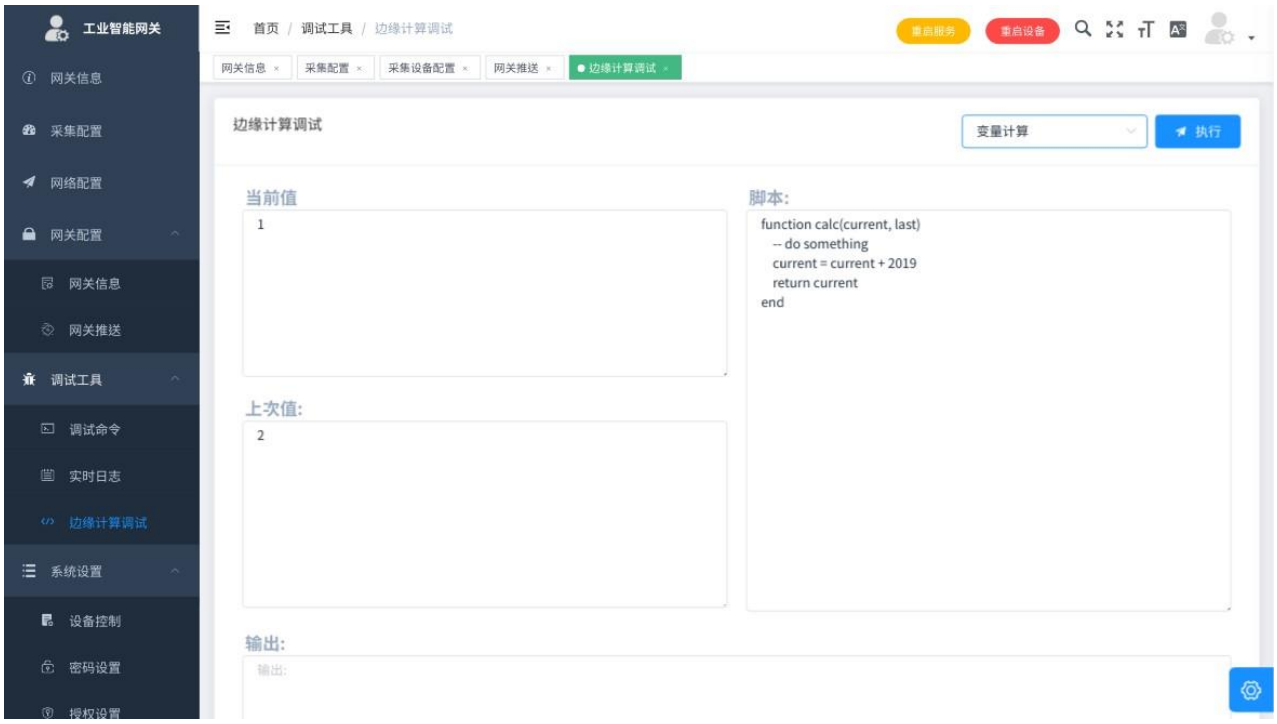


图 23：单变量测试

图 24：单变量配置

7.2 设备计算

规则　设备计算脚本支持一个输入参数，函数名必须为 calc，参数变量名可以自行决定

```
function calc(value)
    --[[
    value:          本次所有采集到的变量,格式为一个 table,可以使用变量名取到值
    return:         计算完返回的变量,建议使用以输入相同的数据结构输出,否则可能会导致推送失败
    如果脚本错误,上报的数据内会增加一个为 script_error 的字段卖,内容为错误内容
    ]]
    value.run_status="正常"
    return value
end
```

测试如图，全局变量输入脚本的参数，然后输入脚本，执行后，输出里即可得到 return 的值注意，输入为严格的 json 格式。

图 25：设备计算测试

配置　在采集配置编辑与添加页面，可以针对变量配置脚本，如下图，选择对应的变量，在脚本框输入经过测试的脚本即可。

图 26：设备计算配置

7.3 内置方法

---

push_message(msg,topic)
--通过现有 mqtt 配置推送消息
--msg: 消 息
--topic:可选,默认为数据 topic

set_key(key,value,persistant)
--写入缓存变量
--key: 变量名
--value: 值
--persistant: 是否持久化,0 存入 redis,1 存入 sqlite

del_key(key,persistant)
--删除缓存变量
--key: 变量名
--persistant: 是否持久化,0 存入 redis,1 存入 sqlite

get_key(key,persistant)
--获取缓存变量
--key: 变量名

--persistant: 是否持久化,0 存入 redis,1 存入 sqlite

--return: 变量的值

get_device_sim()

--获取设备 sim 卡信息

get_device_gps()

--获取设备 GPS 信息

7.4 示例

将　某个变量结果 +1

```
function calc(current,last)
    current=current+1 return
    current
end
```

将　某个变量结果追加一个字符串

```
function calc(current,last)
    current=current .. "woody"
    return current
end
```

联　合两个变量计算出新的变量

```
function calc(value)
    if(value.cnc_runstatus==1 and #value.cnc_alarm>0) then
        value.custom=2
    end
    return value
end
```

