Make IoT Simpler

NJ-iot210 User Manual

4G Industrial Gateway

Updated: 2025-09-05

Translation note: the English body text below is translated from the Chinese original. Embedded product photos and engineering drawings are preserved from the source artwork where used.

Contents

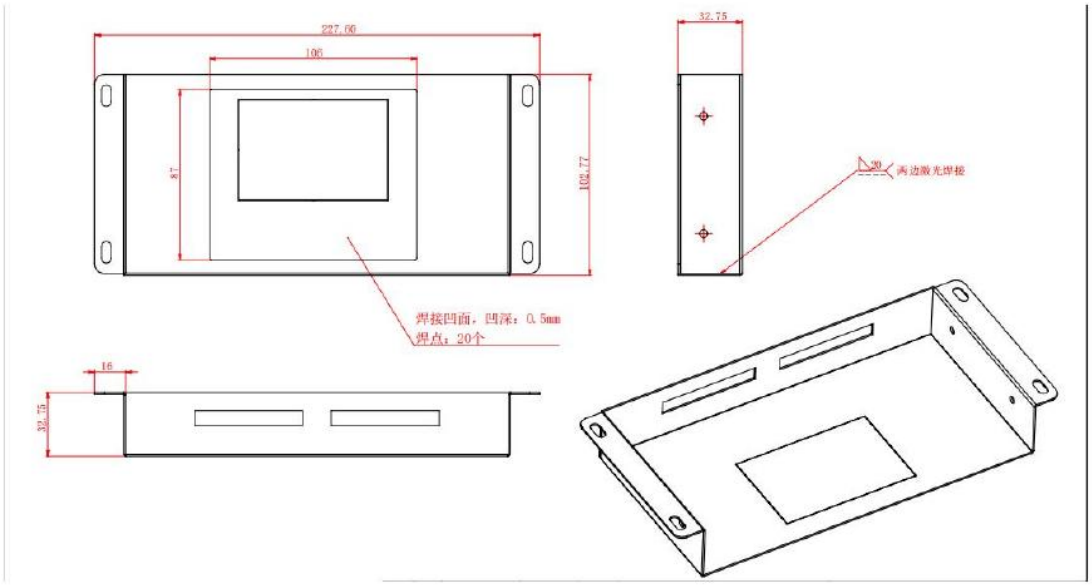
Xiamen Nengjia New Energy Technology Co., Ltd.
www.xmnengjia.com

1. Product Overview

The NJ-iot210 gateway from Nengjia is a high-performance industrial data processing device designed for a wide range of application scenarios. It integrates the core functions of a traditional telemetry terminal, supports multiple communication modes, provides rich peripheral interfaces and industrial-grade protection, and enables encrypted data transmission and multi-platform integration. It can be widely used in water conservancy, environmental protection, power, municipal infrastructure, and similar fields to realize data acquisition, storage, control, alarms, and remote transmission.

- Multi-mode communication: supports 4G, LoRa, Wi-Fi, and Ethernet (RJ45) access to match different deployment scenarios.
- Rich interfaces: provides RS485, RS232, DI/DO, and relay outputs for diverse peripheral connectivity.
- Industrial-grade design: supports 7 V to 50 V wide-range input and includes built-in ESD and reverse-polarity protection for harsh environments.
- Data security: supports TLS/SSL encrypted transport to prevent eavesdropping or tampering.
- Multi-platform integration: can connect to third-party IoT platforms such as Aliyun, HUAWEI CLOUD, Baidu TianGong, and China Mobile OneNET.

2. Product Images and Dimensions



二、技术参数

Overall gateway dimensions: 255 x 131 x 32 mm. The source manual states that these dimensions were measured manually on a flat sample and may vary by 1 to 2 mm depending on measurement method; the delivered product should prevail.

3. Technical Specifications

Item	Specification
Communication interfaces	3 x RS485; 1 x RS232; 2 x general-purpose inputs (DI, 3.3 V); 2 x general-purpose outputs (DO, 3.3 V, relay-controllable); 2 x relay outputs (COM, NO, NC); 2 x 12 V power outputs for peripherals; TF storage card with optional expansion storage.
Network connectivity	4G with Cat.1 support and onboard SIM option; Wi-Fi/LoRa for wireless networking and commissioning; Ethernet RJ45 for wired access.
Power characteristics	Operating voltage: 7 V to 50 V wide-range input.
Protection design	15 kV ESD protection; RS485 electrical isolation; reverse-polarity and over-voltage protection.
Physical characteristics	Operating temperature: -40 C to +85 C (industrial-grade standard).
Protocol support	TCP, UDP, MQTT, HTTP, and other standard IoT communication protocols.
Platform integration	Supports third-party cloud platforms including Aliyun, HUAWEI CLOUD, Baidu TianGong, and China Mobile OneNET.
Data security	TLS/SSL encrypted transport to prevent data interception or tampering.

