AgroSense_4 Channel Relay

LoRaWAN® Manual

V1.0

Author: Yuki

Time: 2025.03.28

Contents

1 Product Description 1

1.1 Introduction 1

1.2 Feature 1

1.3 Parameter 2

2 Technical route 4

2.1 System Framework 4

2.2 Regional frequency band 5

3 Usage 6

3.1 Interface Specification 6

3.1.1 DC Power Input 6

3.1.2 Dual-Channel 0-5V Analog Input 6

3.1.3 Air Temperature & Humidity Detection 7

3.1.4 Four-Channel Relay Control 7

3.2 Usage with TTN &ThingSpeak 7

3.2.1 Network Server configuration 8

3.2.2 Decoder 10

3.2.3 Application Server configuration 14

3.2.4 Connect the Network Server and Application Server 14

3.2.5 Downlink 16

3.3 Usage with Datacake 19

3.3.1 Downlink 27

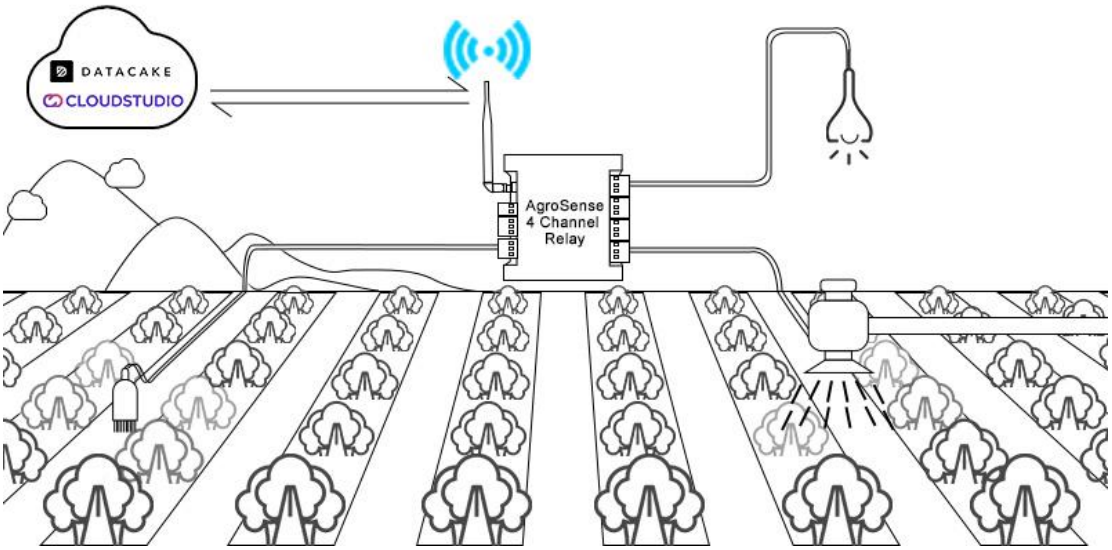
1 Product Description

1.1 Introduction

AgroSense 4-Channel LoRaWAN® Relay Controller is a high-performance IoT device designed for remote control and environmental monitoring applications. It integrates 4 high-load relay outputs, 2 analog input channels (0–5V), and a high-precision temperature and humidity sensor. Supporting long-range wireless communication via the LoRaWAN® protocol (up to 5 km), it is ideal for smart agriculture, industrial automation, and intelligent building scenarios.

The device complies with the LoRaWAN® V1.0.3 OTAA Class C standard and offers high sensitivity (-137 dBm) and strong anti-interference capability. It supports global frequency bands (EU868/US915) and works seamlessly with platforms such as TTN, Datacake, and CloudStutio to enable data upload and remote control.

Users can send downlink commands to control relays, adjust data reporting intervals, and perform scheduled tasks—making it highly adaptable to various automation needs. Local storage for 3,300+ records ensures critical data is retained during connectivity gaps.



1.2 Feature

- Includes 4-channel independently controlled **Relays.**
- Includes 2-channel 0-5V analog **signal input**.
- Integrates a high-accuracy **temperature and humidity sensor.**
- LoRaWAN version: LoRaWAN Specification 1.0.3. OTAA **Class C.**
- Monitor data and upload **real-time** data regularly.

- Modify the product parameters through **AT commands.**
- Support **downlink** to modify the time interval and control relay switch.
- Integrated data logging capability with a storage capacity of up to 3300 records.
- Compatible with Worldwide **LoRaWAN® Networks: Support** the universal frequency bands EU868/ US915.
- **Long Range:** Up to 2 kilometers in the city, up to 10 kilometers in the wilderness, receive sensitivity -137dBm , transmit power up to 22dBm.
- **Data encryption:** Provide end-to-end secure communication, including device authentication and network data encryption, to ensure the security of data transmission and prevent data theft and malicious attacks.
- **High stability and reliability:** good stability in noisy environments, able to penetrate buildings and obstacles, so it can maintain good communication quality in urban and suburban environments.

1.3 Parameter

1. General Parameters

Product Model	AGLW4R01
Temperature Measurement Range	-40°C ~80°C
Temperature Measurement Accuracy	±0.5°C
Temperature Resolution	0.1°C
Humility Measurement Range	0%-100% RH
Humility Measurement Accuracy	±2%
Humility Resolution	0.024% RH
ADC Measurement Range	12-bit ADC
ADC Measurement Accuracy	1/4096
ADC channel count	2
Relay Power Supply	DC 24V
Relay Contact Rating	20A/125V
Relay channel count	4

2.Wireless Parameters

Communication Protocol	Standard LoRaWAN® protocol V1.0.3
Network Access/Operating Mode	OTAA **Class C**
MAX Transmit Power	22dBm
Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

3.Physical Parameters

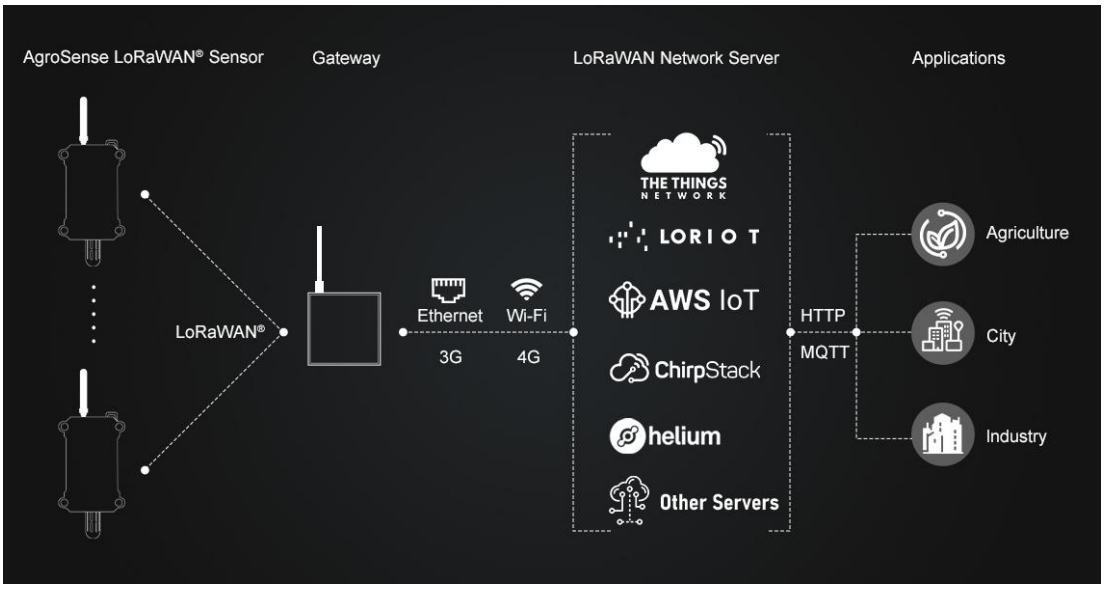
Power Supply	DC 24V
Operating Temperature	-40°C ~85°C
Dimensions	95 × 90 × 40 mm
Mounting	Wall Mounting

2 Technical route

2.1 System Framework

AgroSense 4_Channel Relay uses LoRAWAN technology, and it network architecture includes four parts: End Nodes, Concentrator/Gateway, Network Server and Application Server.

End Nodes	It is responsible for collecting sensing data and then transmitting it to Gateway via the LoRaMAC protocol.
Concentrator/Gateway	It is mainly responsible for transmitting node data to the server.
Network Server	Organize the data into JSON packets and decode them.
Application Server	Display the data.



Uplink:

1.Data Collection & Transmission

Sensor data and transmits it to the Gateway via LoRaWAN® protocol.

2.Gateway Forwarding

The Gateway packages the raw data and forwards to the Network Server.

3.Data Decoding & Routing

The Network Server decodes the payload and forwards it to the designated Application Server.

4.User Monitoring

The Application Server processes the data and updates the user interface (APP), allowing real-time monitoring of data.

Downlink:

1.Command Generation

A downlink commands generated in the Network Server or Application Server through a predefined API/interface. (Example: Set sampling interval to 10 minutes; Control Relay ON/OFF.)

2.Gateway Transmission

The command is encapsulated into a downlink packet and sent to the Gateway via the network.

3.End Node Execution

The Gateway transmits the downlink command to the target End Node using the wireless protocol. The End Node parses the command and performs the corresponding action (e.g., activate relay, modify configuration).