删　除某个变量

```
function calc(value)
    value.cnc_runstatus=nil
    return value
end
```

将　Alarm 的 Array[Object] 展开成普通变量

```
{
  "cnc_alarm": [
    {
      "alarm_no": 1, "alarm_type":
      "T01",
      "alarm_msg": "待机中"
```

```
    },
    {
      "alarm_no": 204,
      "alarm_type": "XXX",
      "alarm_msg": "急停"
    }
  ]
}
```

转换成

```
{
  "cnc_alarm_1_msg": "待机中",
  "cnc_alarm_1_no": 1,
  "cnc_alarm_1_type": "T01",
  "cnc_alarm_2_msg": "急停",
  "cnc_alarm_2_no": 204,
  "cnc_alarm_2_type": "XXX"
}
```

代码

```
function calc(value)
    for i, v in ipairs(value.cnc_alarm) do value['cnc_alarm_' .. i .. '_msg']
        = v.alarm_msg value['cnc_alarm_' .. i .. '_no'] = v.alarm_no
        value['cnc_alarm_' .. i .. '_type'] = v.alarm_type
    end
    value.cnc_alarm = nil
    return value
end
```

将 坐标的 Array[Object] 展开成普通变量

```
{
  "cnc_mecpos": [
    {
      "axis": "X",
      "value": -84.4
    },
    {
      "axis": "Y",
```

```
        "value": 2.986
      },
      {
        "axis": "Z",
        "value": -108.181
      }
    ]
}
```

转换成

```
{
  "cnc_mecpos_X":    -84.4,
  "cnc_mecpos_Y":    2.986,
  "cnc_mecpos_Z":    -108.181
}
```

代码

```
function calc(value)
    for i, v in ipairs(value.cnc_mecpos) do
        value['cnc_mecpos_'..v.axis]=v.value
    end
    value.cnc_mecpos=nil
    return value
end
```

8 Modbus 输出

网关可以设置通过 Modbus 输出采集到的数据。

如下图，打开网关控制界面，在 Modbus 服务配置里输入端口(不填默认 502)并启用，即可开启 Modbus 功能。

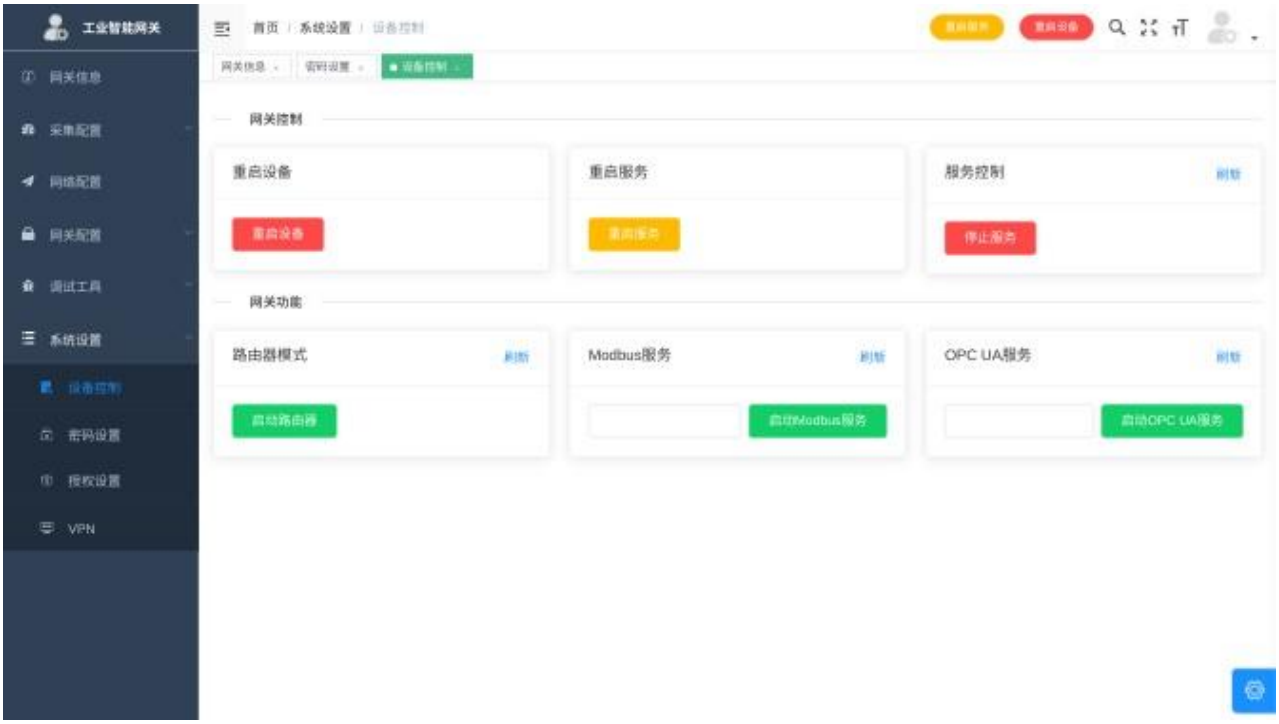


图 27：Modbus 输出

8.1 操作方法

1) 勾选需要输出到 Modbus 的变量；
2) 如果变量类型为空，需要手动调整变量类型；
3) 点击保存生成 Modbus 地址；
4) 可以导出为点位表；
5) 所有变量均存于保持寄存器。

