RS485 TO POE ETH (B) MQTT And JSON User Manual

From Waveshare Wiki
Jump to: navigation, search

(/wiki/File:RS485_TO_POE_ETH01.jpg)

Overview

- MQTT and JSON can be used alone or together. JSON supports Modbusconvert RTU format to JSON format.

Main features:

1. Use the MQTT-based protocol to establish a connection with the server, and use the form of subscription to publish data communication.
2. Support independent design and automatic collection of Modbus RTU registers.
3. Support the conversion of specific Modbus register content into JSON format and send it regularly and actively.
4. Support adding device ID, time, and any string in JSON format.
5. Support embeddings in JSON format.
6. Support NTP protocol, and get the time automatically.
7. Support unsigned data and signed data, support decimal point representation, and

support 4-byte length data.

8. All configurations can be completed in interface configuration, and the user's independent configuration does not need to be customized.

JSON Examples

MODBUS RTU To JSON

- Modbus RTU to JSON can realize an automatic collection of Modbus RTU tables, and is automatically uploaded to the cloud server following the JSON format.
- Here we explain this usage with a specific case.

MODBUS Table

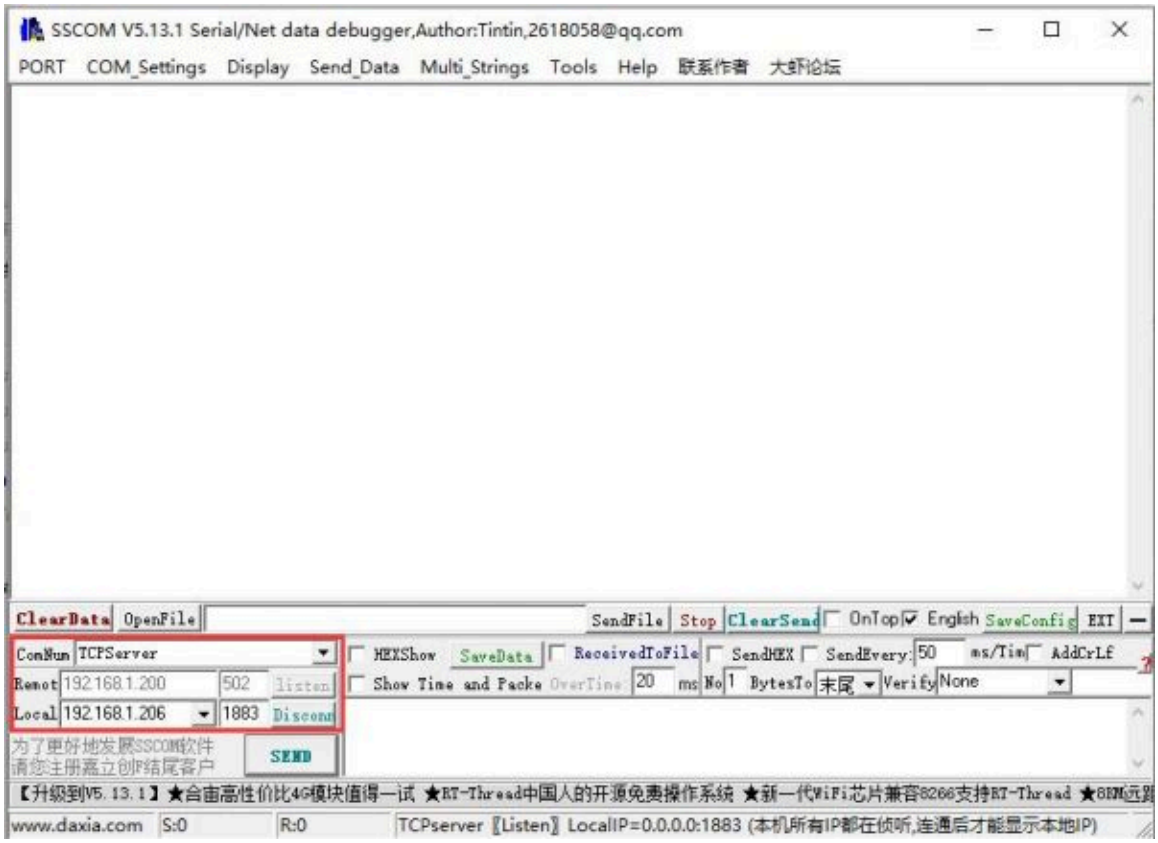
- Suppose there is a Modbus table with a function code of 3 and an address of 1. Its register addresses and parameter names are as follows. Where the byte length is 4, it means that 2 registers need to be read continuously.

Resister Address	Parameter	Byte Length	Note
0	Current total active power	4	Unsigned, keep 2 decimal places
97	Phase A voltage	2	Unsigned, keep 1 decimal place
98	Phase B voltage	2	
99	Phase C voltage	2	
100	Phase A current	2	Unsigned, keep 2 decimal place
101	Phase B current	2	
102	Phase C current	2	
119	Frequency	2	
356	Phase A active power	4	Unsigned, keep 3 decimal place
358	Pahse B active power	4	
360	Phase C active power	4	
362	Total active power	4	

- The so-called signed means that the highest bit of 2 bytes or 4 bytes is the sign bit, for example, 0xFFFF will be recognized as -1. Keeping 2 decimal places means that after the data is converted as an integer, the decimal point moves from the rightmost to the left 2 digits.

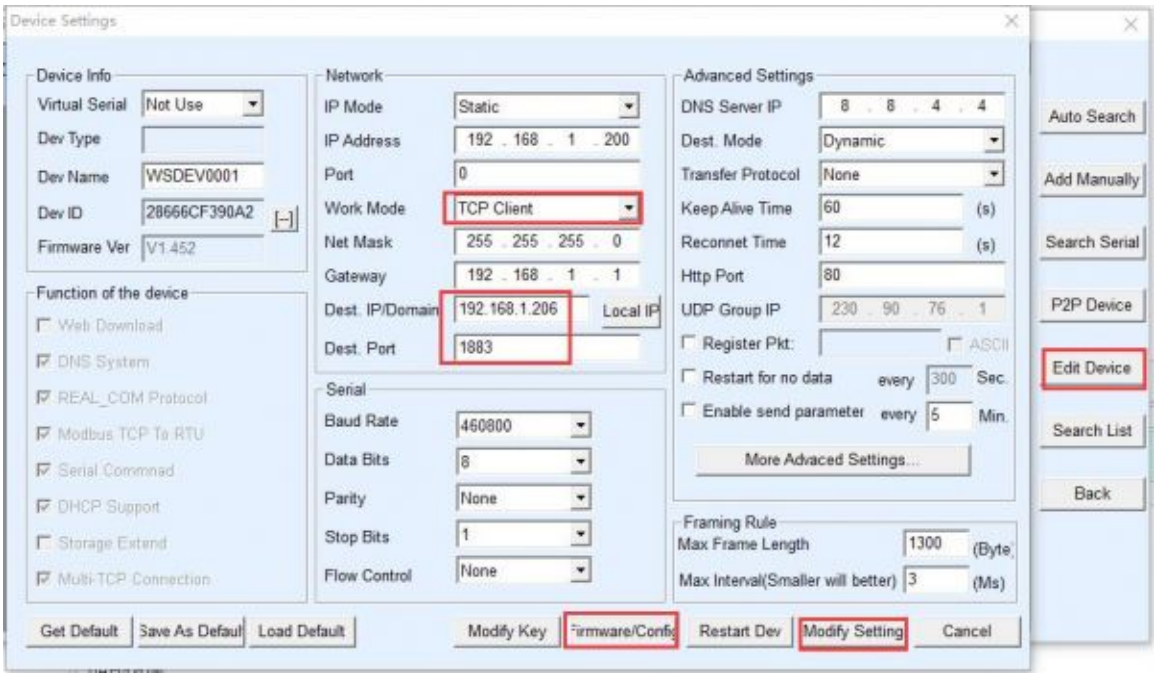
Device Config

- We configure the device as a client.
- Use SSCOM to monitor a TCP server on port 1883 of the local computer.