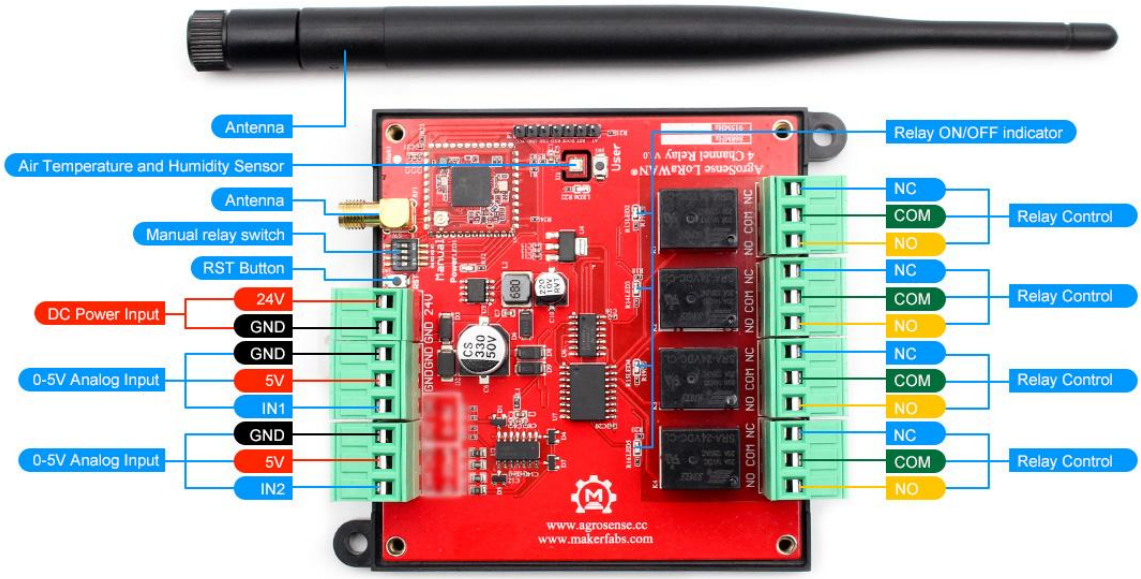
2.2 Regional frequency band

At the present moment, our product solely accommodates compatibility with the US915 and EU868.

area	frequency band	center frequency
China	470-510MHz	CN486MHz
America	902-928MHz	US915MHz
Europe	863-870MHz	EU868MHz
Korea	920-923MHz	KR922MHz
Australia	915-928MHz	AU923MHz
New Zealand	921-928MHz	NZ922MHz
Asia	920-923MHz	AS923MHz

3 Usage

3.1 Interface Specification



3.1.1 DC Power Input

AgroSense_4 Channel Relay operates from an external DC power source.

Power Specifications	DC 24V ±10%
Terminal Definitions	24V: Positive terminal (power input)
	GND: Negative terminal (ground)

3.1.2 Dual-Channel 0-5V Analog Input

AgroSense_4 Channel Relay provides 2-channel analog sensor interfaces (0-5V), enabling flexible deployment of analog signal-based sensors.(e.g.pressure, light intensity)

Sensor Power Output	5V ± 0.2V Max load current: 500mA
Terminal Definitions	GND: Negative terminal (ground)
	5V: Positive terminal (power output)
	IN1/IN2: (Signal Input)

3.1.3 Air Temperature & Humidity Detection

AgroSense 4_Channel Relay integrates a high-precision temperature and humidity sensor AHT20, capable of monitoring and feedback of environmental parameters to provide data support for system regulation.

- Temperature Sensor: -40°C to +85°C
- Relative Humidity Sensor: 0% to 100% RH

3.1.4 Four-Channel Relay Control

AgroSense 4_Channel Relay is equipped with 4 independent relay outputs, supporting high-load switching control. Users can flexibly control various electrical devices through the relay on/off status to achieve automated management.

Smart Control Modes	**Independent Channel Control:** Each relay supports individual programming and real-time status monitoring
	Auto Mode: Receives Downlink commands from cloud platform and Scheduled automation tasks
	Manual Mode: Toggle Manual relay switch for local control
Contact Type:SPDT	**COM** : Connects to load power
	NO : Closed when relay energized
	NC : Open when relay energized
High-Capacity Switching	Single channel rating : 20A/125AC or 20A/14VDC

Safety Warnings

⚠ For inductive loads (e.g., motors, solenoids):Must install RC snubber circuit.
⚠ DO NOT exceed rated specifications – Contact arcing may cause fire.

3.2 Usage with TTN &ThingSpeak

In the phase, We use The Things Network(TTN) as data server, and Thingspeak as console to display data& control the relays.

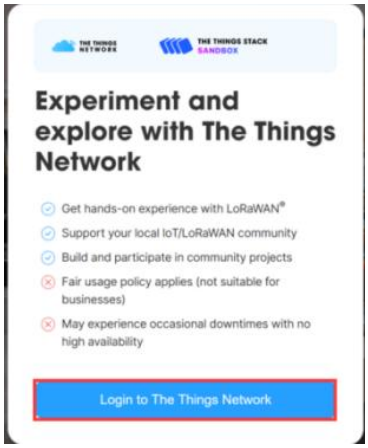
we need to configuration the country/area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

DEV EUI	Unique identification of device, authorized by IEEE
APP EUI	Unique identification of application
APP Key	One of the join network parameters on OTAA mode, calculated by DE EUI

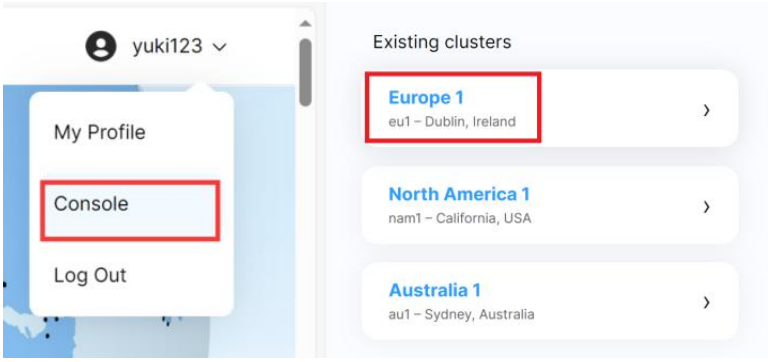
- End Nodes and Gateway: AgroSense_4 Channel Relay.(The AgroSense series is applicable)
- Network Server: The Things Network. (Datacake, Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

3.2.1 Network Server configuration

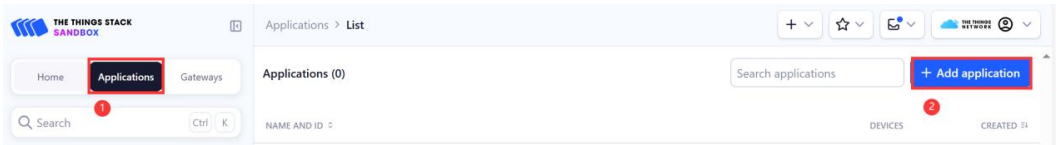
- Open The Things Network in your browser and login it. (Or register an account)



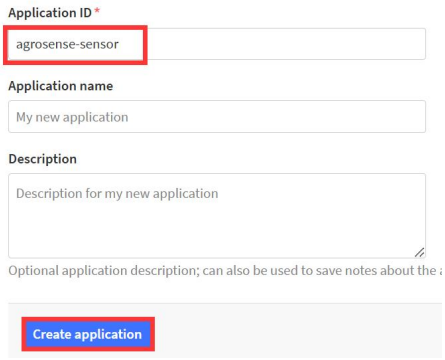
- Click “Console” and select clusters. (we take the European region for example.)



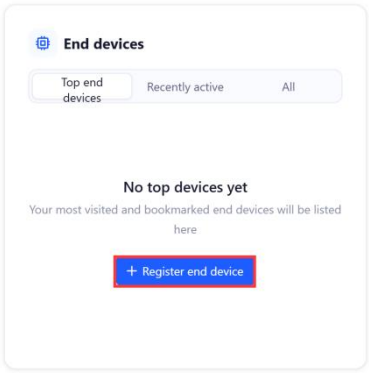
- Click “Go to applications” --> “+ Create application”.



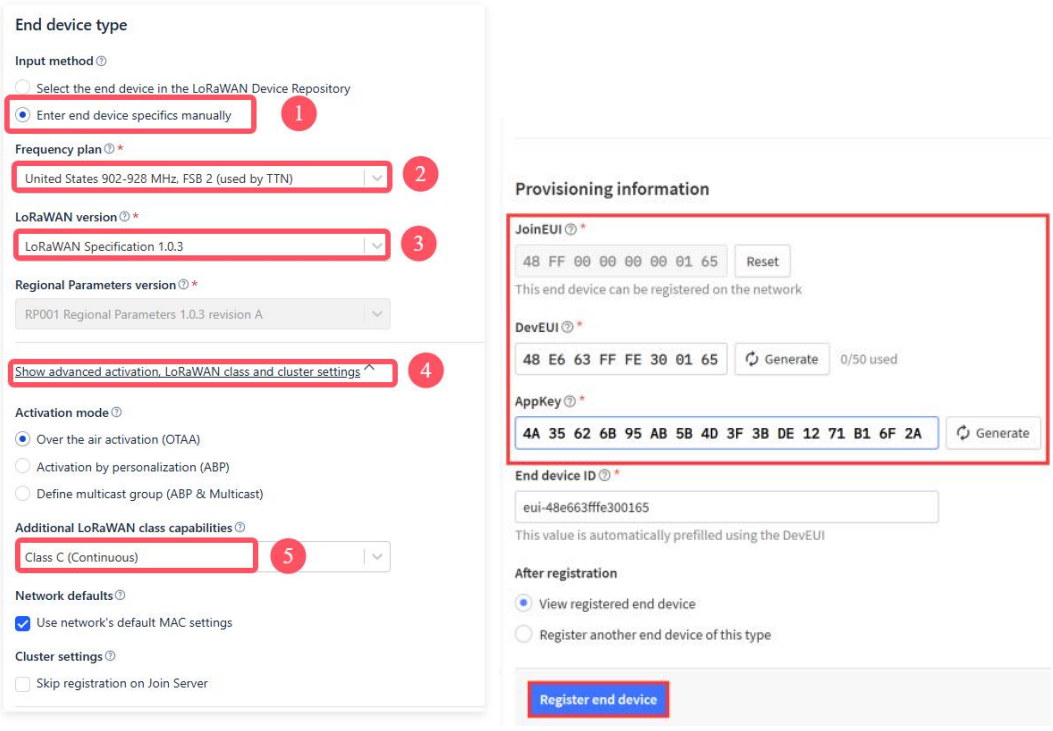
- Write the Application ID and click “Create application”.