8.2 协议解释

- 所有的 Int 均为 Int64 长整形；
- 浮点为 Double 双精度浮点型；

- 字符串长度最大为 100。

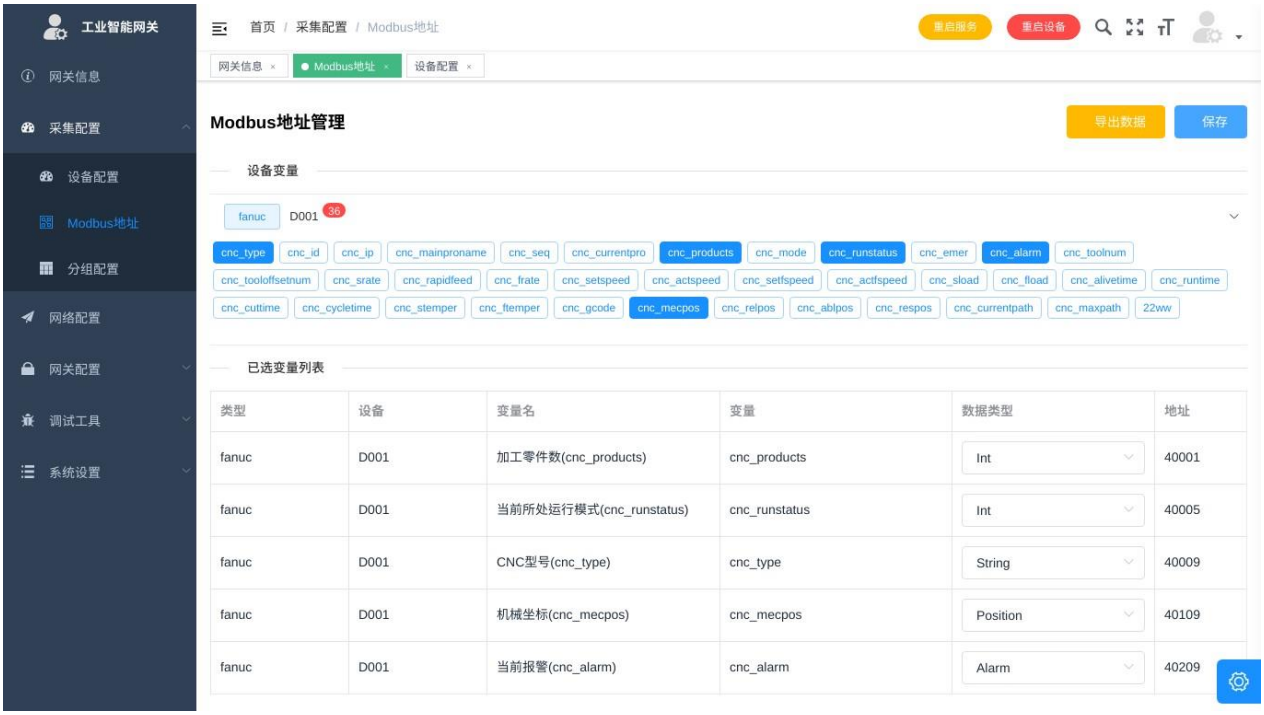


图 28：Modbus 输出

- Alarm 类型为一个复杂类型，地址前 4 位为报警数量，后面每四位为一个报警的报警编号，数据类型为 Int64，Alarm 总长度为 100 位.
  - 例如 cnc_alarm 的地址为 40209
  - 40209 读取 Int64 为报警数量
  - 40413 读取 Int64 为第一个报警的报警号
  - 40417 读取 Int64 为第二个报警的报警号
  - .....
- Position 类型为一个复杂类型，地址前 4 位为轴数，后面每八位为一个轴信息，每一个轴信息的前 4 位为轴名称，字符串类型，后四位为对应的值，Double 类型，Position 总长度为 100 位。
  - 例如 cnc_mecpos 的地址为 40109
  - 40109 读取 Int64 为轴数量
  - 40113 读取 String(4) 为第一个轴名称

– 40117 读取 Double 为第一个轴的值

– 40121 读取 String(4) 为第二个轴名称

– 40125 读取 Double 为第二个轴的值

–.....

9 常见问题

1) 为什么我访问不了网关页面?

测试一下网络是否可达，在您的主机上 ping 网关地址，如果地址不通，您需要配置电脑 IP 使电脑与网关在同一网段，如果能 ping 通，请联系售后

2) 忘记网关地址怎么办?

断开所有有线连接，插入 4G 卡，在网关管理平台通过 VPN 访问

3) 为什么采集不到数据?