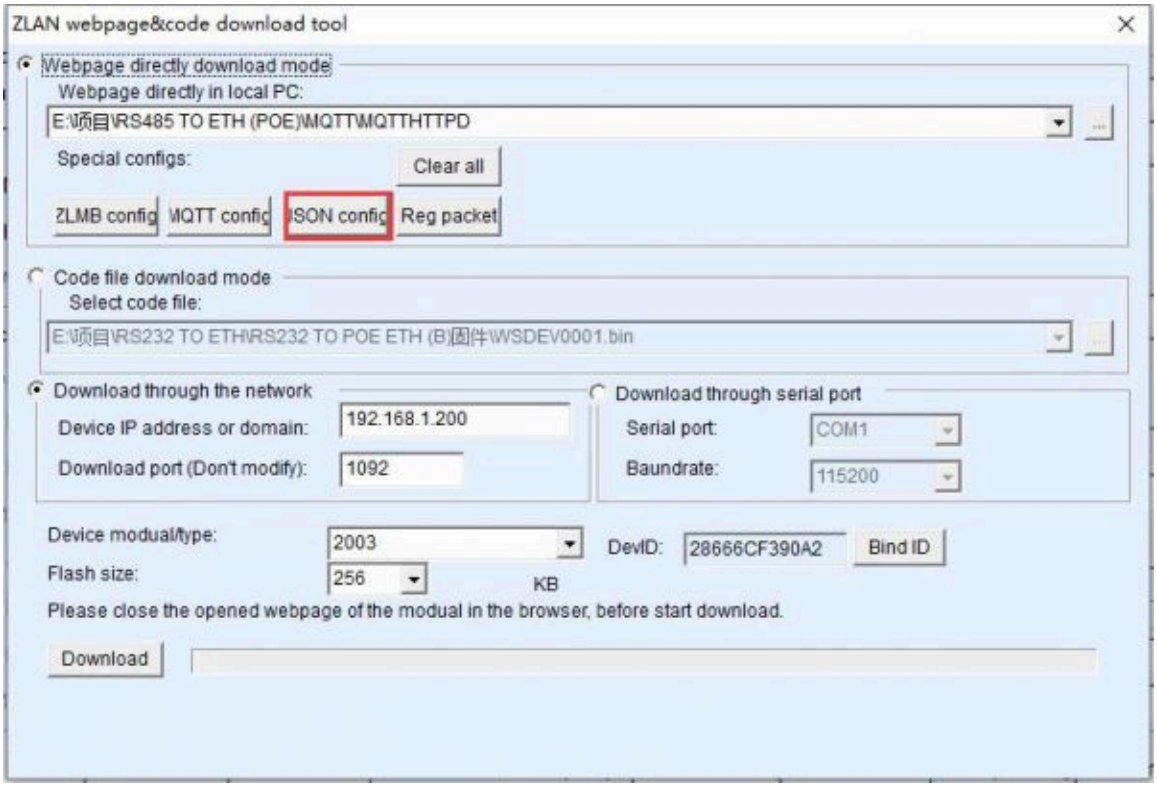
(/wiki/File:RS485_TO_POE_ETH03.jpg)

- Configure the device via Vircom:



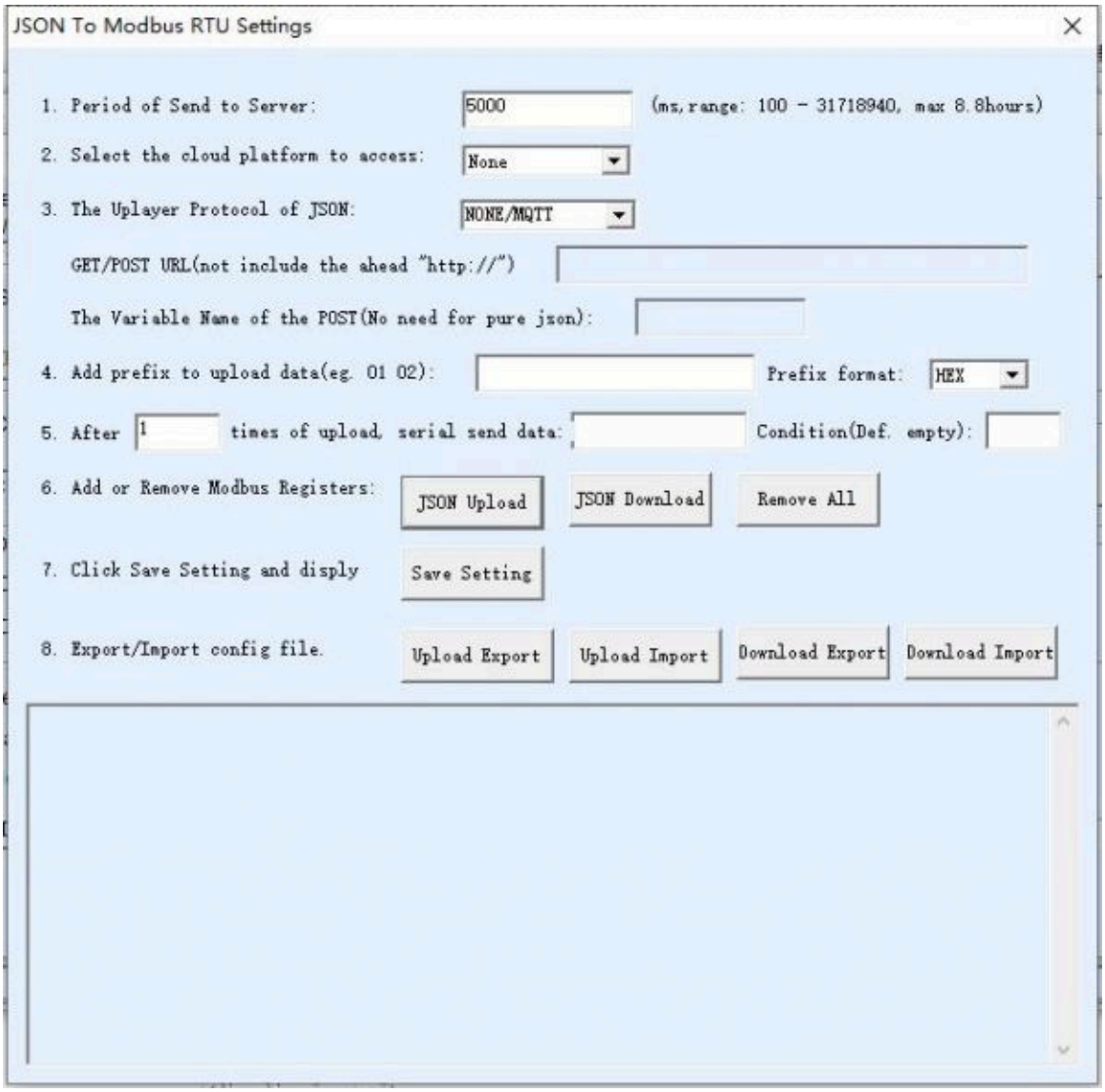
(/wiki/File:RS485_TO_POE_ETH04.jpg)

- Click Modify configuration to connect the device to the SocketDlgTest tool. Enter device editing again dialog box. Click the "Firmware and Configuration" button.



(/wiki/File:RS485_TO_POE_ETH05.jpg)

- First click "Web Directory Download" to enter the configuration download mode. Then select a new empty directory, such as the MQTTHTTPD directory. To prevent the previous design from remaining, please click the "Remove All" button first, so that the previous design content can be removed. The design file will be saved in this directory and can be downloaded to the device by clicking the "Download" button later.
- Click the "JSON Configuration" button.



(/wiki/File:RS485_TO_POE_ETH06.jpg)

- The parameters here are as follows:

1. **Period of Send to Server:** The default JSON data is sent to the server every time. The server is the destination IP set in the device configuration interface just now, and the unit is milliseconds.
2. **Select the cloud platform to access:** provide functions such as connecting massive devices to the cloud, bidirectional message communication between devices and the cloud, batch device management, remote control, and monitoring, OTA upgrades, device linkage rules, etc., and can flexible transfer device data to Huawei Other cloud services.
3. **The Uplayer Protocol of JSON:** three options (NONE/MQTT, HTTP POST, HTTP GET), choose NONE/MQTT, and do not need to set the domain name, address, and variable name below, choose the other two to fill in according to the actual use.
4. **GET/POST URL:** users can choose whether to add or not, which is mainly used to distinguish devices.
5. **Add prefix to upload data:**This is mainly used to judge a cycle, and this function can be used according to requirements.
6. **Add or remove Modbus registers:** After clicking, you can design the Modbus registers, and you can also view the current content. Delete means deleting all the Modbus registers

designed, which is convenient for restarting the design.

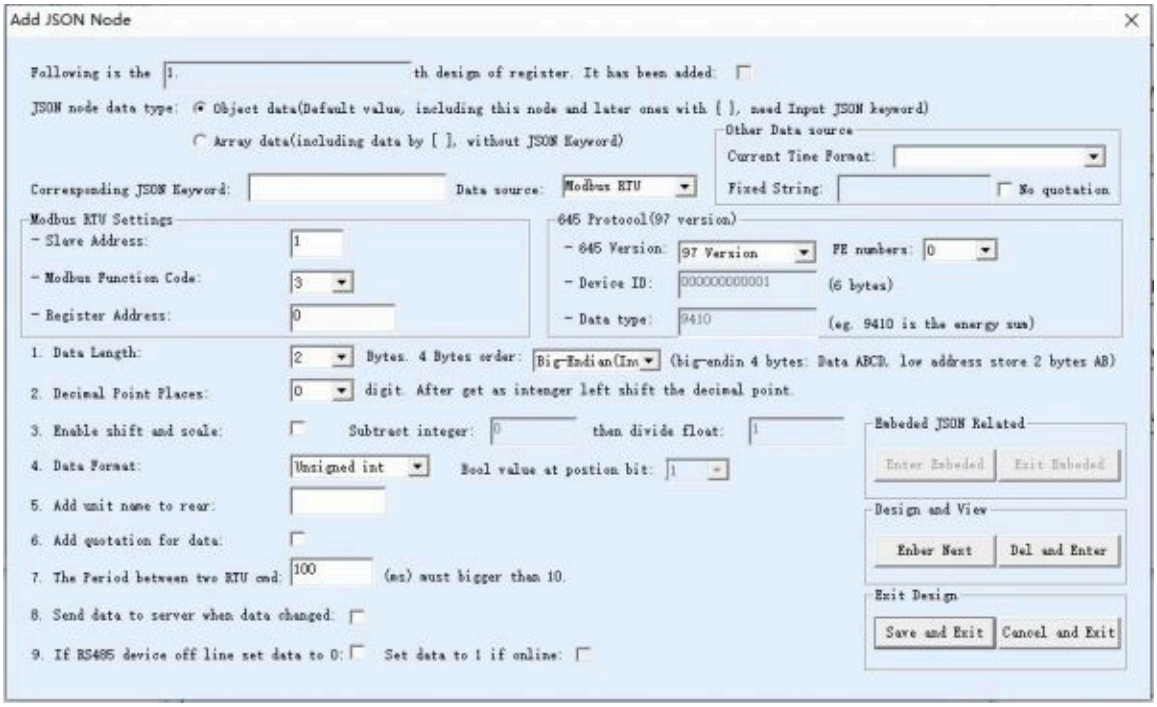
7. **Save setting:** After the design is completed, only click this button to save the data to the download directory just now, and then download it to the device.