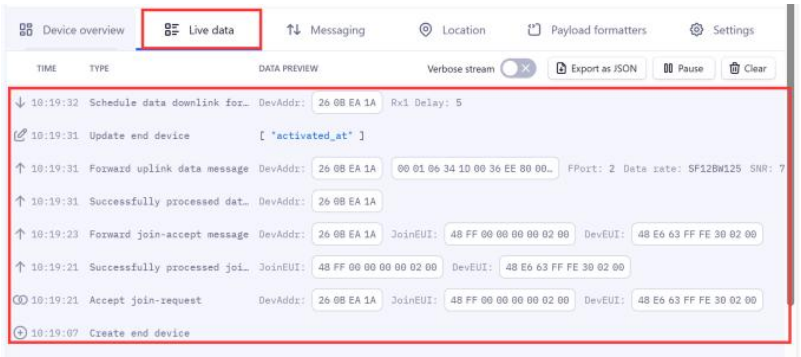
- Click “+ Register and device”.



- Fllowing the steps, **select class C** and input the DEV EUI/ APP EUI/ APP Key (notice: JoinEUI=APP EUI) and subsequently click on "Register end device" to complete the registration process.

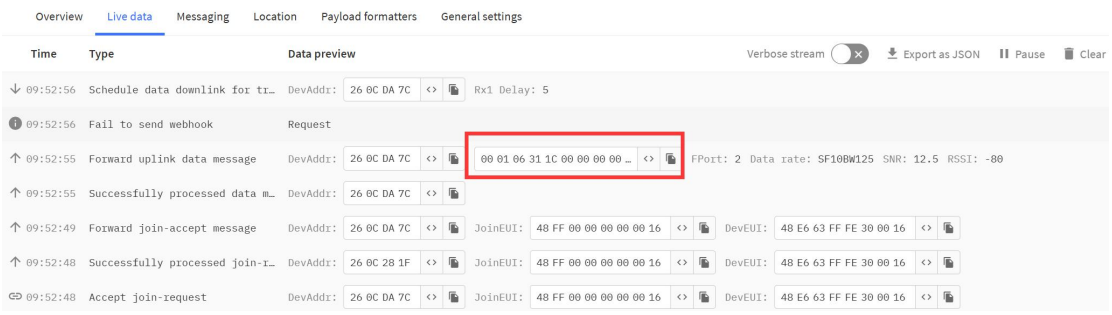


- Connects to 24v DC power supply and press RES button, you can see the device is connected successfully in the TTN.



3.2.2 Decoder

- Now, we need to decoder the data.



Data length	Data description	Value range	Explanation
byte 0	Data packet sequence number high 8 bits	0-0xFFFF	Counting starts from 0 and increments, resetting back to 0 after reaching 65535
byte 1	Data packet sequence number low 8 bits		
byte 2	Relay Status_K1	0/1	0 is invalid, 1 is valid.
byte 3	Relay Status_K2		
byte 4	Relay Status_K3		
byte 5	Relay Status_K4		
byte 6	Humility sensor bits 8 to 15		This value is obtained after magnifying the data by 10 times. To obtain the actual relative humidity value, the real value needs to be calculated by dividing it by 10. For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the relative humidity value obtained is 0x00000285= 645. After converting and dividing by 10, the actual relative humidity is 6.45%RH.
byte 7	Humility sensor bits 0 to 7		
byte 8	Temperature sensor bits 8 to 15		This value is obtained after magnifying the data by 10 times. To obtain the actual Temperature value, the real value needs to be calculated by dividing it by 10. For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the relative humidity value obtained is 0x00000285= 645. After converting and dividing by 10, the actual Temperature is 6.45℃.
byte 9	Temperature sensor bits 0 to 7		
byte 10	ADC_1 bits 8 to 15		The value is obtained by amplifying the data by 10 times, and the actual value needs to be divided by 10 to convert to the actual voltage. The purpose of multiplying by 10 is to retain one decimal place of the voltage value. For example, if the value is 0x21 = 33, then the voltage is 3.3V.
byte 11	ADC_1 bits 0 to 7		
byte 12	ADC_2 bits 8 to 15		The value is obtained by amplifying the data by 10 times, and the actual value needs to be divided by 10 to convert to the actual voltage. The purpose of multiplying by 10 is to retain one decimal place of the voltage value. For example, if the value is 0x21 = 33, then the voltage is 3.3V.
byte 13	ADC_2 bits 0 to 7		
byte 14	NC		
byte 15	NC		
byte 16	NC		
byte 17	NC		
byte 18	data transmission interval bits 24 to 31		The time interval for data transmission has been increased by a factor of 1000. The unit is seconds.
byte 19	data transmission interval bits 16 to 23		
byte 20	data transmission interval bits 8 to 15		

byte 21	data transmission interval bits 0 to 7		
Downlink			
Fport 1	Change the data sending interval		
Fport 2	Upload the quantity of the latest local logged data		
Fport 6	Change the Relay K1 ON/OFF	0/1	0 is OFF, 1 is ON.
Fport 7	Change the Relay K2 ON/OFF		
Fport 8	Change the Relay K3 ON/OFF		
Fport 9	Change the Relay K4 ON/OFF		

Example: 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0xDE, 0X00, 0X0C, 0X00, 0X21, 0X00, 0X00, 0X00, 0X00, 0x00, 0x09, 0x27, 0xC0

Data parsing：

Humility value is 51.2%.

Temperature value is 22.2℃.

ADC1 value is 1.2v

ADC2 value is 3.3v

Relay1 status is ON.

Relay2 status is OFF.

Relay3 status is OFF.

Relay4 status is OFF.

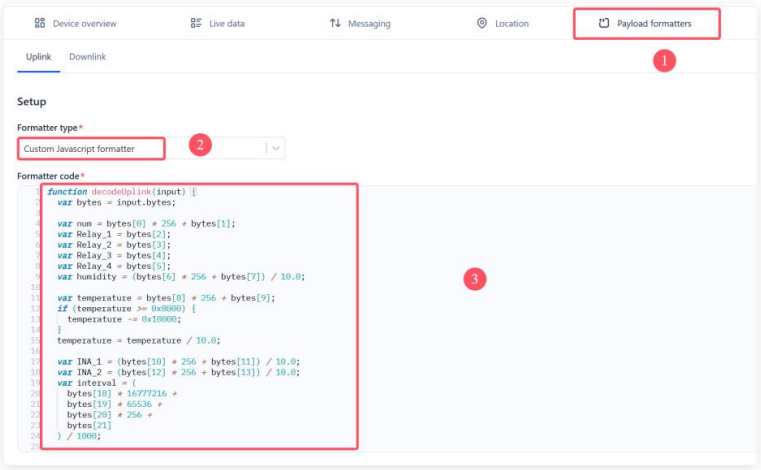
Data transmission interval value is 600s.

- Know how to decode it after, we need to write it in code. (You can check it out on Github)

```
function decodeUplink(input) {
  var bytes = input.bytes;

  var num = bytes[0] * 256 + bytes[1];
  var Relay_1 = bytes[2];
  var Relay_2 = bytes[3];
  var Relay_3 = bytes[4];
  var Relay_4 = bytes[5];
```

```
  var humidity = (bytes[6] * 256 + bytes[7]) / 10.0;
  var temperature = bytes[8] * 256 + bytes[9];
  if (temperature >= 0x8000) {
    temperature -= 0x10000;
  }
  temperature = temperature / 10.0;
  var INA_1 = (bytes[10] * 256 + bytes[11]) / 10.0;
  var INA_2 = (bytes[12] * 256 + bytes[13]) / 10.0;
  var interval = (
    bytes[18] * 16777216 +
    bytes[19] * 65536 +
    bytes[20] * 256 +
    bytes[21]
  ) / 1000;

  return {
    data: {
      Relay_1: Relay_1,//RELAY1     :0-OFF; 1-ON
      Relay_2: Relay_2,//RELAY2     :0-OFF; 1-ON
      Relay_3: Relay_3,//RELAY3     :0-OFF; 1-ON
      Relay_4: Relay_4,//RELAY4     :0-OFF; 1-ON
      INA_1: INA_1,//0-5V ADC
      INA_2: INA_2,//0-5V ADC
      temperature: temperature,
      humidity: humidity,
      interval: interval
    },
    warnings: [],
    errors: []
  };
}
```

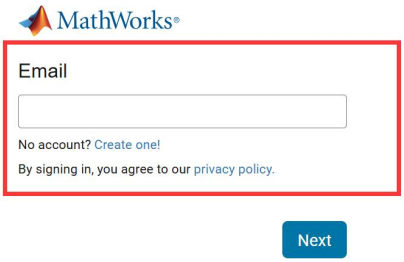
- Select “Payload formatters” and follow the steps.