- 检测设备地址是否正确
- 在调试页面 ping 设备地址看是否可达

10 支持采集设备分类

10.1 CNC

10.1.1 发那科(Fanuc)

支持发那科全系列，包括并不限于下列型号

FANUC Series 0i-MODEL A

FANUC Series 0i-MODEL B

FANUC Series 0i-MODEL C Note1)

FANUC Series 0i-MODEL D

FANUC Series 0i Mate-MODEL D

FANUC Series 0i-MODEL F

FANUC Series 0i Mate-MODEL F

FANUC Series 0i-PD

FANUC Series 0i-PF

FANUC Series 15/150-MODEL B

FANUC Series 15i/150i-MODEL A

FANUC Series 15i/150i-MODEL B

FANUC Series 16/160-MODEL B

FANUC Series 16/160-MODEL c

FANUC Series 18/180-MODEL B

FANUC Series 18/ 180-MODEL c

FANUC Series 21/210-MODEL B

FANUC Series 16/160i-MODEL A

FANUC Series 18i/180i-MODEL A

FANUC Series 21i/210i-MODEL A

FANUC Series 16i/160i-MODEL B

FANUC Series 18i/180i-MODEL B

FANUC Series 21i/210i-MODEL B

FANUC Series 16i/ 160i-P

FANUC Series 18i/180i-P

FANUC Series 16i/160i-L

FANUC Series 16i/160i-W

FANUC Series 18i/180i-W

FANUC Series 30i-MODEL A

FANUC Series 31i-MODEL A

FANUC Series 32i-MODEL A

FANUC Series 30i-MODEL B

FANUC Series 31i-MODEL B

FANUC Series 32i-MODEL B

FANUC Series 35i-MODEL B

FANUC Series 30i-P MODEL B

FANUC Series 31i-P MODEL B

FANUC Series 30i-L MODEL B

FANUC Series 31i-L MODEL B

FANUC Series 31i-W MODEL A

FANUC Series 31i-W MODEL B

FANUC Power Mate i-MODEL H

FANUC Power Mate i-MODEL D

FANUC Power Motion i-MODEL A

.

10.1.2 西门子(Siemens)

目前市面上的西门子系统主要有 808d，810d，802dsl，828d，828dsl，840d，840dsl。其中西门子官方支持 828d，828dsl，840dsl，且系统版本必须满足 sw4。5sp3。西门子官方在该版本系统内置 opcuaserver，客户需向西门子官方购买 opcua 授权进行通讯数据采集，每台费用在 3000 左右。其余西门子 CNC 型号官方均不支持通讯数据读写。本公司针对西门子驱动器进行研究解析分析，实现了针对西门子驱动器的免授权数据读写，支持了驱动器具备网口的 808d，802d，828d，82dsl，840dsl。同时也支持了 PCU 版本在 5。0 的 810d，840d 系统，支持 X86，arm 等硬件架构，可以轻松的在任意平台移植，满足工业网关的低成本高稳定的需求。

---

siemens808d

siemens810d

siemens802dsl

siemens828d

siemens828dsl

siemens840d(采集方式需要部署程序，有蓝屏风险，不建议采集但是可以采集)

siemens840dsl

10.1.3 三菱(Mitsubishi)

MitsubishiCNCM700 系列

MitsubishiCNCM700V 系列

MitsubishiCNCM70 系列

MitsubishiCNCM70V 系列

MitsubishiCNCE70 系列

MitsubishiCNCC70 系列

MitsubishiCNCM800 系列

MitsubishiCNCM80 系列

.

10.1.4 哈斯(Hass)

HassCNC 由于官方接口限制，只支持官方定义固定功能的宏程序打印。目前市面上的 HassCNC 分为 24 针串口以及网口两种，均支持宏程序打印功能，支持 X86，arm 等硬件架构，可在任意平台移植，满足工业网关的低成本高稳定的需求。

支持所有 24 针串口以及网口所有版本。

10.1.5 凯恩帝(knd)

支持系统版本信息 V4.3 以上的系统。

10.1.6 海德汉 (Heidenhain)

Heidenhain 530
Heidenhain 620
Heidenhain 640

10.1.7 兄弟 (Brother)

支持自带网口全系列Brother CNC

10.1.8 广州数控 (GSK)

GSK 988
GSK 980tdi
GSK 980mdi
GSK 980mdc
GSK 980tdc
GSK 25i

10.1.9 新代 (Syntec)

.

支持 V2 及以上版本

10.116.10以前的版本不支持V2
10.116.10x ~ 10.116.24x V3
10.116.24x ~ 10.116.36 V4
10.116.36x ~最 新

10.2 PLC

10.2.10 ModBus

支持标准 ModBus 协议，支持 TCP 和 RTU 两种模式。

10.2.11 西门子(Siemens)

SiemensSmart200/300/400/1200/1500

10.2.12 三菱(Mitsubishi)

MitsubishiFx 系列/Q 系列/A 系列

10.2.13 欧姆龙(Omron)

OmronCH 系列

10.2.14 罗克韦尔(AB)

全系列

10.3 ROBOT

10.3.1 库卡(KUKA)