8. **Export/import EXCELL config file:** easy to export and import JSON configuration.

- Now click on the "Add/View" button. For the first row of the previous Modbus table:

Register Address	Parameter	Byte Length	Note
0	Current total power	4	Unsigned, keep 2 decimal places

- The corresponding configuration is as follows:



(/wiki/File:RS485_TO_POE_ETH07.jpg)

- The parameters here are described as follows:

1. **The first register:** 1 here means that the current design interface is to configure the first register.

2. **It has been added:** If ticking indicates that it has been added, a tick will appear when viewing the configured information.

3. **JSON node data:** select object data and array data according to requirements.

4. **Corresponding JSON keyword:** corresponding to the keyword in the uploaded JSON. For example "CurrentW": 232.12.

5. **Slave address:** Modbus table address.

6. **Modbus function code:** currently supports 03 and 04 function codes.

7. **Register address:** the corresponding 0 here.

8. **Data length:** here corresponds to 4 bytes.

9. **Decimal point places:** 2 digits are reserved here.

10. **Enable shift and scale**.

11. **Data format:** here corresponds to an unsigned integer.

12. **Add unit name to rear:**: according to the data, set the corresponding unit.

13. **Add question to data:** Yes tick, no tick.

14. Serial port polling time: here is set to 100ms. It refers to the polling interval between this register and the next register, not the polling interval of this instruction.

15 and 16. Function selection, choose according to your needs.

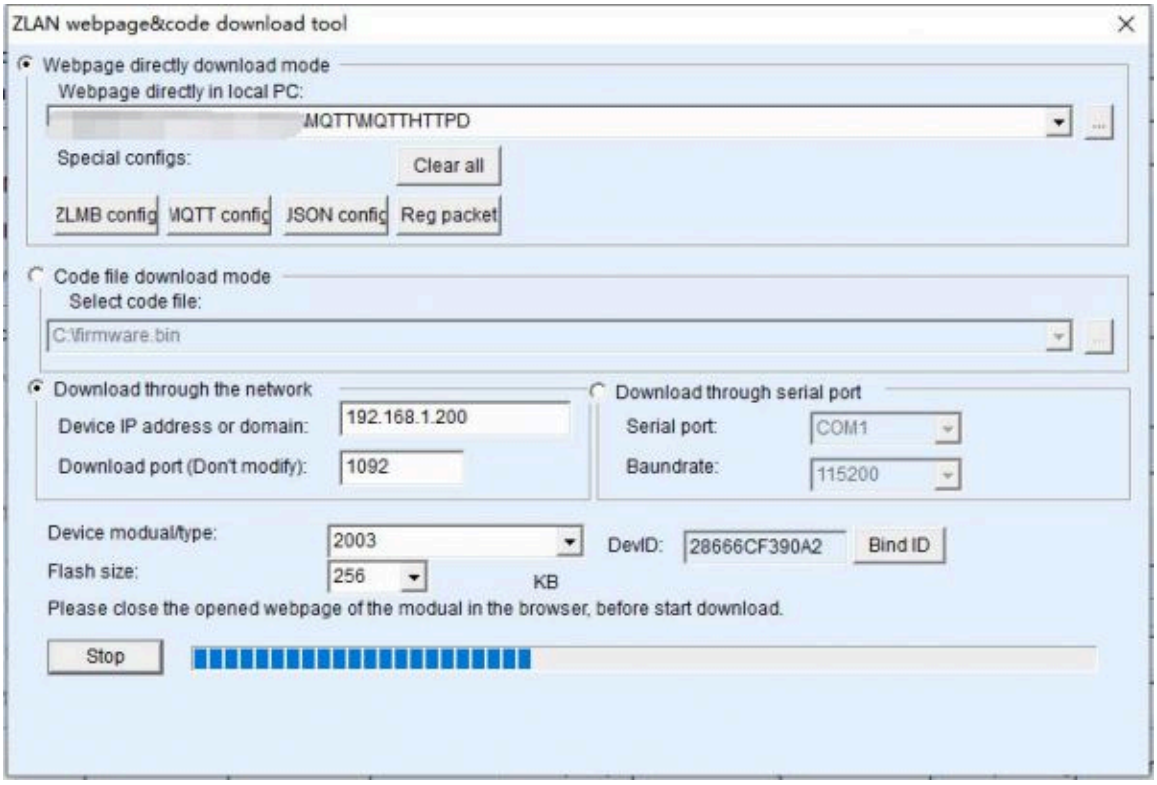
17. **Enable next:** click to enter the setting of the next register.

18. **Del and enter:**delete the one that has been added now, and configure the next register.

19. **Save and exit:** After completing the design, click "Save JSON Configuration" on the previous interface.

20. **Cancel and exit:** cancel all current designs, if you want to view the design content, you can click this button to exit.

- Click the "Enable next" button here to continue designing other registers in the Modbus table. After designing all the registers in the table, click "Finish Design", and then click "Save and exit" to exit. Then click the "Download button" on the "Download web" page.

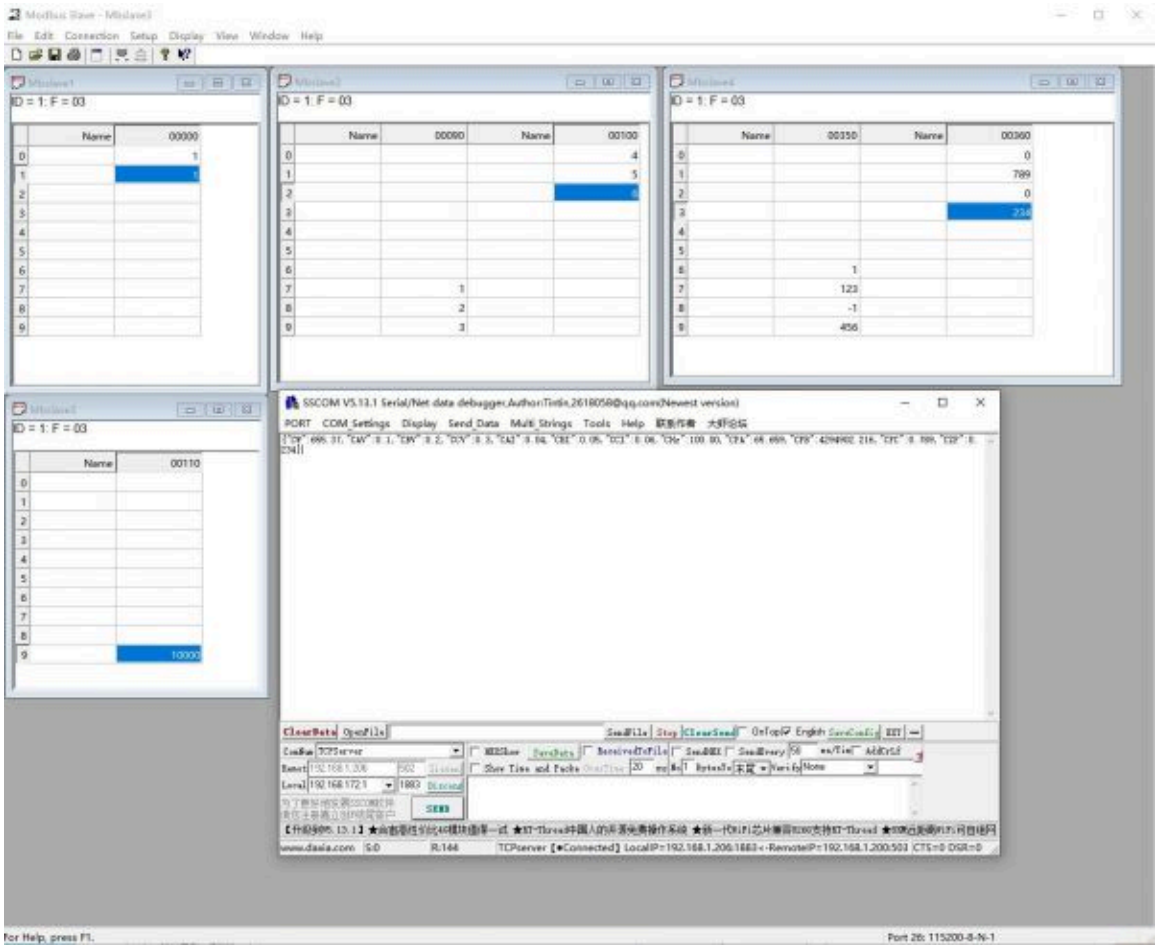


(/wiki/File:RS485_TO_POE_ETH08.jpg)

- Then click "OK" and the device will restart automatically. If it does not restart, please restart manually.