4. Wiring Instructions

The bottom terminal block is ordered from left to right as follows.

No.	Name	Description
1	485_A1	RS485 channel 1 A line
2	485_B1	RS485 channel 1 B line
3	485_A2	RS485 channel 2 A line
4	485_B2	RS485 channel 2 B line
5	485_A3	RS485 channel 3 A line
6	485_B3	RS485 channel 3 B line
7	232_tx1	RS232 channel 1 TX
8	232_rx1	RS232 channel 1 RX
9	gnd	Ground
10	DI_1	Pulse input pin 1
11	gnd	Ground
12	DI_2	IO input pin 1
13	DI_3	IO input pin 2
14	gnd	Ground
15	NO1	Relay 1 output; open circuit when power is off
16	CON1	Relay 1 common input
17	NC1	Relay 1 output; closed when power is off
18	NO2	Relay 2 output; open circuit when power is off
19	CON2	Relay 2 common input
20	NC2	Relay 2 output; closed when power is off
21	DO_1	IO level output 1 for logic-level signaling
22	DO_2	IO level output 2 for logic-level signaling
23	gnd	Ground
24	12V_output_1	12 V output 1
25	gnd	Ground
26	12V_output_2	12 V output 2

27	gnd	Ground
28	VDD_IN	12 V input

5. Interface Definitions

For easier identification in the protocol, each interface type is assigned a dedicated ID. The ID definitions are shared across the IOT-210 and IOT200 series devices.

Serial Port IDs

Serial Port	ID
485_1	01
485_2	02
485_3	03
232_1	04
TTL_1	05

DI Input IDs

DI Input	ID	Wiring Position
DI_1	01	10
DI_2	02	12
DI_3	03	13

DO Output IDs

There are six DO outputs in total: two logic-level outputs, two custom-voltage relay switches whose output voltage follows the input voltage, and two 12 V relay-switched outputs.

DO Output	ID	Wiring Position	Type
DO_1	01	21	IO level output
DO_2	02	22	IO level output
DO_3	03	15,16,17	Custom-voltage relay switch
DO_4	04	18,19,20	Custom-voltage relay switch
DO_5	05	23,24	12 V output relay switch
DO_6	06	25,26	12 V output relay switch

6. Communication Protocol

6.1 Basic Frame Format

Field	Value
Packet header	4E4A
Command word	Variable by command
Message body length	N
Message body	xxxxxxxx
Packet tail	5852

6.2 IO Configuration Protocol

6.2.1 Read IO Input Status

Field	Value
Command word	01
Message body	Empty
Request example	4e4a01005852
Response example	4e4a01060100020103015852

IO ID	Level	IO ID	Level	IO ID	Level
01	00	02	01	03	01

When an IO input is left unconnected, the default level is high. An external low-level signal pulls it low, and an external high-level signal pulls it high. IO state changes are reported immediately, and the current IO state is also reported once every 30 minutes.

6.2.2 Set IO Output Parameters

Offset	Field	Length (bytes)	Description
0	Control mode	1	0 = scheduled mode; 1 = manual mode. Manual mode does not use scheduling parameters.
1	Number of time slots	1	Number of time-slot records to issue.
2-N	Time-slot data packet	Variable	Contains the scheduled time slots and the IO outputs to be controlled.
Offset	**Field**	**Length (bytes)**	**Description**
0	Hour	1	For example, 0x10 means 16:00.
1	Minute	1	For example, 0x32 means 50 minutes, i.e. 16:50.
2	Count	1	Number of IO outputs to control.

3	ID	1	ID of the IO output to control.
4	Action	1	Action for that IO output: 0 = Off, 1 = On.
5	ID	1	Next IO output ID, repeated as needed.
6	Action	1	Next IO output action, repeated as needed.

For logic-level output IO, 0 means low level and 1 means high level (3.3 V). For relay-control IO, 0 means relay off and 1 means relay on.

6.2.3 Read IO Output Parameters

Field	Value
Command word	03
Message body	Empty
Request example	4e4a03005852
Response format	4e4a03xxxxxxxx5852
Payload note	The xxxxxxxx payload uses the same content definition as the Set IO Output Parameters command.

6.2.4 Remote Relay Control

Offset	Field	Length (bytes)	Description
0	Count	1	Number of IO outputs to control.
1	ID	1	ID of the IO output to control.
2	Action	1	0 = Off, 1 = On.
3	ID	1	Next IO output ID, repeated as needed.
4	Action	1	Next IO output action, repeated as needed.

Example: 4e4a0407030101020103015852. In scheduled mode, the commanded state is maintained for one hour and then the device returns to scheduled operation. In manual mode, the post-command state is retained until it is changed again.