全系列

ABB

全系列

---

11 点位地址

11.1 Fanuc

内置变量

点位名称	点位地址	描述	数据类型
加工零件数	cnc_products	cnc 生产件数	Number
CNC 的 IP 地址	cnc_ip	CNC 的 IP 地址	String
开机时间	cnc_alivetime	数控系统工作时间秒	Number
运行时间	cnc_runtime	加工程序运行总时间秒	Number
切削时间	cnc_cuttime	加工总时间秒	Number
循环时间	cnc_cycletime	单次加工时间秒	Number
CNC 型号	cnc_type	说明机床类型	String
当前刀具号	cnc_toolnum	当前加工主轴上的刀具编号	String
当前刀补编号	cnc_tooloffsetnum	当前加工刀具对应刀补号	String
执行的 NC 主程序号	cnc_mainproname	cnc 当前加工执行的主程序号	String
当前加工程序语句号	cnc_seq	cnc 当前执行程序内容的语句号	Number
当前加工程序内容	cnc_currentpro	cnc 当前执行程序/指定程序名称内容	String
当前所处操作模式	cnc_mode	操作面板操作模式对应的模式	Enum
是否急停	cnc_emer	设备是否处于急停状态	Boolean
绝对坐标	cnc_ablpos	编程前会指定一个原点，建立坐标系	List[Object]
机械坐标	cnc_mecpos	机床厂家设定的坐标系	List[Object]
相对坐标	cnc_relpos	在刀尖当前所在位置建立的坐标系	List[Object]
剩余坐标	cnc_respos	剩余坐标是 G01 的切削终点的距离	List[Object]
CNC ID	cnc_id	CNC ID	String
快速移动倍率	cnc_rapidfeed	操作面板上快速移动倍率旋钮对应数值	Number
主轴设定速度 S	cnc_setspeed	宏变量中程序运行时定义的设定主轴转速	Number
主轴实际转速 S	cnc_actspeed	加工中主轴实际转速	Number
进给设定转速 S	cnc_setfspeed	宏变量中程序运行时定义的进给主轴转速(矢量值)	Number
进给实际转速 S	cnc_actfspeed	加工中进给轴实际转速矢量值	Number
主轴负载	cnc_sload	主轴负荷值	Number
进给轴负载	cnc_fload	各进给轴的负荷值	Number
主轴温度	cnc_stemper	主轴电机温度	Number
伺服温度	cnc_ftemper	伺服电机温度	List[Object]
主轴倍率	cnc_srate	操作面板主轴旋率对应的数值	Number

切削倍率	cnc_frate	操作面板上进给倍率旋钮对应数值	Number
加工状态	cnc_gcode	判断 G0/G1 或其他 G 代码加工状态或加工坐标系	String
当前所处运行模式	cnc_runstatus	当前设备运行状态	Enum
程序列表	cnc_programlist	程序列表	List[Object]
最大通道	cnc_maxpath	多通道设备最大通道数	Number
当前通道	cnc_currentpath	多通道设备当前通道	Nubmer
报警	cnc_alarm	报警	List[Object]

添加点位

变量类型	变量名示例	变量地址示例	数据类型
宏变量	macro1	100	Number
设备 PLC	plc1	G0012	Number
设备诊断	plc1	4000	Number/List[Object]
设备参数	pamar1	6217	Number/List[Object]

数据字典

cnc_mode cnc_mode

```
{
    "Mdi":0,
    "Memory":1,
    "None":2,
    "Edit":3,
    "Handle":4,
    "Jog":5,
    "TeachinJog":6，
    "TeachinHandle":7，
    "Incfeed":8，"Refer-
    ence":9，"Re-
    mote":10，"Other":11
}
```

cnc_runstatus cnc_runstatus

```
{
    "RESET":0,
    "STOP":1,
    "HOLD":2,
    "START":3,
    "MSTR":4,
    "Other":5
}
```

cnc_emer cnc_emer

```
{
    "NotEmergency":0,
    "Emergency":1,
    "Reset":2,
    "Wait":3,
    "Others":4
}
```

11.2 Mitsubishi

内置变量

点位名称	点位地址	描述	数据类型
加工零件数	cnc_products	cnc 生产件数	Number
CNC 的 IP 地 址	cnc_ip	CNC 的 IP 地 址	String
系统日期	cnc_systemdate	数控系统工作时间时	Number
系统时间	cnc_systemtime	加工程序运行总时间秒	Number
开机时间	cnc_alivetime	数控系统工作时间秒	Number
运行时间	cnc_runtime	加工程序运行总时间秒	Number
切削时间	cnc_cuttime	加工总时间秒	Number
循环时间	cnc_cycletime	单次加工时间秒	Number
CNC 型号	cnc_type	说明机床类型	String
当前刀具号	cnc_toolnum	当前加工主轴上的刀具编号	String
当前刀补 D 编号	cnc_tooloffsetdnum	当前刀补 D 编号	String
当前刀补 H 编号	cnc_tooloffsethnum	当前刀补 H 编号	String
执行的 NC 主程序号	cnc_mainproname	cnc 当前加工执行的主程序号	String
当前加工程序语句号	cnc_seq	cnc 当前执行程序内容的语句号	Number
当前加工程序内容	cnc_currentpro	cnc 当前执行程序/指定程序名称内容	String
当前所处操作模式	cnc_mode	操作面板操作模式对应的模式	Enum
是否急停	cnc_emer	设备是否处于急停状态	Boolean
绝对坐标	cnc_ablpos	编程前会指定一个原点，建立坐标系	List[Object]
机械坐标	cnc_mecpos	机床厂家设定的坐标系	List[Object]
相对坐标	cnc_relpos	在刀尖当前所在位置建立的坐标系	List[Object]
剩余坐标	cnc_respos	剩余坐标是 G01 的切削终点的距离	List[Object]
快速移动倍率	cnc_rapidfeed	操作面板上快速移动倍率旋钮对应数值	Number