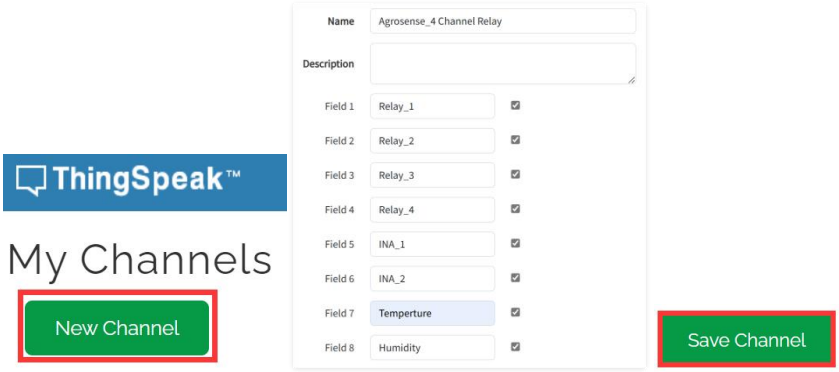
3.2.3 Application Server configuration

In the Application Server configuration, we need to create ThingSpeak channel and get Channel ID and API Key, this is the key to our connection to TTN.

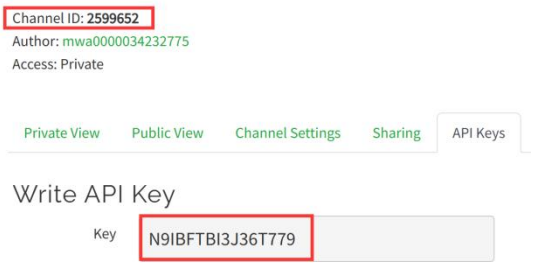
- Login to the ThingSpeak. (Or register an account)



- Click “New Channel”, fill in the Channel name and field names and click “Save Channel”.

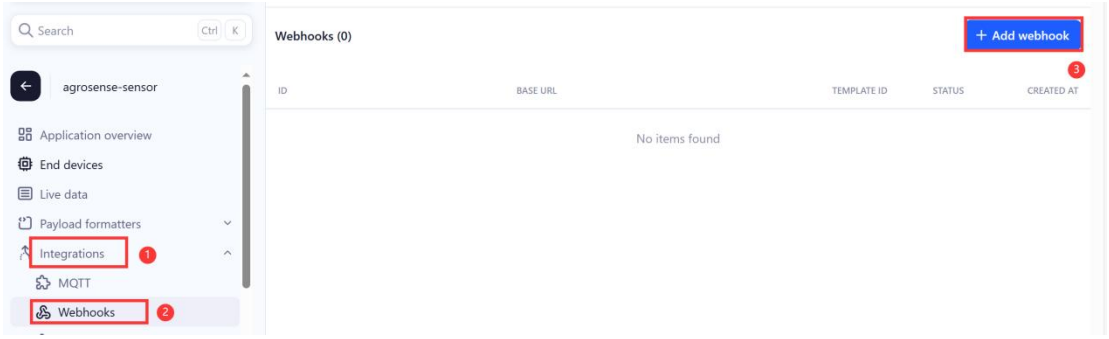


- After successful creation, copy the Channel ID and API Key.

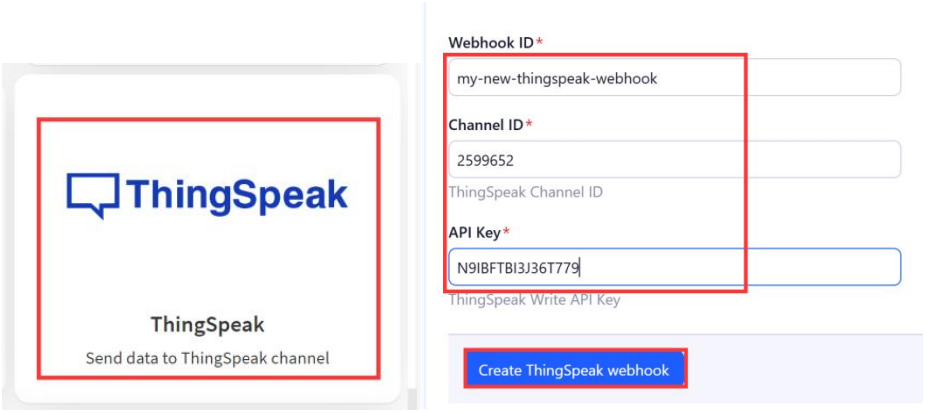


3.2.4 Connect the Network Server and Application Server

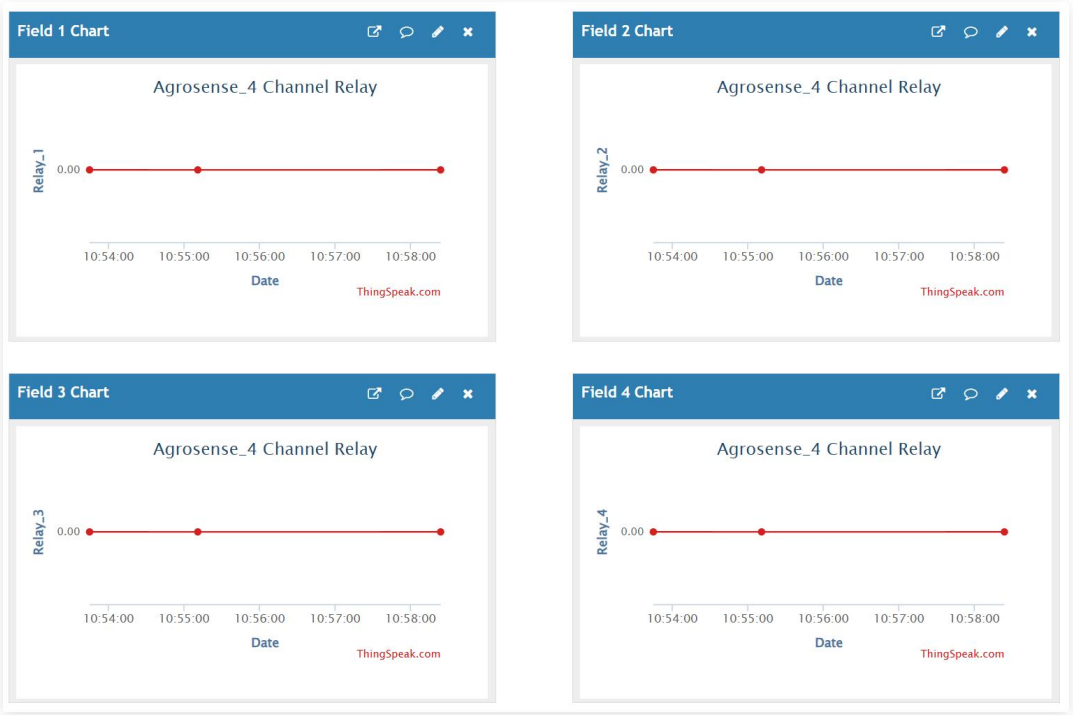
- In the TTN, click “integrations” --> “Webhooks” --> “+ Add webhook”.

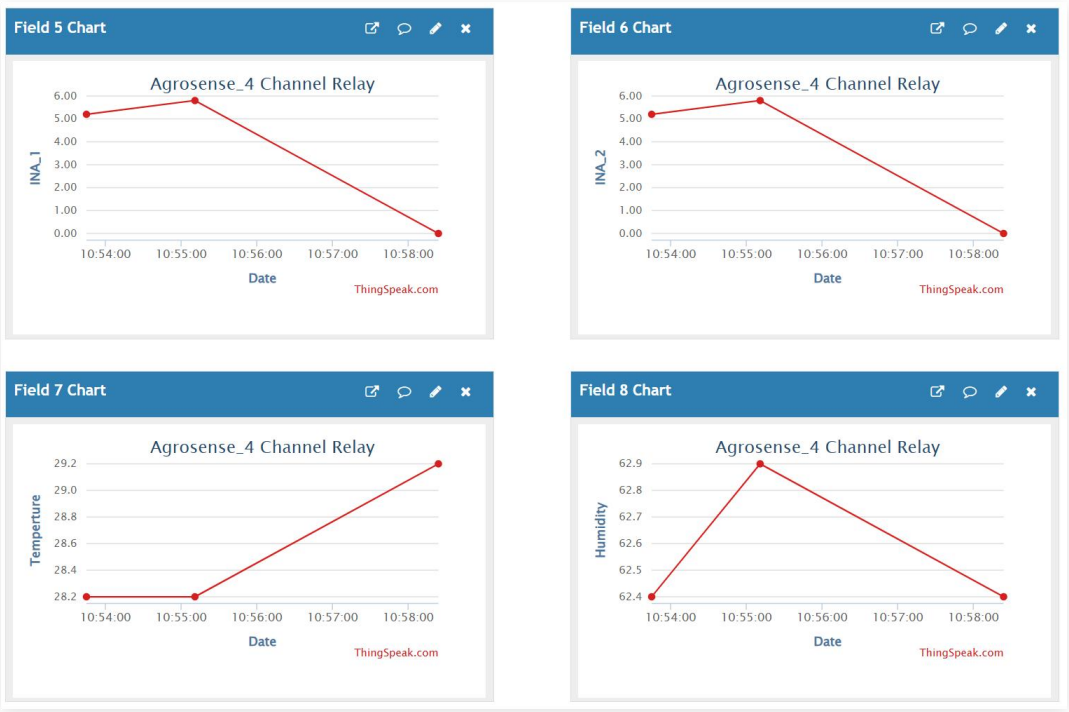


- Select “ThingSpeak”, Fill in the Webhook ID and paste the Channel ID and API Key, click “Create ThingSpeak Webhook”.