Create MODBUS Analog Meter

- Here, Modbus Slave is used to analog a table (a serial device is required to simulate Modbus devices, and I use USB TO RS232/485/TTL (https://www.waveshare.com/usb-to-rs232-485-ttl.htm) here).

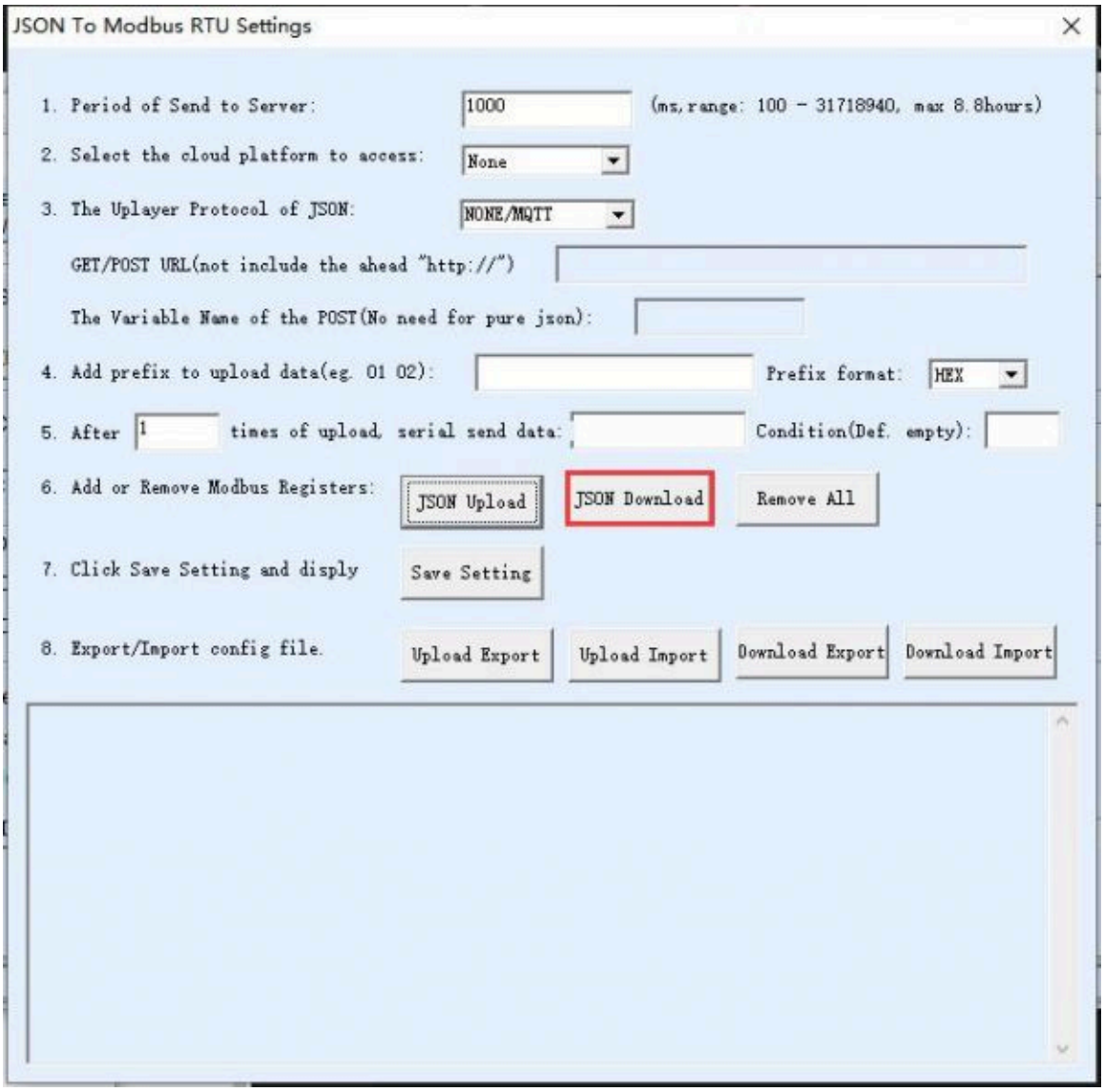


(/wiki/File:RS485_TO_POE_ETH018.jpg)

- The test results show that the instrument simulated by the Modbus slave tool can be collected by the gateway. At the same time, it can be sent to the server software of the SSCOM simulation according to the JSON format.

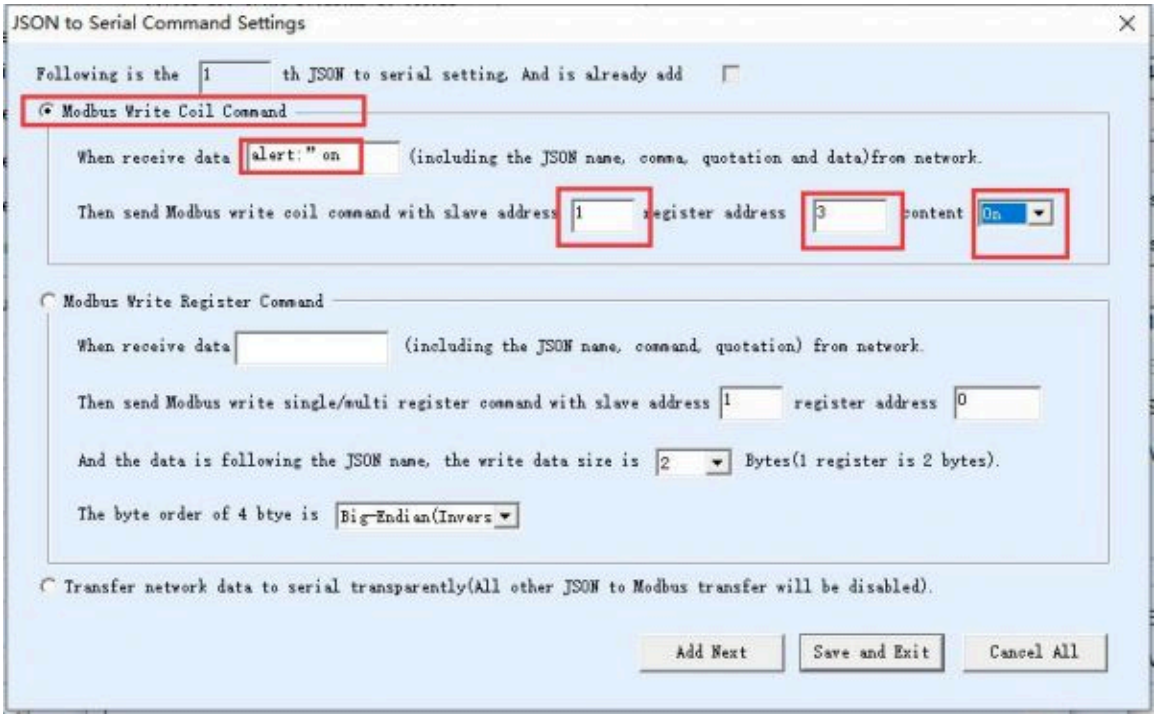
JSON To MODBUS RTU

- JSON to Modbus RTU supports 05/06/16 commands. If you need to use the 15 commands to set multiple coils, please use the 05 command multiple times.
- According to the length of the number of bytes, the system will automatically select 06 or 16 commands to send. Here is an example of setting the coil and setting the register respectively.
- If you receive {alert: "on" } JSON data, you need to use the 05 commands to set the station address 02, the coil starting from register 03. Then click "Send JSON" in the JSON to Modbus interface.