6.2.5 Read and Report IO Output Status

Field	Value
Command word	05
Message body	Empty
Request example	4e4a05005852
Response example	4e4a0511190904103216010102010301040105010601 5852

Offset	Field	Length (bytes)	Description
0	Year	1	Device clock year; for example 0x19 + 2000

			= 2025.
1	Month	1	For example, 0x09 = September.
2	Day	1	For example, 0x04 = day 4.
3	Hour	1	For example, 0x10 = 16:00.
4	Minute	1	For example, 0x32 = 50 minutes.
5	Second	1	For example, 0x16 = 22 seconds.
6/7	Output 1 ID / state	2	State: 0 = Off, 1 = On.
8/9	Output 2 ID / state	2	State: 0 = Off, 1 = On.
10/11	Output 3 ID / state	2	State: 0 = Off, 1 = On.
12/13	Output 4 ID / state	2	State: 0 = Off, 1 = On.
14/15	Output 5 ID / state	2	State: 0 = Off, 1 = On.
16/17	Output 6 ID / state	2	State: 0 = Off, 1 = On.

By default, the current IO output status is reported every 30 minutes. Any IO output change is also reported immediately.

6.3 Controller Data Acquisition Protocol

6.3.1 Device Uplink of Controller Data

On the uplink channel, controller data reported by the device is sent in transparent passthrough format and matches the collected data exactly.

6.3.2 Platform Query Format for Controller Data

Transparent mode: when configuring the uplink server, serial ports can be bound to a server link. Commands sent through that link are transparently forwarded to all serial ports bound to it.

Private protocol: the data format is the same as the basic IO configuration frame format.

Field	**Value**
Command word	06
Example	4e4a060a010801040200002271ab5852
Meaning	The device sends command 01040200002271ab to serial port 1.

Offset	**Field**	**Length (bytes)**	**Description**
0	Serial port ID	1	Target serial port to receive the command.
1	Command length	1	Length of the command to be sent.
2-N	Command	Variable	Command payload to send to the target serial port.

7. Configuration Software Instructions

7.1 Connection Instructions

The device enables serial port 485_1 by default, uses a baud rate of 9600, and uses no parity. For first-time configuration, physically connect to serial port 485_1 and read the device parameters once before editing so they are synchronized into the software.

7.2 Network Configuration

- The device provides three backhaul methods: 4G, Ethernet, and Wi-Fi.
- The default backhaul is 4G. The device ships with an onboard 4G SIM solution; if a customer needs to use a self-supplied SIM card, the source manual says this must be communicated in advance.
- Ethernet backhaul uses the WAN port and currently supports DHCP only. Using the LAN port to provide network access to external devices is not yet open.
- Wi-Fi networking currently supports STA mode and requires the network name and password. AP mode is not yet open.

7.3 Uplink Network Configuration

- The device currently supports TCP and MQTT uplink methods and can maintain up to two uplink connections.
- In MQTT mode, if the client ID is left blank, the device IMEI is used as the client ID during connection.
- Serial-port binding: when a link is bound to one or more serial ports, data collected from those serial ports is reported to that server. If the server sends transparent commands, they are forwarded to all bound serial ports. If it sends private-protocol commands, they are forwarded only to the specified serial port.
- A single server can bind multiple serial ports, and different servers can bind the same serial port without conflict.

7.4 Serial Port Configuration

- Serial port selection: choose the serial port number that matches the device to be connected. Chapter 4 provides the detailed mapping.
- Serial packet splitting: the default is timeout-based splitting and the source manual recommends using it first. Timeout-based splitting treats a packet as complete when no data is received for a set period; the recommended value is 200 ms.
- Delimiter mode: a packet is considered complete when the configured delimiter is detected.
- Fixed-length mode: data longer than the configured packet length is reported in separate packets.
- Polling configuration: set the polling interval, choose HEX or ASCII data format, and configure one or more polling commands. The device reads once about 30 seconds after boot and then follows the configured polling interval.
- Return configuration: passthrough is the default. Data can optionally be prefixed or suffixed to simplify server-side parsing.
- Append start address: for Modbus devices, the start address can be prepended to the uploaded payload.
- Protocol header and protocol tail: these strings are added to the beginning and end of the uploaded data respectively.

Example from the source manual: if the polling command is 010300000002C40B, the append-start-address option is enabled, the protocol header is aa, and the protocol tail is bb, and the returned polling data is 010304FFDD01005B8D, then the data reported to the server becomes aa0000010304FFDD01005B8Dbb.

7.5 Parameter Setting and Batch Configuration

- After all parameters have been configured, click Set Parameters. The log window returns the string `seccess` on success and `fail` on failure. After a successful write, click Restart for the settings to take effect.
- Batch configuration: exporting parameters generates a JSON file. When connecting another device, import that JSON file to synchronize the saved configuration to the new device, then click Set Parameters.