- Press RST button, wait about a minute, you will successfully see the data in ThingSpeak.(You will recive the data every hour.)





3.2.5 Downlink

The downlink has two functions:

- Modification time interva (Fport1)

Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport2)

Users can view previous data based on this feature.

- Change the Relay ON/OFF (Fport6-9)

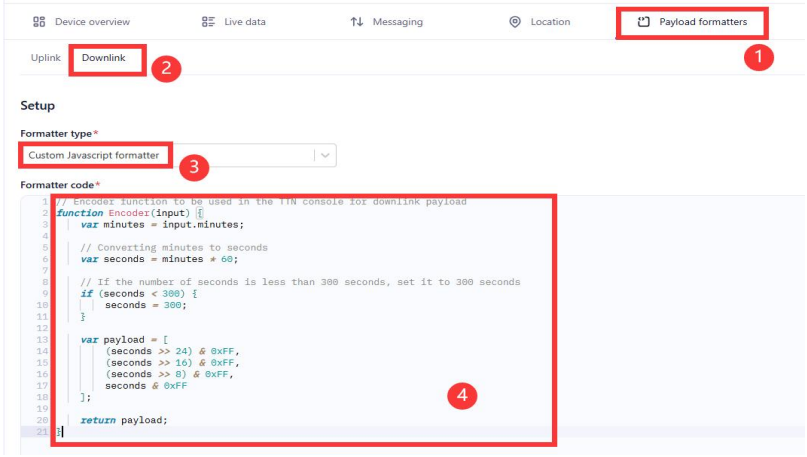
Relay Channel	Fport	Explanation
Relay K1	6	0 is OFF, 1 is ON.
Relay K2	7	
Relay K3	8	
Relay K4	9	

Modify the time interval :

1 、 If you need to change time Interval (Default 60 minutes), you can click “Payload

formatters-->Downlink" and follow the steps.

Formatter code you can find in Github.



2、Click "Save changes".

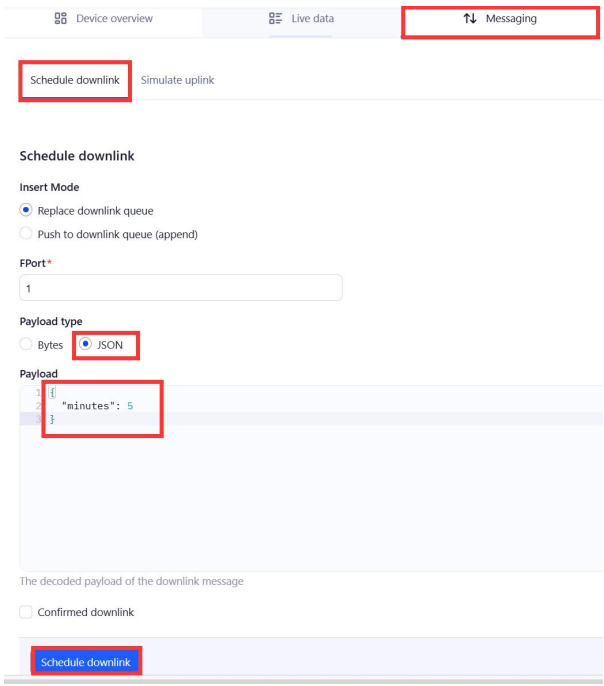
3、Click "Messaging-->Schedule downlink".

Note: you must use this format:

{

"minutes": 5

}

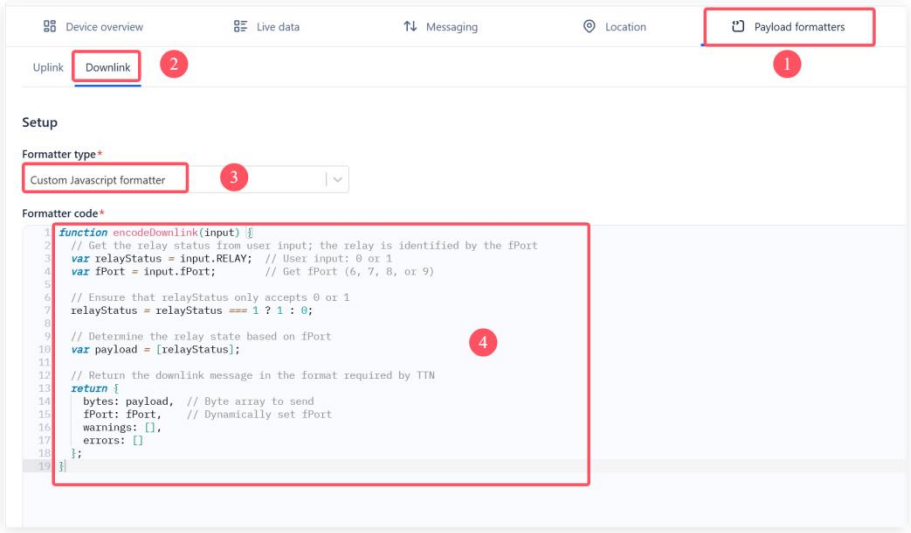


4、The modified interval will be updated after the next data upload.

Change the Relay ON/OFF:

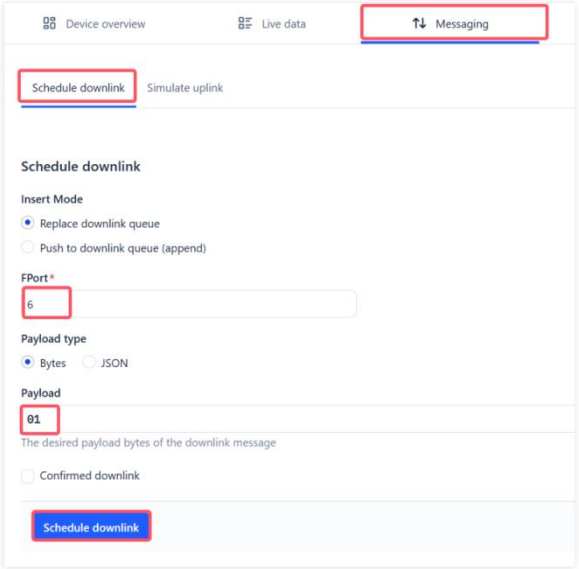
1、Click “Payload formatters-->Downlink” and follow the steps.

Formatter code you can find in Github.



2、Click “Save changes”.

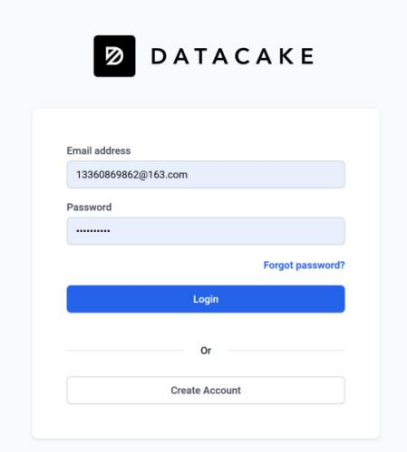
2、Click “Messaging-->Schedule downlink”. The relay will ON/OFF immediately after the modification. (00 is OFF, 01 is ON)



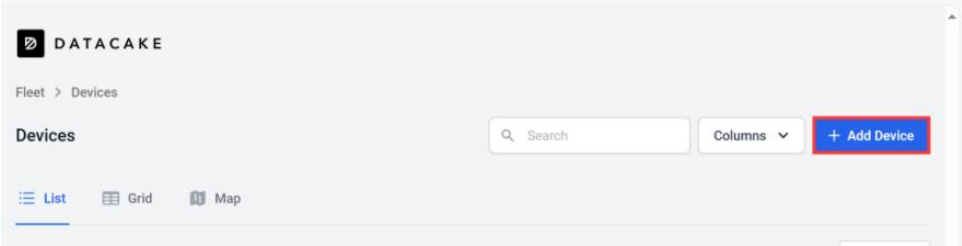
3.3 Usage with Datacake

In this phase, we use DataCake(https://datacake.co/) as the data server & console.