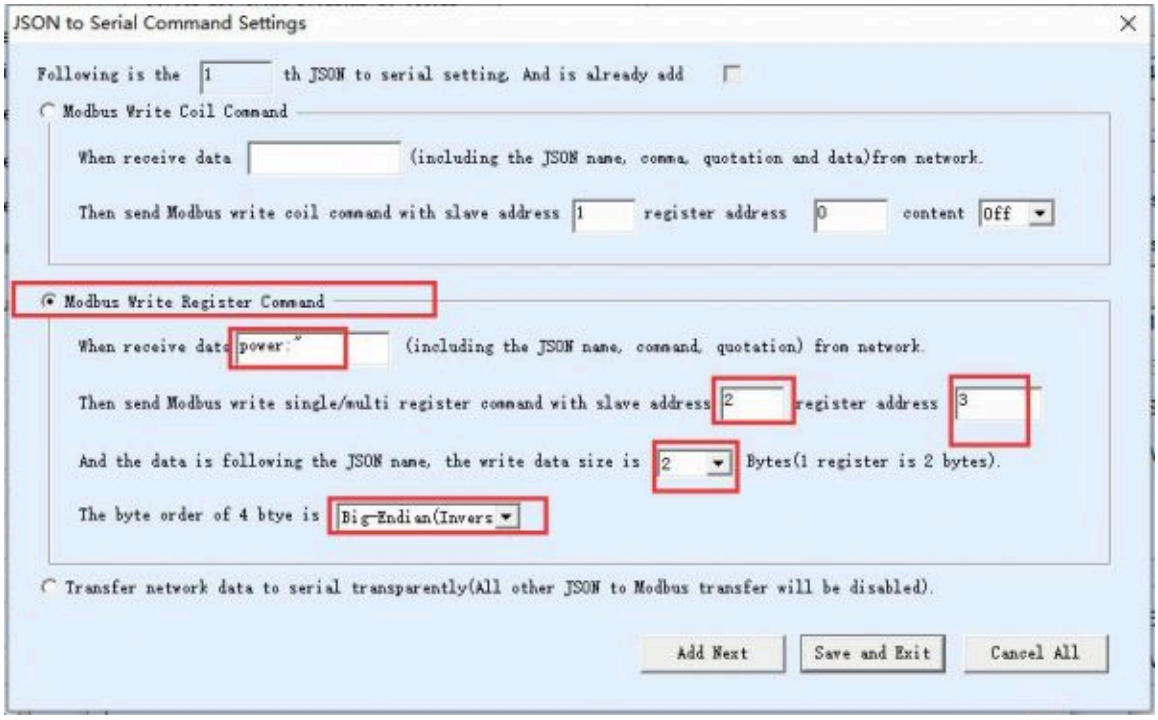
(/wiki/File:RS485_TO_POE_ETH019.jpg)

- The configuration interface is as follows: Note that the alert: "on here needs to be written.



(/wiki/File:RS485_TO_POE_ETH020.jpg)

- Click "Next" to add another delivery conversion, otherwise click "Save all and exit". After returning to the main interface, click "Save JSON Settings", and then click "Back". Then pay attention to clicking "Download" on the download interface. This completes the configuration.
- If {power: "12345"} is sent now, the value of power 12345 needs to be set to station address 2, register 3. Then set as follows:



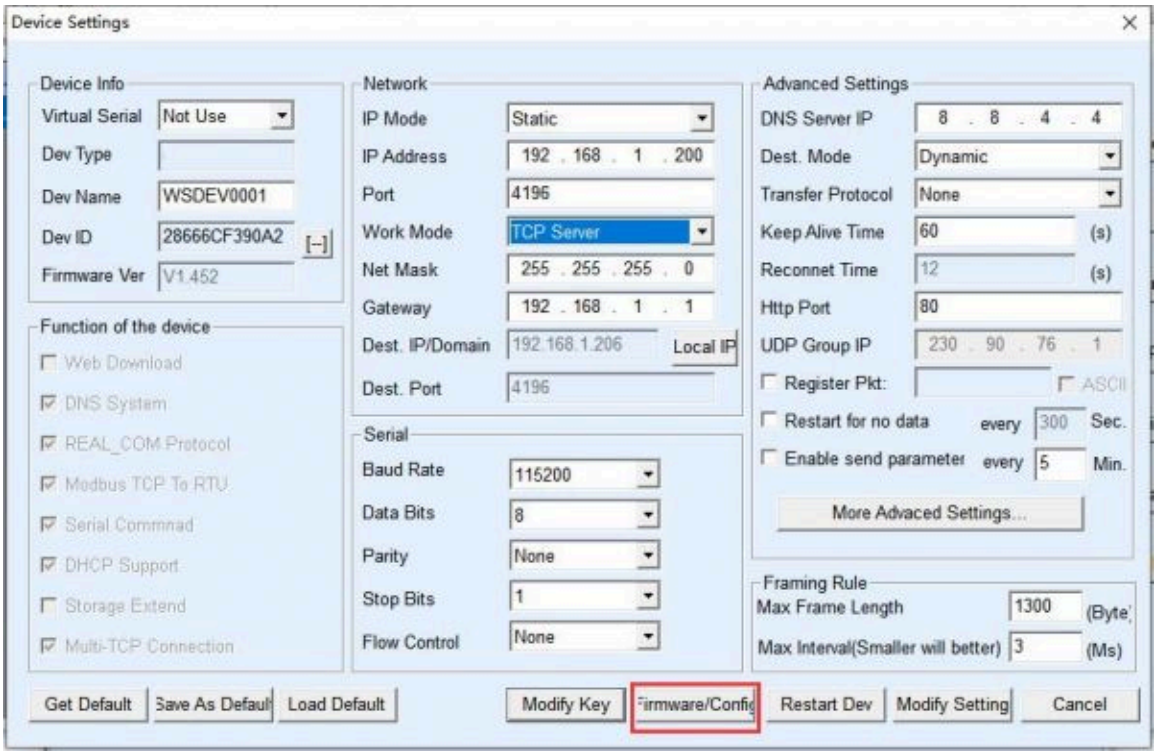
(/wiki/File:RS485_TO_POE_ETH021.jpg)

- Note that you only need to enter power:" for the keyword here, and you don't need to enter the following 12345, because this value changes, but you need to enter a colon. If there are quotation marks in the delivered data, you also need to enter quotation marks.

MQTT

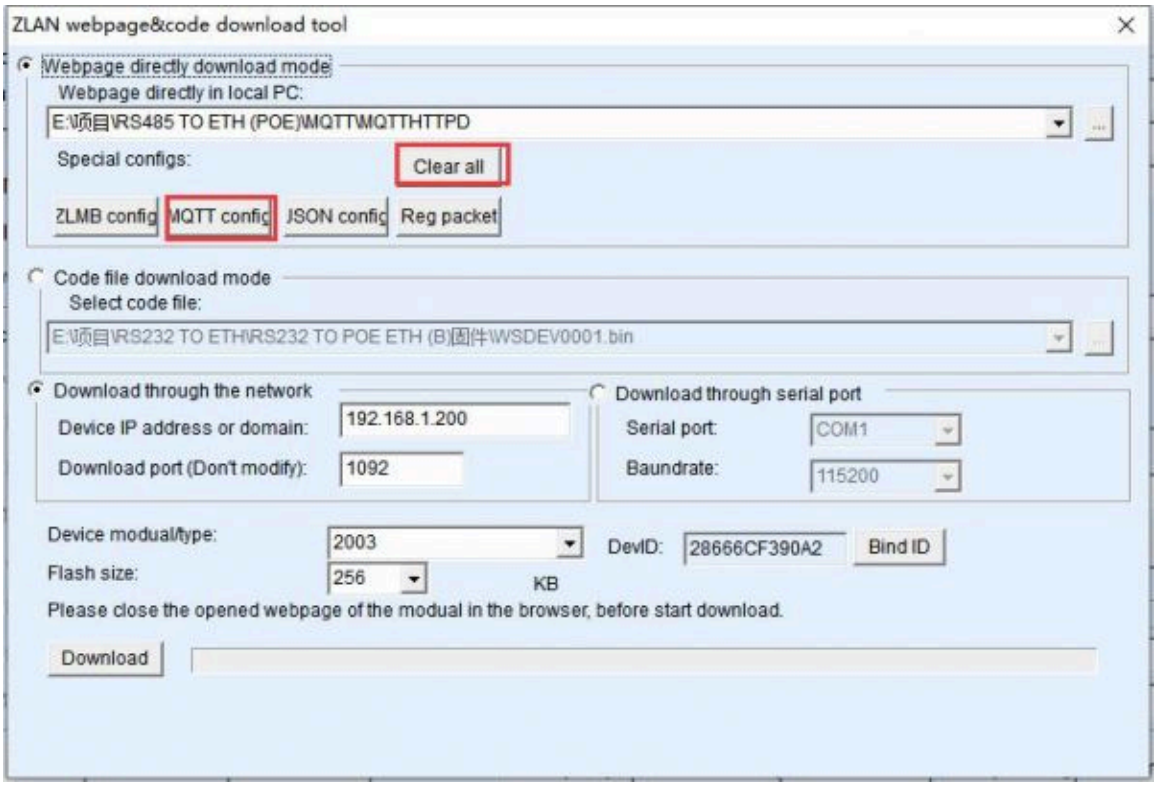
Device Config

- First search for the device, then click "Config":