主轴设定速度 S	cnc_setspeed	宏变量中程序运行时定义的设定主轴转速	Number
主轴实际转速 S	cnc_actspeed	加工中主轴实际转速	Number
进给设定转速 S	cnc_setfspeed	宏变量中程序运行时定义的进给主轴转速（矢量值）	Number
进给实际转速 S	cnc_actfspeed	加工中进给轴实际转速矢量值	Number
主轴负载	cnc_sload	主轴负荷值	Number
进给轴负载	cnc_fload	各进给轴的负荷值	Number
主轴倍率	cnc_srate	操作面板主轴旋率对应的数值	Number
切削倍率	cnc_frate	操作面板上进给倍率旋钮对应数值	Number
加工状态	cnc_gcode	判断 G0/G1 或其他 G 代码加工状态或加工坐标系	String
当前所处运行模式	cnc_runstatus	当前设备运行状态	Enum
报警	cnc_alarm	报警	List[Object]

添加点位

变量类型	变量名示例	变量地址示例	数据类型
宏变量	macro1	100	Number
设备 PLC	plc1	R0012	Number

数据字典

cnc_mode cnc_mode

```
{
    "JOG模式中":0,
    "手轮模式中":1,
    "增量模式中":2,
    "手动任意进给模式中":3，
"参考点返回模式中":4，"
自动初始设定模式中":5，
    "JOG手轮同时模式中":6,
    "Null1":7,
```

```
    "内存模式中":8,
    "纸带模式中":9,
    "在线运行模式中":10,
    "MDI模式中":11,
    "Null2":12,
    "Null3":13,
    "子系统控制l运行模式中":14，
    "Null4":15

}
```

---

cnc_runstatus cnc_runstatus

```
{
    "RESET":0,
    "STOP":1,
    "HOLD":2,
    "START":3,
    "Others":4
}
```

---

cnc_emer cnc_emer

```
{
    "NotEmergency": 0,
    "Emergency": 1
}
```

11.3 Siemens

内置变量

点位名称	点位地址	描述	数据类型
CNC ID	cnc_id	CNC ID	String
CNC 型号	cnc_type	说明机床类型	String
软件版本	cnc_softver_version	软件版本	String
CNC 的 IP 地址	cnc_ip	CNC 的 IP 地址	String
报警	cnc_alarm	报警	List[Object]
是否急停	cnc_emer	设备是否处于急停状态	Boolean
执行的 NC 主程序号	cnc_mainproname	cnc 当前加工执行的主程序号	String
加工零件数	cnc_products	cnc 生产件数	Number
当前所处操作模式	cnc_mode	操作面板操作模式对应的模式	Enum
循环时间	cnc_cycletime	单次加工时间秒	Number
当前刀具号	cnc_toolnum	当前加工主轴上的刀具编号	String
机械坐标	cnc_mecpos	机床厂家设定的坐标系	List[Object]
相对坐标	cnc_relpos	在刀尖当前所在位置建立的坐标系	List[Object]
剩余坐标	cnc_respos	剩余坐标是 G01 的切削终点的距离	List[Object]
主轴设定速度 S	cnc_setsspeed	宏变量中程序运行时定义的设定主轴转速	Number

主轴实际转速 S	cnc_actsspeed	加工中主轴实际转速	Number
进给设定转速 S	cnc_setfspeed	宏变量中程序运行时定义的进给主轴转速（矢量值）	Number
进给实际转速 S	cnc_actfspeed	加工中进给轴实际转速矢量值	Number
主轴负载	cnc_sload	主轴负荷值	Number
进给轴负载	cnc_fload	各进给轴的负荷值	Number
当前所处运行模式	cnc_runstatus	当前设备运行状态	Enum
主轴倍率	cnc_srate	操作面板主轴旋率对应的数值	Number
切削倍率	cnc_frate	操作面板上进给倍率旋钮对应数值	Number
电机温度	cnc_temper	电机温度	Number
轴名称	cnc_axisname	轴名称	String
剩余时间	cnc_remtime	剩余时间	Number
母线电压	cnc_voltage	母线电压	Number
实际电流	cnc_current	实际电流	Number
刀具半径补偿编号	cnc_toolradiusnum	刀具半径补偿编号	Number
刀具长度补偿编号	cnc_toollengthnum	刀具长度补偿编号	Number
刀具 X 补偿	cnc_toollengthx	刀具 X 补偿	Number
刀具 Z 补偿	cnc_toollengthz	刀具 Z 补偿	Number
刀沿位置	cnc_tooledg	刀沿位置	Number