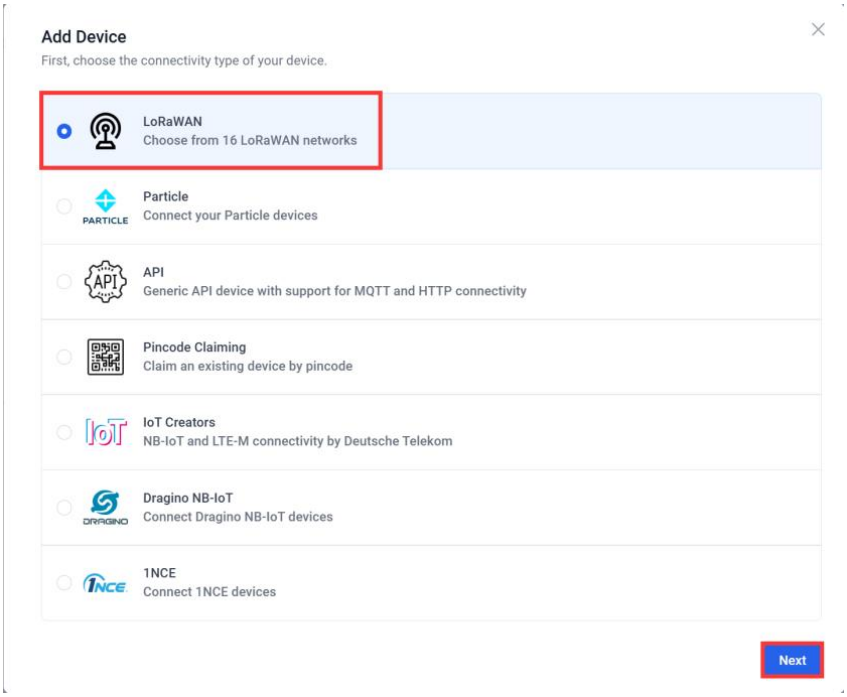
1、Login datacake or Create Account



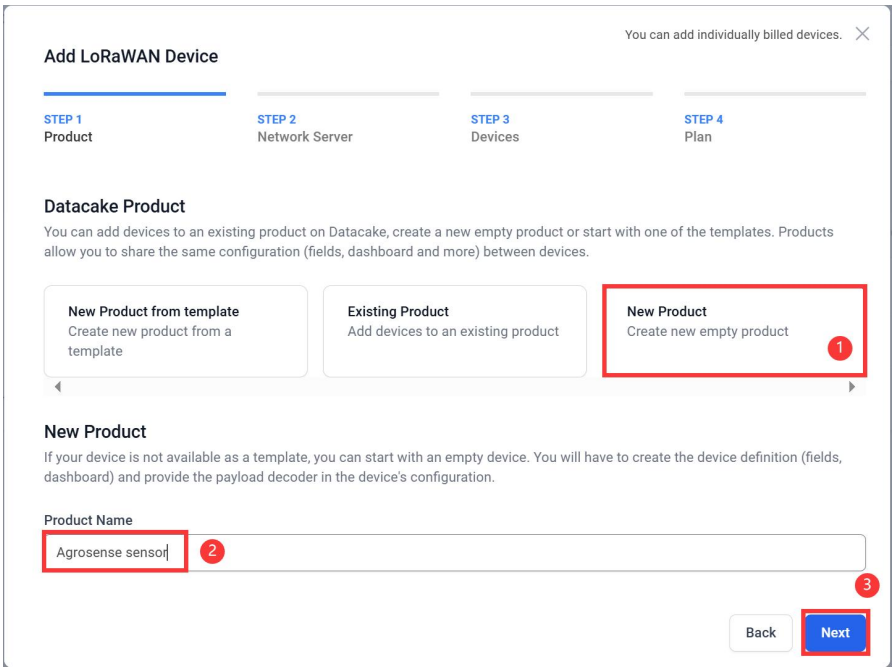
2、Click “Add Device”



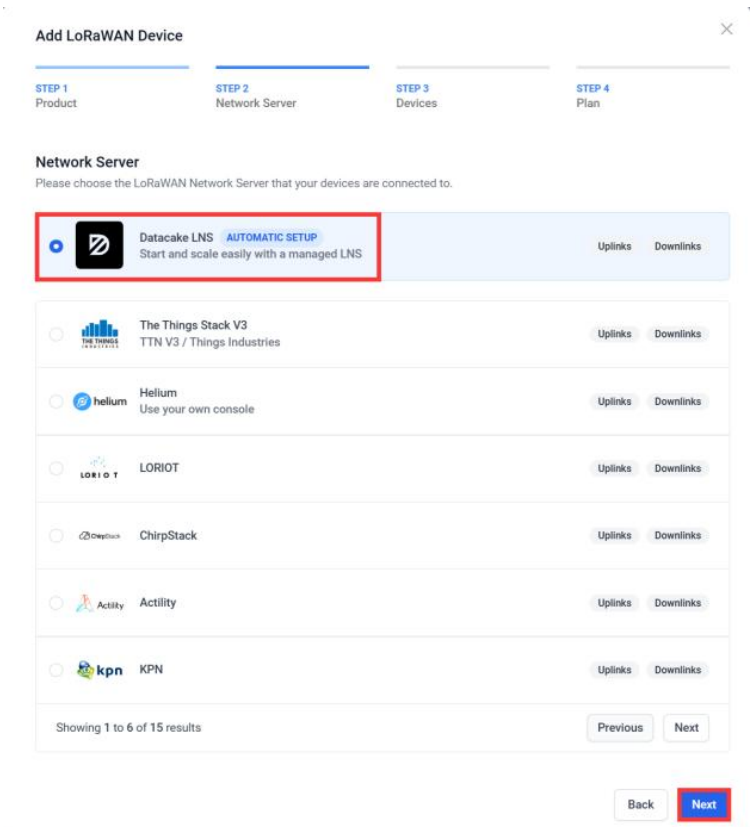
3、Select LoRaWAN and click “Next”



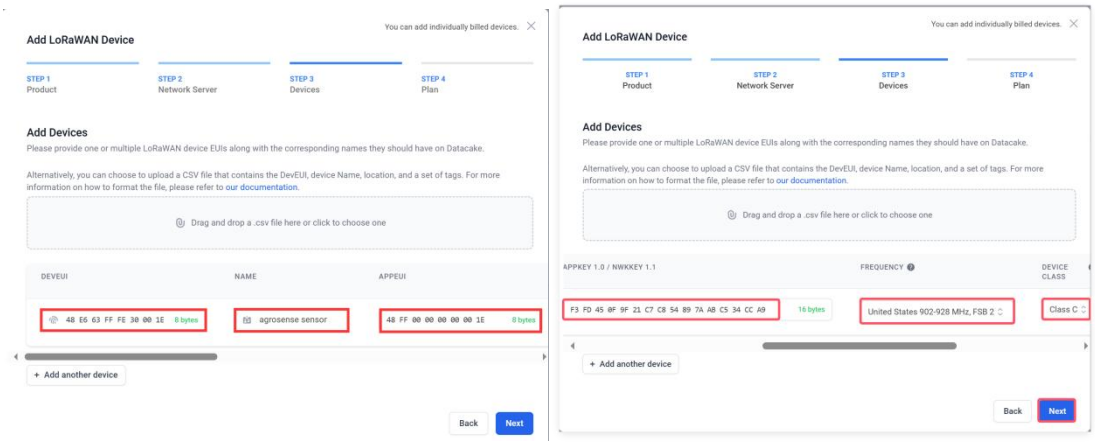
4、Select a Product based on your needs, take "Create new empty product" as an example.



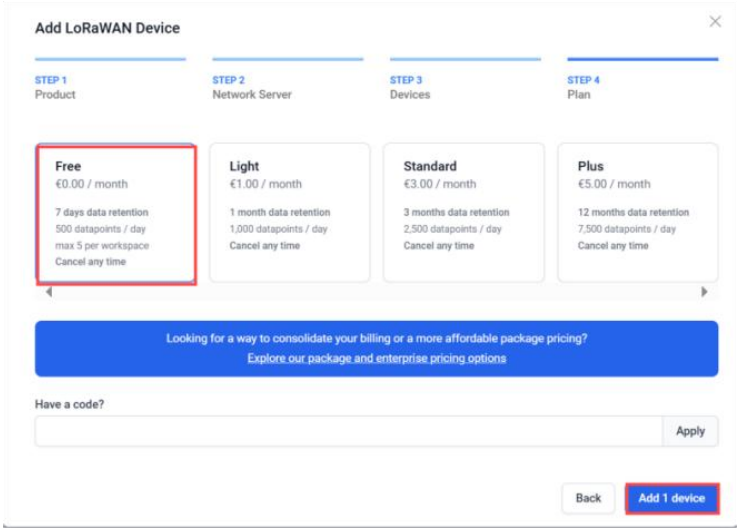
5、Select “Datacake LNS”



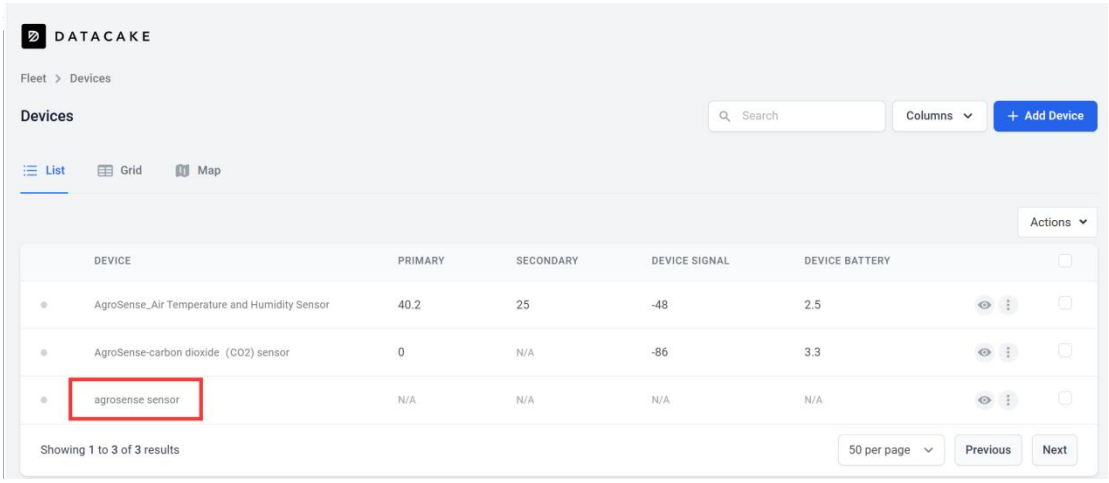
6、Enter DEVEUI、APPEUI、APPKEY、FREQUENCY(take 915 for example) and DEVICE CLASS.