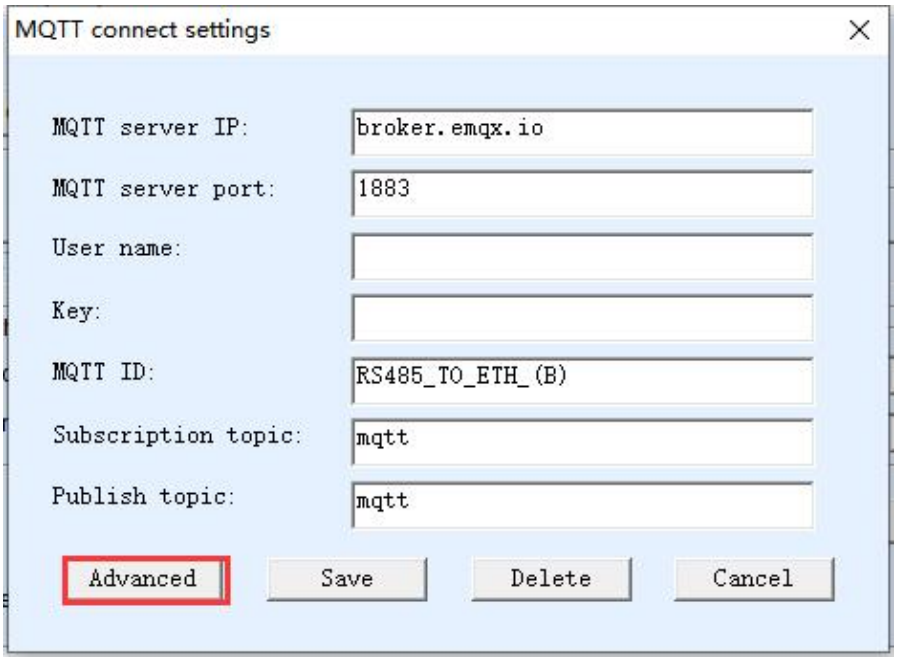
(/wiki/File:RS485_TO_POE_ETH025.jpg)

- Click "Firmware and Configuration" to pop up the configuration download and design dialog box:



(/wiki/File:RS485_TO_POE_ETH026.jpg)

- Here select "Webpage Directory Download", then select an empty directory, such as the MQTTHTTPD directory, and then click MQTT Configuration.



(/wiki/File:RS485_TO_POE_ETH027.jpg)

- The configuration instructions here are as follows:

1. **MQTT server IP:** fill in here the IP of the MQTT server, up to 30 characters.

2. **Username:** It is the username of the MQTT server.

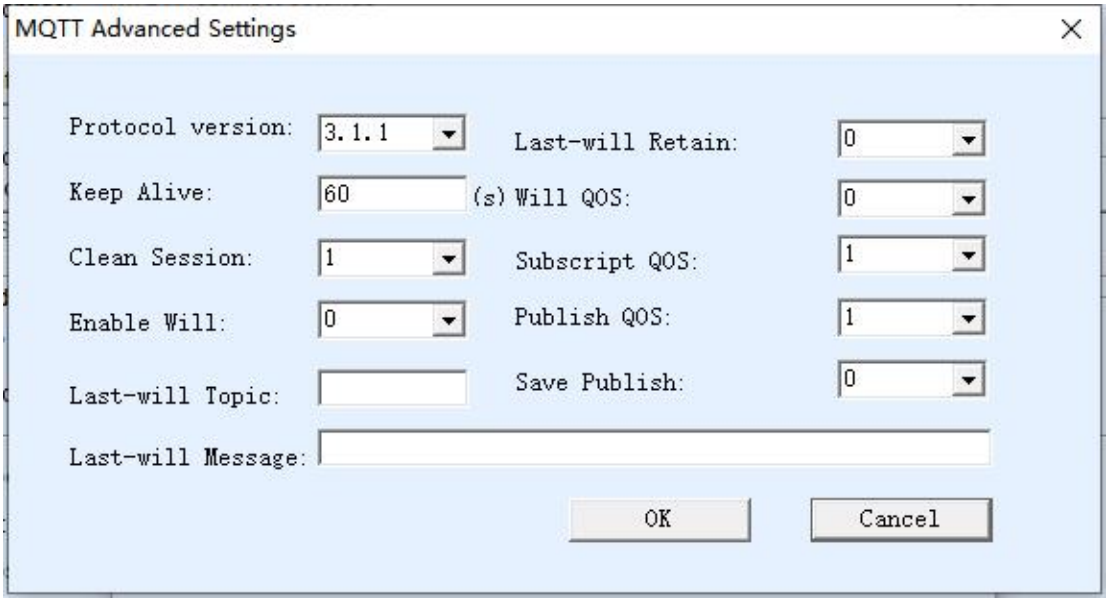
3. **Password:** It is the login password of this user.

4. **MQTT ID:** It is the client ID of MQTT.

5. **Subscription topic:** It is the topic that this device subscribes to. When other devices publish this topic, the server will send it to this device. If you are just publishing, generally you do not need to fill in this field.

6. **Publish topic:** The topic of the data sent to the server when the device converts the serial port to MQTT.
7. **Advanced:** used to configure advanced parameters.
8. **Save:** Click this button to save the design, and then click the "Download button" in the web download directory to download.

- Now click "MQTT Advanced Settings" (generally no need to configure advanced parameters):

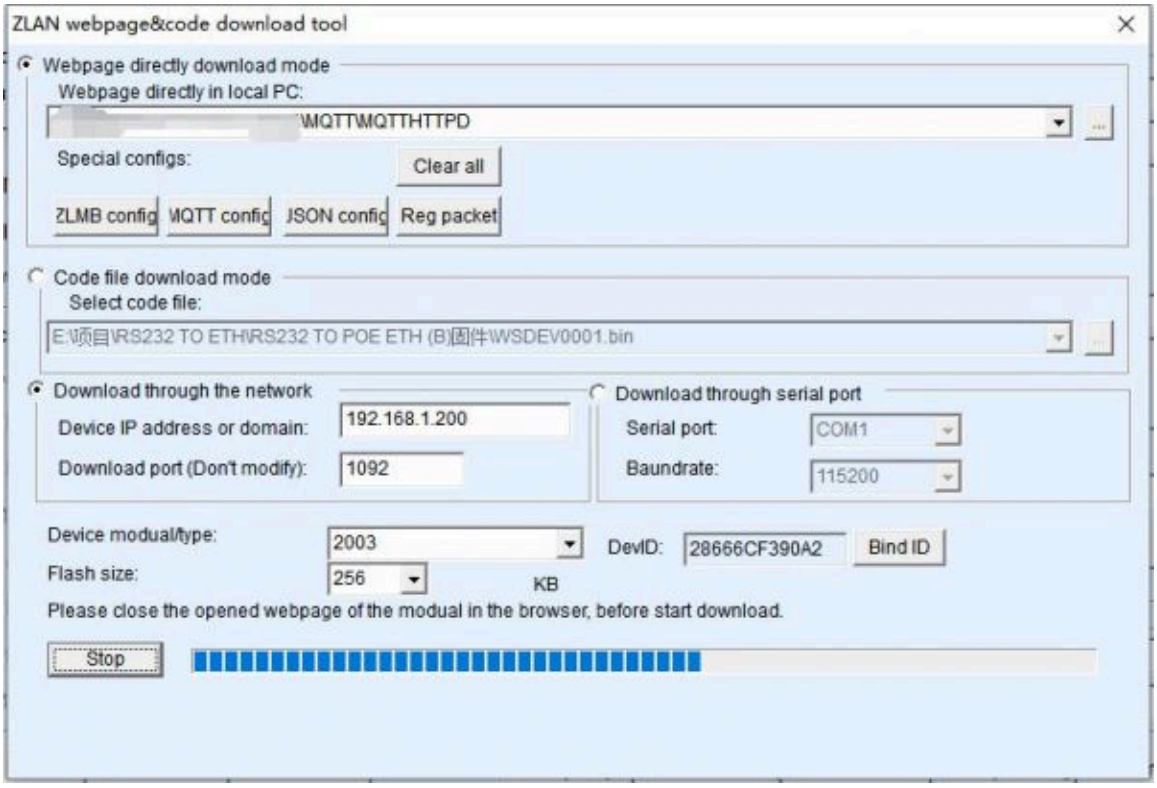


(/wiki/File:RS485_TO_POE_ETH028.jpg)

Note:

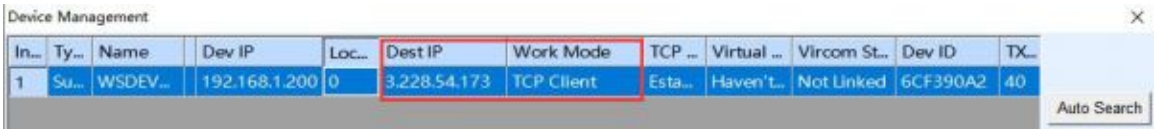
1. **Protocol version:** 3.1.1 is the current popular version, if you want to choose the 3.1 version, you can select it here.
2. **Keep alive:** MQTT heartbeat time, 10 sec at least, 60 sec by default.
3. **Clean session:** whether the server cleans the session after the client is offline.
4. **Enable will:** whether to enable the will.
5. **Last-will topic:** the topic of the last will.
6. **Last-will message:** the message of the last will.
7. **Last-will retain:** whether the server sends the last-will message when the client is offline.
8. **Will QOS:** the quality of the will.
9. **Subscript QOS:** the quality of the subscription. Sometimes you need to set it to 0 so as to avoid an offline state when it reloads.
10. **Publish QOS:** the quality of the publishing message from the client. Sometimes you need to set it to 0 so as to avoid an offline state when it reloads.
11. **Save publish:** whether to save the last message in the server. (if there is a new client subscription, it will send it to the client.)

- We do not modify any advanced parameters here, just click: "Save MQTT Config" and then click "Download".