数据字典

cnc_mode　　cnc_mode

```
{
    "JOG": 0,
    "TEACHIN":1,
    "MDA":2,
    "AUTO":3,
    "REPOS":4,
    "REFPOINT":5,
    "VAR":6,
    "INC":7,
    "OTHER":8
}
```

cnc_runstatus　　cnc_runstatus

```
{
    "RESET":0,
    "STOP":1,
```

```
    "HOLD":2,
    "START":3，"SPEN-
     DLE_CW_CCW":4,
     "Others":5
}
```

11.4 KND

内置变量

点位名称	点位地址	描述	数据类型
加工零件数	cnc_products	cnc 生产件数	Number
CNC 的 IP 地址	cnc_ip	CNC 的 IP 地址	String
CNC ID	cnc_id	CNC ID	String
CNC 型号	cnc_type	说明机床类型	String
切削时间	cnc_cuttime	加工总时间秒	Number
循环时间	cnc_cycletime	单次加工时间秒	Number
开机时间	cnc_alivetime	数控系统工作时间秒	Number
运行时间	cnc_runtime	加工程序运行总时间秒	Number
绝对坐标	cnc_ablpos	编程前会指定一个原点，建立坐标系	List[Object] 机
械坐标	cnc_mecpos	机床厂家设定的坐标系	List[Object]
相对坐标	cnc_relpos	在刀尖当前所在位置建立的坐标系	List[Object] 报
警	cnc_alarm	报警	List[Object]
执行的 NC 主程序号	cnc_mainproname	cnc 当前加工执行的主程序号	String
快速移动倍率	cnc_rapidfeed	操作面板上快速移动倍率旋钮对应数值	Number

切削倍率	cnc_frate	操作面板上进给倍率旋钮对应数值	Number
当前所处运行模式	cnc_runstatus	当前设备运行状态	Enum
当前所处操作模式	cnc_mode	操作面板操作模式对应的模式	Enum
软件版本	cnc_softversion	软件版本	String
FPGA 版本	cnc_fpgaversion	FPGA 版本	String
T 型图版本	cnc_ladderversion	T 型图版本	String
是否就绪	cnc_isready	是否就绪	Boolean
未就绪原因	cnc_notreadyreason	未就绪原因	String
当前工件坐标系	cnc_curworkcoor	当前工件坐标系	String
当前系统坐标系	cnc_curworkcoorsystem	当前系统坐标系	String
手动切削倍率	cnc_jograte	手动切削倍率	Number

11.5 Brother

内置变量

点位名称	点位地址	描述	数据类型
加工零件数	cnc_products	cnc 生产件数	Number
报警	cnc_alarm	报警	List[Object]
绝对坐标	cnc_ablpos	编程前会指定一个原点，建立坐标系	List[Object]
机械坐标	cnc_mecpos	机床厂家设定的坐标系	List[Object]
相对坐标	cnc_relpos	在刀尖当前所在位置建立的坐标系	List[Object]
剩余坐标	cnc_respos	剩余坐标是 G01 的切削终点的距离	List[Object]
CNC 版本	cnc_version	CNC 版本	String
CNC 的 IP 地 址	cnc_ip	CNC 的 IP 地 址	String
CNC ID	cnc_id	CNC ID	String
快速移动倍率	cnc_rapidfeed	操作面板上快速移动倍率旋钮对应数值	Number
切削倍率	cnc_frate	操作面板上进给倍率旋钮对应数值	Number
主轴倍率	cnc_srate	操作面板主轴旋率对应的数值	Number
当前刀具号	cnc_toolnum	当前加工主轴上的刀具编号	Number
主轴实际转速 S	cnc_actspeed	加工中主轴实际转速	Number
进给实际转速 S	cnc_actfspeed	加工中进给轴实际转速矢量值	Number
当前所处运行模式	cnc_runstatus	当前设备运行状态	Enum
当前所处操作模式	cnc_mode	操作面板操作模式对应的模式	Enum
执行的 NC 主程序号	cnc_mainproname	cnc 当前加工执行的主程序号	String
CNC 型号	cnc_type	说明机床类型	String

数据字典

cnc_mode cnc_mode

```
{
    "MANU":0,
    "MDI":1,
    "MEM":2,
    "EDIT":3,
    "MANU+MDI":4,
    "MEM+EDIT":5,
}
```

---

cnc_runstatus cnc_runstatus

```
{
    "电源关闭":1,
    "等待":2,
    "工作":3,
    "停止":4,
    "错误":5,
}
```

---

12 HTTP 接口列表

12.1 用户

登陆请求

- Method: POST
- URL: /user/login
- Headers：Content-Type:application/json
- Body:

```
{
  "username" :"admin",
  "password" :"password"
}
```

返回

- 204 OK
- 401 未登录

获取用户信息请求

- Method: GET
- URL: /user/info
- Headers：Content-Type:application/json JWT:token