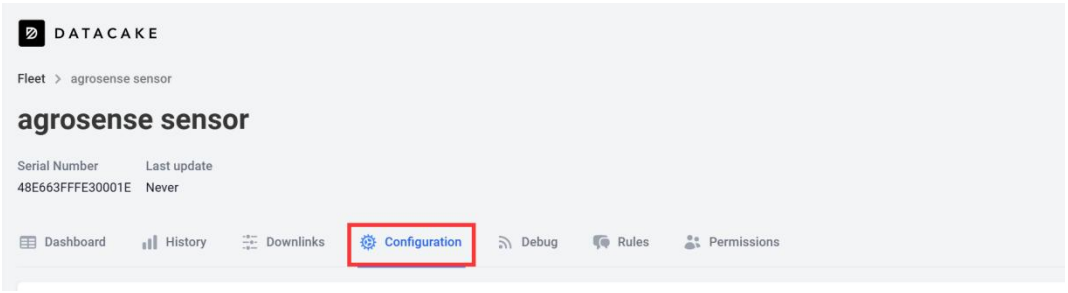
7、Choose the type according to your needs, and click “Add 1 device”.

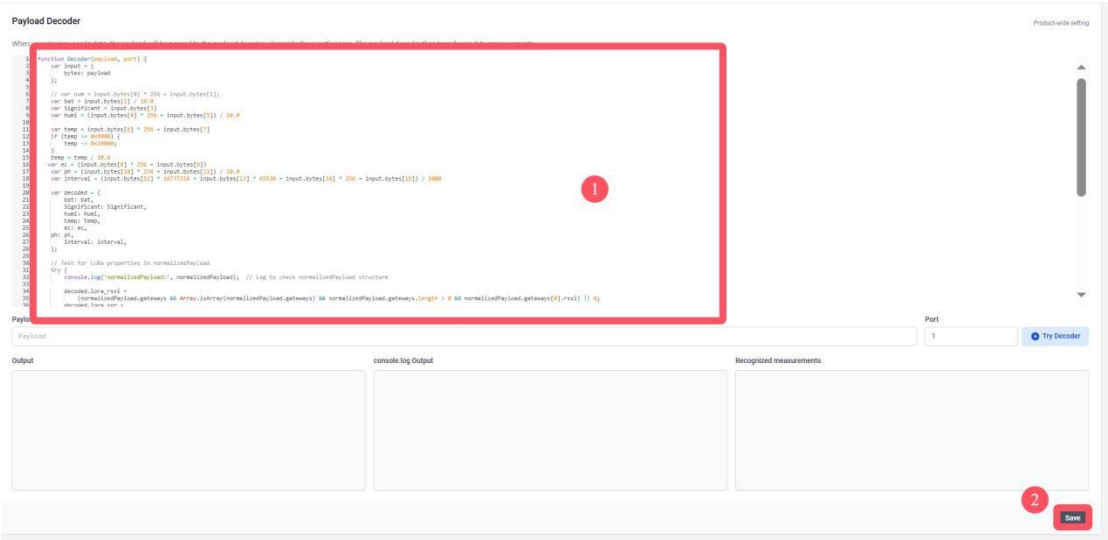


8、Click to go to the device you just added.



9、Click “Configuration”, enter Decoder and click “Save”.(You can check it out on Guihub)



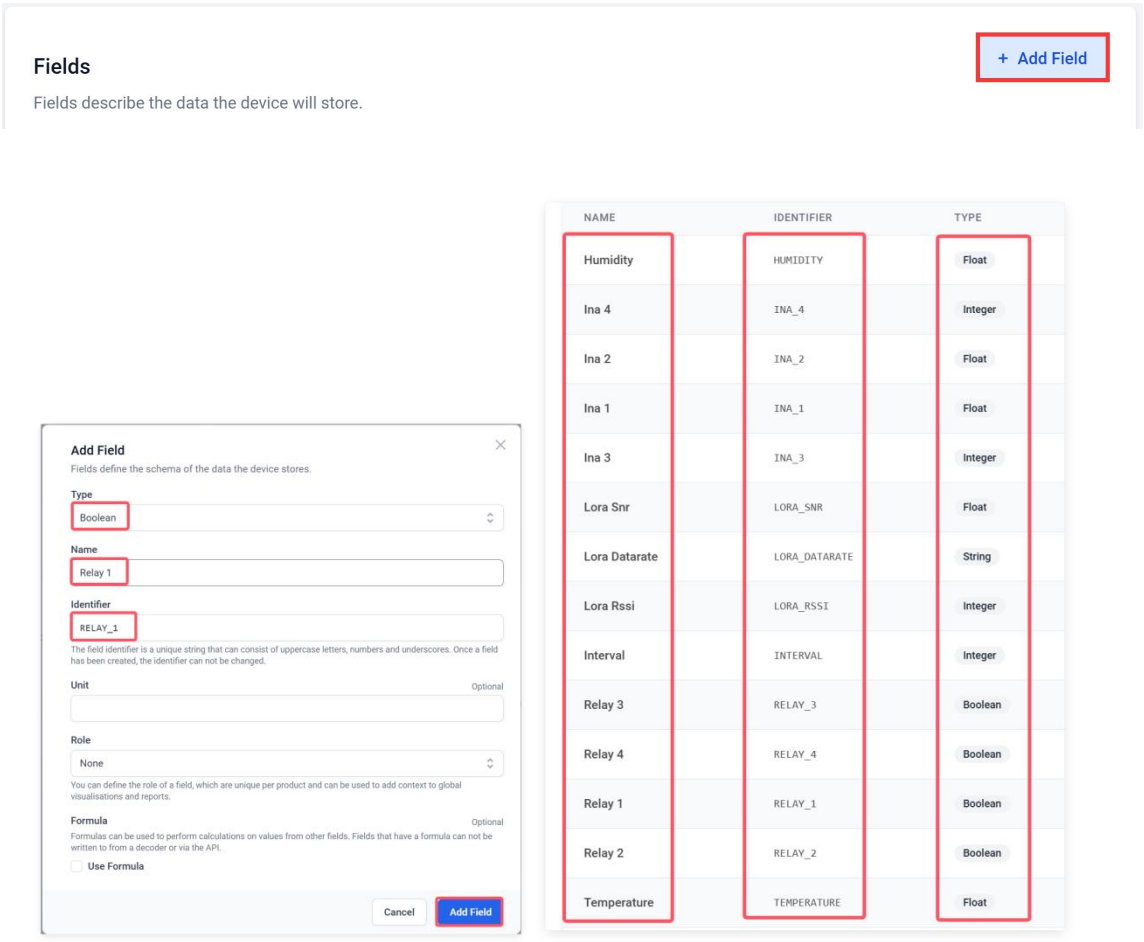



```
function Decoder(payload, port) {
    var input = {
        bytes: payload
    };

    var num = input.bytes[0] * 256 + input.bytes[1]
    var Relay_1 = input.bytes[2]
    var Relay_2 = input.bytes[3]
    var Relay_3 = input.bytes[4]
    var Relay_4 = input.bytes[5]
    var humidity = ( input.bytes[6] * 256 + input.bytes[7] ) / 10.0
    var temperature = input.bytes[8] * 256 + input.bytes[9]
    if (temperature >= 0x8000)
     {
            temperature -= 0x10000;
      }
    temperature = temperature / 10.0

    var INA_1 = ( input.bytes[10] * 256 + input.bytes[11] ) / 10.0
    var INA_2 = ( input.bytes[12] *256 + input.bytes[13] ) / 10.0
    var INA_3 = ( input.bytes[14] *256 + input.bytes[15] ) / 10.0
    var INA_4 = ( input.bytes[16] *256 + input.bytes[17] ) / 10.0
```

```
    var interval = (input.bytes[18]* 16777216 + input.bytes[19]* 65536 + input.bytes[20] * 256 + input.bytes[21])
/ 1000

    var decoded =
    {
        Relay_1:Relay_1,
        Relay_2:Relay_2,
        Relay_3:Relay_3,
        Relay_4:Relay_4,
        INA_1:INA_1,
        INA_2:INA_2,
        INA_3:INA_3,
         INA_4:INA_4,
         temperature:temperature,
         humidity:humidity,
         interval:interval,
    };

    // Test for LoRa properties in normalizedPayload
 try {
  // RSSI 和 SNR 解析
  if (normalizedPayload.gateways && normalizedPayload.gateways.length > 0) {
    decoded.lora_rssi = normalizedPayload.gateways[0].rssi || 0;
    decoded.lora_snr = normalizedPayload.gateways[0].snr || 0;
  } else {
    decoded.lora_rssi = 0;
    decoded.lora_snr = 0;
  }

  // 数据速率解析
  decoded.lora_datarate = normalizedPayload.spreading_factor
                          || normalizedPayload.data_rate
                          || (normalizedPayload.networks && normalizedPayload.networks.lora &&
normalizedPayload.networks.lora.dr)
                          || "unknown";

} catch (error) {
  console.log('LoRa property parsing error:', error);
  decoded.lora_rssi = 0;
  decoded.lora_snr = 0;
  decoded.lora_datarate = "unknown";
}
```

```
return [
   { field: "Relay_1", value: decoded.Relay_1 },
   { field: "Relay_2", value: decoded.Relay_2 },
   { field: "Relay_3", value: decoded.Relay_3 },
   { field: "Relay_4", value: decoded.Relay_4 },
   { field: "INA_1", value: decoded.INA_1 },
   { field: "INA_2", value: decoded.INA_2 },
   { field: "INA_3", value: decoded.INA_3 },
   { field: "INA_4", value: decoded.INA_4 },
   { field: "humidity", value: decoded.humidity },
   { field: "temperature", value: decoded.temperature },
   { field: "interval", value: decoded.interval },
   { field: "lora_rssi", value: decoded.lora_rssi },
   { field: "lora_snr", value: decoded.lora_snr },
   { field: "lora_datarate", value: decoded.lora_datarate },
];
}
```