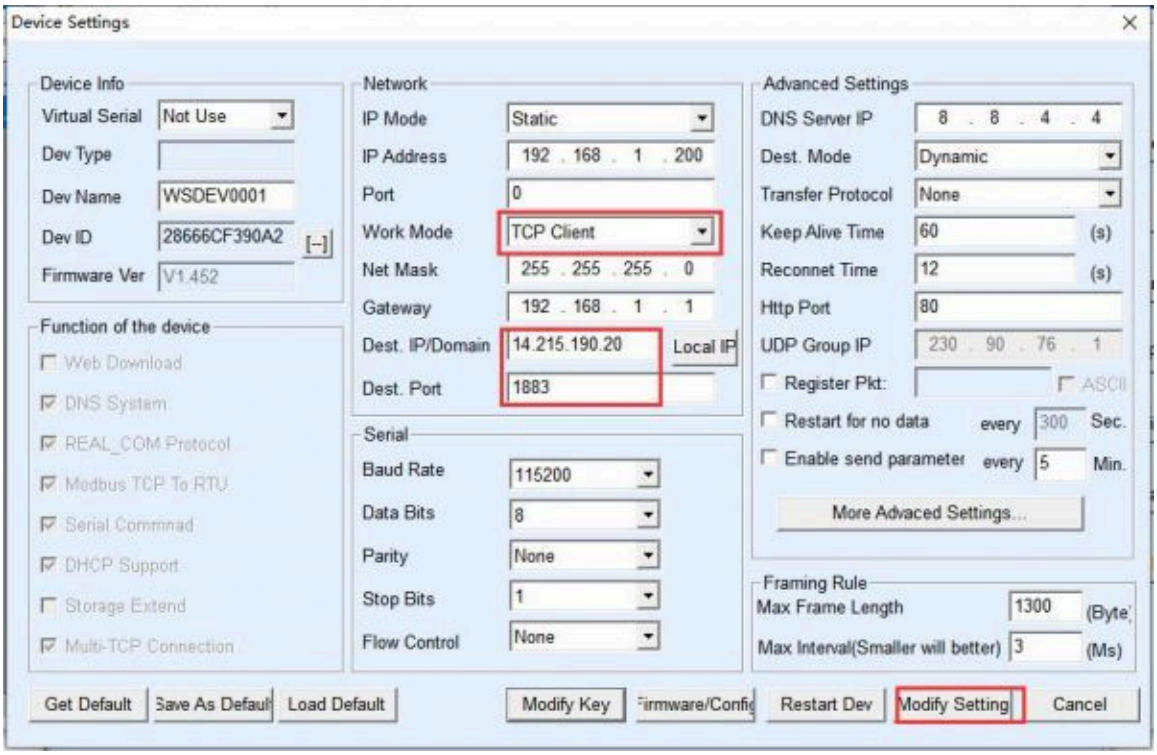
(/wiki/File:RS485_TO_POE_ETH029.jpg)

- Click OK after downloading, then it will return to the device management dialog. You can see that the device target IP, work mode, and target port have been automatically modified as MQTT settings.



(/wiki/File:RS485_TO_POE_ETH030.jpg)

- If it is not modified automatically, you need to set the target IP, work mode, and target port in the device settings. Then click "Modify Setting".



(/wiki/File:RS485_TO_POE_ETH031.jpg)

Data Transfer Test

- The LINK LED of the device will be on (the blue light in the middle) after connecting, which means the device has been connected to the MQTT server normally.
- Connect the UART device and RS485 TO POE ETH (B), and then open SSCOM.

(/wiki/File:RS485_TO_POE_ETH032.jpg)

- Open the serial port with the same baud rate as the device, and send the data "dev send", and then see the returned data "dev send" in the receiving window. This is because we publish the dev send message to the MQTT server on the MQTT topic. But at

the same time, our device also subscribes to the MQTT topic, so the server will immediately send us a subscription message, and the content of the subscription message is dev sent. This information is sent and downloaded as the payload of MQTT, and output from the serial port through transparent transmission. If other devices send information, this device can also receive data.

- Generally speaking, users can directly transmit serial port commands (such as Modbus RTU) directly to the MQTT server. In addition, the JSON function can also be used, which adopts the form of automatic Modbus RTU format collection and regular JSON format upload. In addition, you can also find Waveshare Electronics to customize some non-standard instruments and host computer protocol formats.

MQTT+JSON To MODBUS RTU

- Combining the above JSON and MQTT can achieve the following functions:

1. The MQTT-based protocol is used to establish a connection with the server, and data communication is performed in the form of subscription and publication.
2. Support independent design and automatic collection of Modbus RTU registers.
3. Support the conversion of specific Modbus register contents into JSON format for regular and active uploading.
4. Support adding device ID in JSON format to facilitate cloud identification of devices.

- If you need MQTT + JSON to Modbus RTU function, you can design MQTT and JSON separately, in no particular order. Don’ t click the "Clear Design" button after designing one type, and click the "Download" button to download the content to the device after designing both. Generally, you can manually restart the device after downloading it to load the settings.
- Show results:
- MQTT webpage (http://www.emqx.io/online-mqtt-client#/recent_connections) debug:

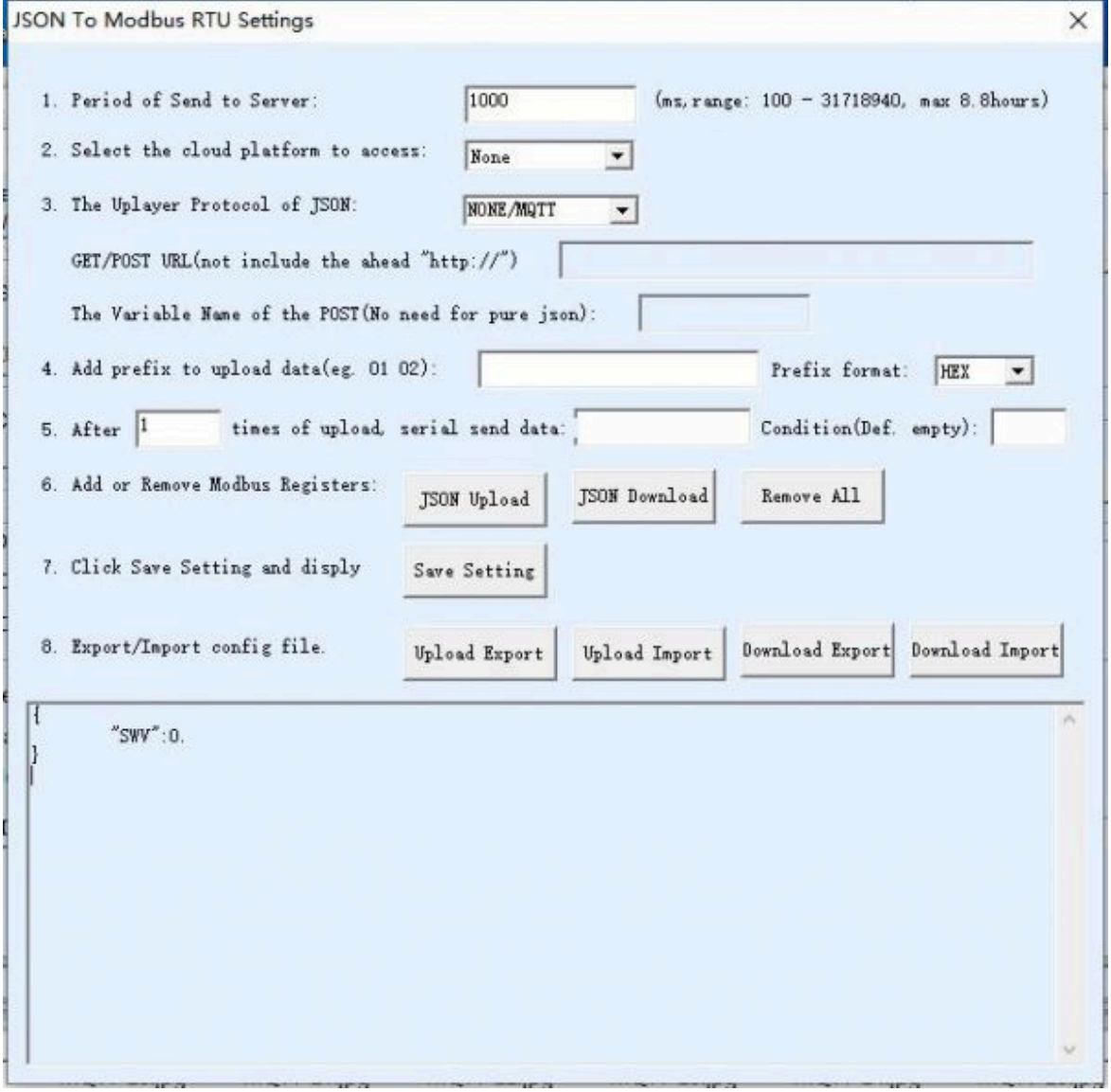

(/wiki/File:485POE_JSON_MQTT_33.jpg)