返回

- 200

```
{
  "avatar":"",
  "name":"admin"
}
```

- 401 未登录

修改密码请求

- Method: GET
- URL: /user/info

- Headers：Content-Type:application/json JWT:token

返回

- 200

```
{
"avatar":"",
"name":"admin"
}
```

- 401 未登录

12.2 采集设备列表请求

- Method: GET
- URL: /device
- Headers：Content-Type:application/json JWT:token

返回

- 200

```
[
  {
      "baud": 9600,
      "data_bits": 8,
      "did": "FanucTest112",
      "enable": true,
      "id": 1,
      "interval": 1.0,
      "ip": "192.168.1.1",
      "items": [
          {
          "address": "",
          "address_type": "",
          "function_code": 3,
          "id": 1,
          "interval": 1，"item_name":
          "cnc_type"，"size": 0,
          "sub_address": 0,
```

```
            "value_type": 1
            }
        ],
        "model": "all",
        "parity": "N"， "port":
        8193， "protocol": "fa-
        nuc",
        "serial": "/dev/ttyO2"，
        "slave_id": 1,
        "stop_bits": 1，
        "type": "fanuc"
    }
]
```

- 401 未登录

请求单个设备

请求

- Method: GET
- URL: /device/device_id
- Headers： Content-Type:application/json JWT:token

返回

- 200

```
{
    "baud": 9600,
    "data_bits": 8,
    "did": "FanucTest112"，
    "enable": true,
    "id": 1,
    "interval": 1.0,
    "ip": "192.168.1.1",
    "items": [
        {
        "address": "",
        "address_type": ""，
        "function_code": 3,
        "id": 1,
        "interval": 1， "item_name":
```

```
        "cnc_type"， "size": 0,
        "sub_address": 0,
        "value_type": 1
        }
    ],
    "model": "all",
    "parity": "N"， "port":
    8193， "protocol": "fa-
    nuc",
    "serial": "/dev/ttyO2"，
    "slave_id": 1,
    "stop_bits": 1，
    "type": "fanuc"
}
```


- 401 未登录

修改设备
请求

- Method: PUT
- URL: /device/device_id
- Headers：Content-Type:application/json JWT:token
- Body:


```
{
    "baud": 9600,
    "data_bits": 8,
    "did": "FanucTest112"，
    "enable": true， "interval":
    1.0,
    "ip": "192.168.1.1",
    "items": [
        {
        "address": "",
        "address_type": ""，
        "function_code": 3,
        "id": 1,
        "interval": 1， "item_name":
        "cnc_type"， "size": 0,
        "sub_address": 0,
        "value_type": 1
```

```
        }
    ],
    "model": "all",
    "parity": "N"， "port":
    8193， "protocol": "fa-
    nuc",
    "serial": "/dev/ttyO2"，
    "slave_id": 1,
    "stop_bits": 1，
    "type": "fanuc"
}
```

返回

- 204 OK
- 400 请求参数错误
- 404 未找到资源
- 409 授权数量不匹配

删除设备

请求

- Method: DELETE
- URL: /device/device_id
- Headers：Content-Type:application/json JWT:token

返回

- 204 OK
- 404 未找到资源

获取设备采集数据

请求

- Method: GET
- URL: /device/device_id/info
- Headers：Content-Type:application/json JWT:token

返回

- 200

```
{
    "cnc_respos": {
        "ts": 1566384648053,
        "value": [
            {
```

```
                "axis": "X",
                "value": 0.0
            },
            {
                "axis": "Y",
                "value": 0.0
            },
            {
                "axis": "Z",
                "value": 0.0
            }
        ]
    },
    "cnc_type": {
        "ts": 1566384649570,
        "value": "FANUC 0i MF"
    }
}
```


- 404 未找到资源

通过设备名控制变量
请求

- Method: PUT
- URL: /device/device_id/control_by_name
- Headers： Content-Type:application/json JWT:token
- Body:


```
{
    "key": "M100",
    "value": 1,
    "value_type": 1
}
```


返回

- 204 OK
- 400 请求参数错误
- 404 未找到资源

- 409 授权数量不匹配

通过设备 **ID** 控制变量
请求

- Method: PUT
- URL: /device/device_id/control_by_id
- Headers：Content-Type:application/json JWT:token
- Body:

```
{
  "key": "M100",
  "value": 1,
  "value_type": 1
}
```

返回

- 204 OK
- 400 请求参数错误
- 404 未找到资源
- 409 授权数量不匹配

操作设备临时变量
请求

- Method: PUT
- URL: /device/device_id/temp_value
- Headers：Content-Type:application/json JWT:token
- Body:

```
[
  {
      "key": "M100",
      "action": "get",
      "value": 111,
      "per": 1
  }
]
```

- key 是变量名称
- action 是操作方式,get,set,delete
- value 要写入的值,get,delete 方法忽略
- per 指定要写入的区域,0 redis,1 sqlite redis 变量重启失效,sqlite 位持久保存

返回

- 204 OK
- 400 请求参数错误
- 404 未找到资源
- 409 授权数量不匹配

13 售后服务

13.1 技术服务和质保期服务计划

产品质保期：12 个月。

产品售后服务内容：主要针对网关产品的使用，机床采集端口配置说明，以及其它产品相关配置提供售后服务；机床端参数配置需根据实际情况到现场进行服务，用户需额外支付相应项目调试服务费，具体根据“项目设备清单”而定。

服务方式：电话、电子邮件、远程操控方式、现场服务。