(/wiki/File:485POE_JSON_

(/wiki/File:485POE_JSON_MQTT_35.jpg

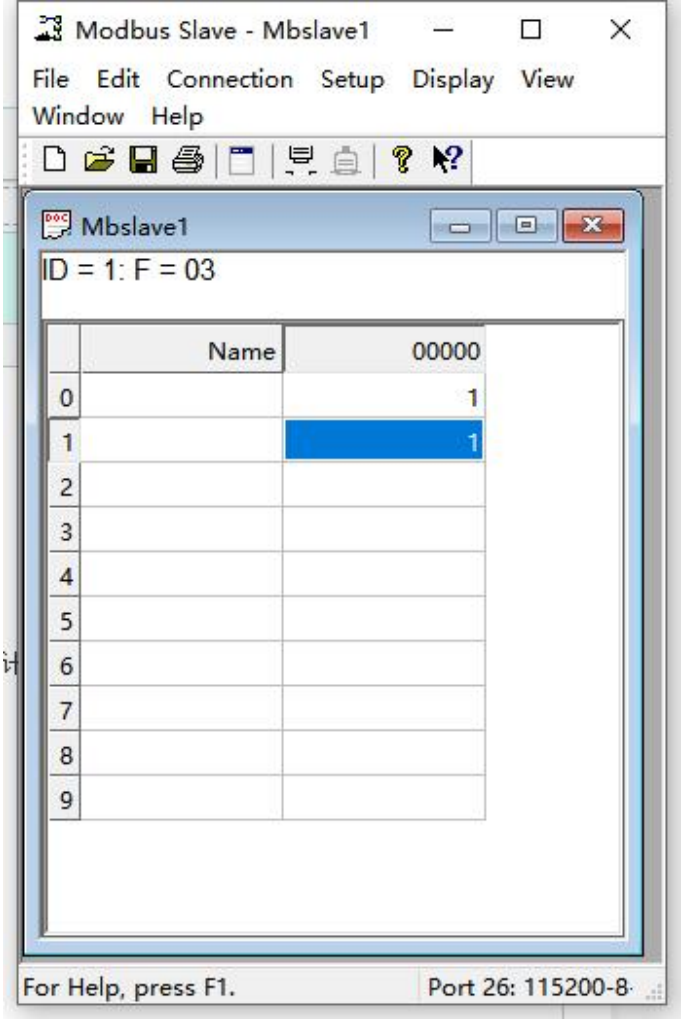
Figure MQTT web configuration

- JSON setting



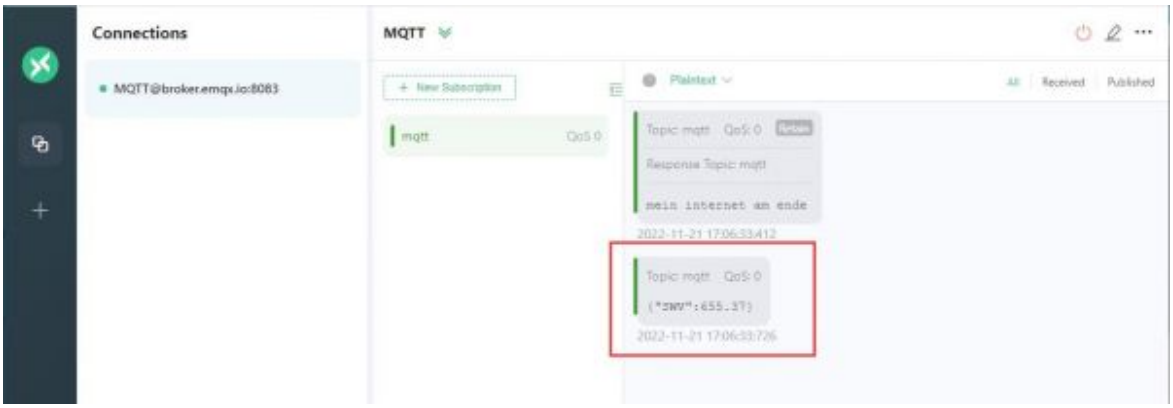
(/wiki/File:JSON_setting01.jpg)

- MODBUS analog table setting



(/wiki/File:JSON_setting02.jpg)

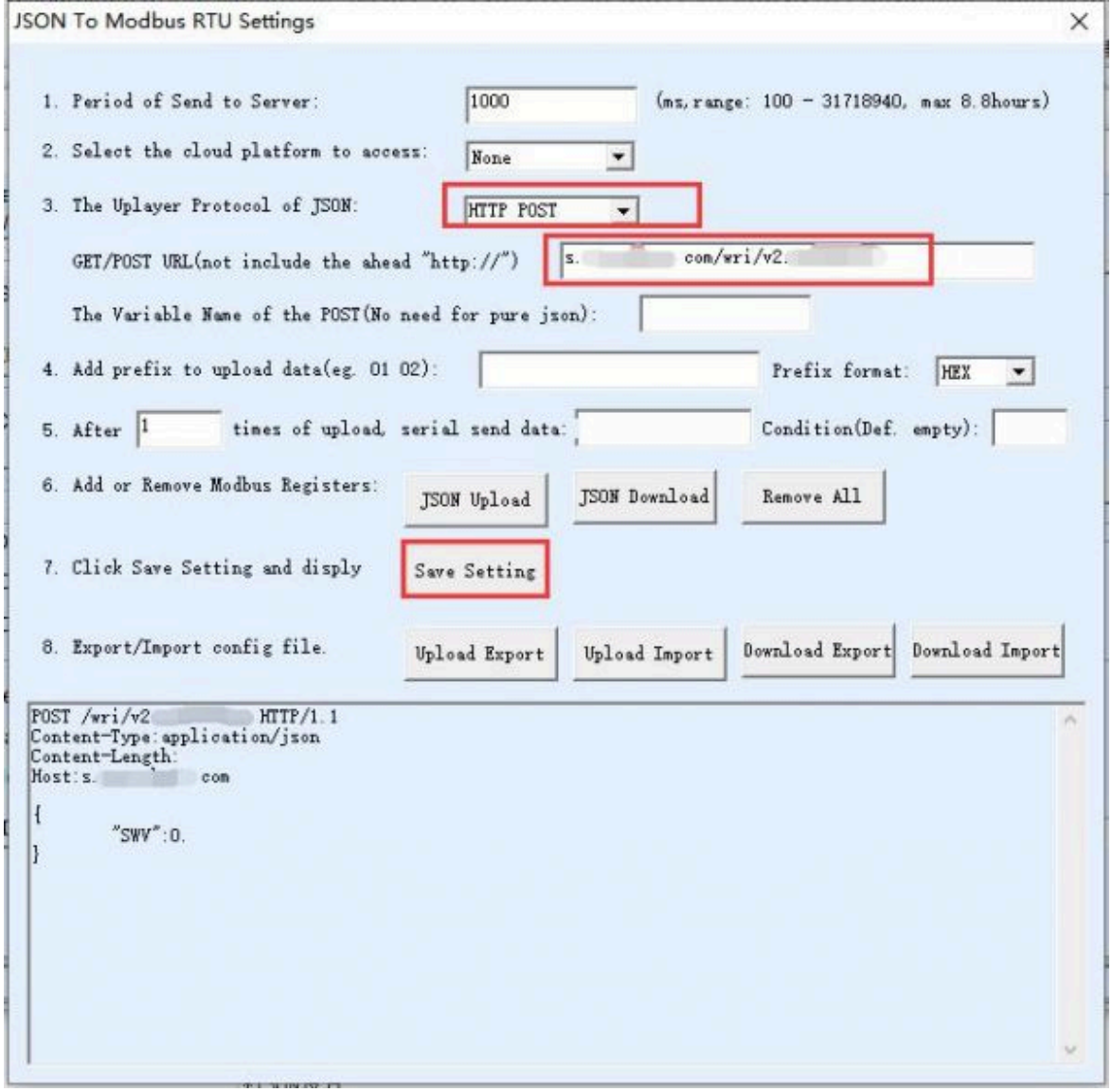
- Communication effect of MQTT + JSON to Modbus RTU



(/wiki/File:JSON_setting03.jpg)

HTTP POST/GET+JSON

- In addition to MQTT, the host computer protocol can also choose HTTP protocol, and upload data through POST and GET instructions. The following takes the POST command as an example to introduce.



(/wiki/File:POST%2BJSON01.jpg)

- The Vircom version adds two options in the JSON to Modbus RTU setting, as shown in the figure:

1. Upper-layer protocol of JSON: If it is protocol-free or MQTT protocol, please select the first item: "NONE/MQTT". If it is "HTTP POST", please select the second item "HTTP POST", if it is "HTTP GET", please select the third item "HTTP GET".

1. POST/GET URL: When selecting POST or GET, you must fill in the URL. For example, if the URL is http://s.a.com/wri/v2 (http://s.a.com/wri/v2), remove the previous "http://" and directly fill in "s.a.com/wri/v2".

- Other JSON structure design methods are the same as those introduced before, and when you click the "Save JSON Settings" button later, if "POST/GET" is selected, HTTP header format information will be added in front of the JSON data to support the HTTP transmission protocol.
- This "POST/GET" design method is simple and practical, and can easily and quickly transmit instrument data such as Modbus RTU to the server in the form of "HTTP POST/GET + JSON".