10、Follow the steps to add a field. (Every fields is the same way)



11、Press RST button, wait until the sensor connects to the gateway successfully, you will see the

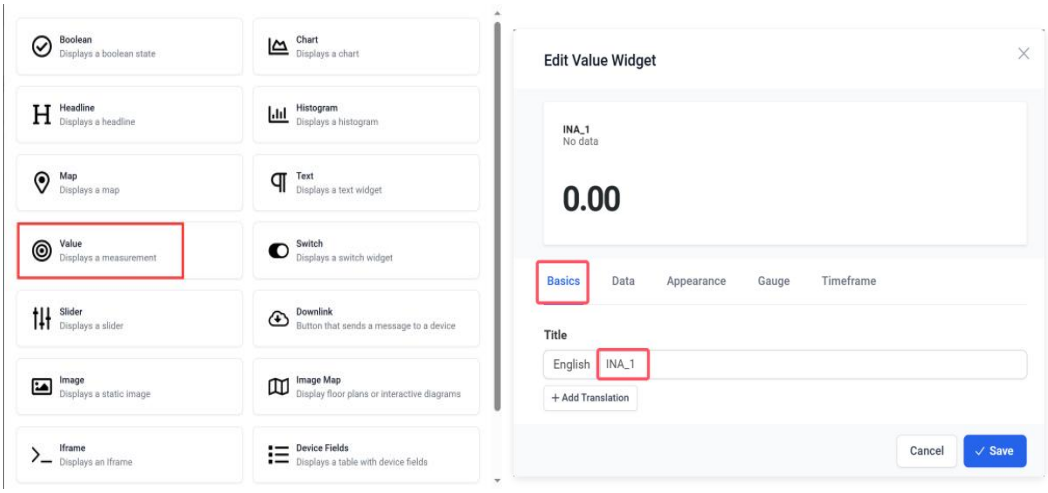
data the sensor is currently reading.

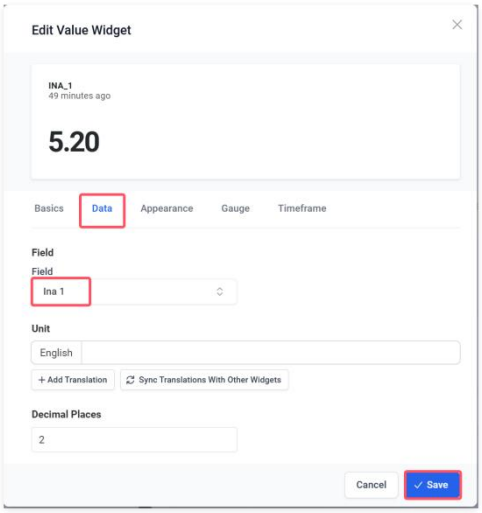
NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
Humidity	HUMIDITY	Float	N/A	46.3	17 minutes ago
Ina 4	INA_4	Integer	N/A	0	17 minutes ago
Ina 2	INA_2	Float	N/A	5.2	17 minutes ago
Ina 1	INA_1	Float	N/A	5.2	17 minutes ago
Ina 3	INA_3	Integer	N/A	0	17 minutes ago
Lora Snr	LORA_SNR	Float	N/A	11.2	17 minutes ago
Lora Datarate	LORA_DATARATE	String	N/A	SF7BW125.0	17 minutes ago
Lora Rssi	LORA_RSSI	Integer	N/A	-72	17 minutes ago
Interval	INTERVAL	Integer	N/A	10	17 minutes ago
Relay 3	RELAY_3	Boolean	N/A	False	17 minutes ago
Relay 4	RELAY_4	Boolean	N/A	False	17 minutes ago
Relay 1	RELAY_1	Boolean	Primary	False	17 minutes ago
Relay 2	RELAY_2	Boolean	Secondary	False	17 minutes ago
Temperature	TEMPERATURE	Float	N/A	29.2	17 minutes ago

12、To get a better look at the data, we can add widget.

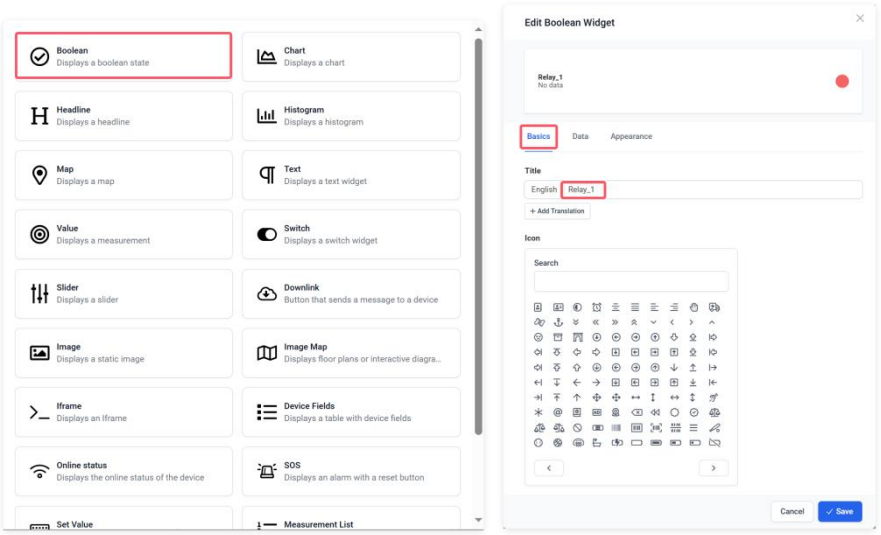
Click “Dashboard-->switch-->+ Add Widget”.

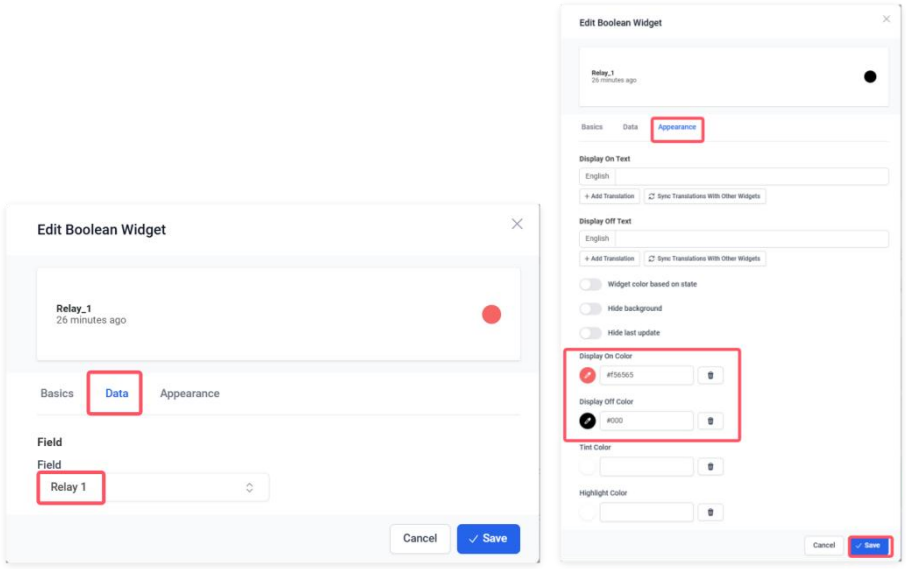
13、Select “Value” and set Title, Field. INA_2, INA_3, INA_4 as the same way.



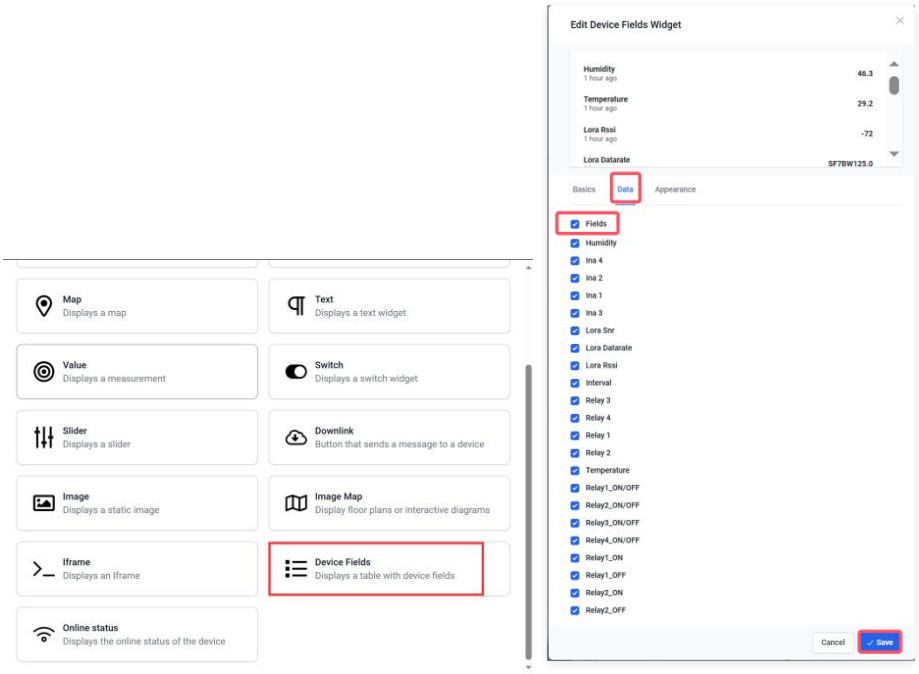


14、Select “Boolean” and set Title, Field as well as the status color. Relay_2, Relay_3, Relay_4 as the same way.

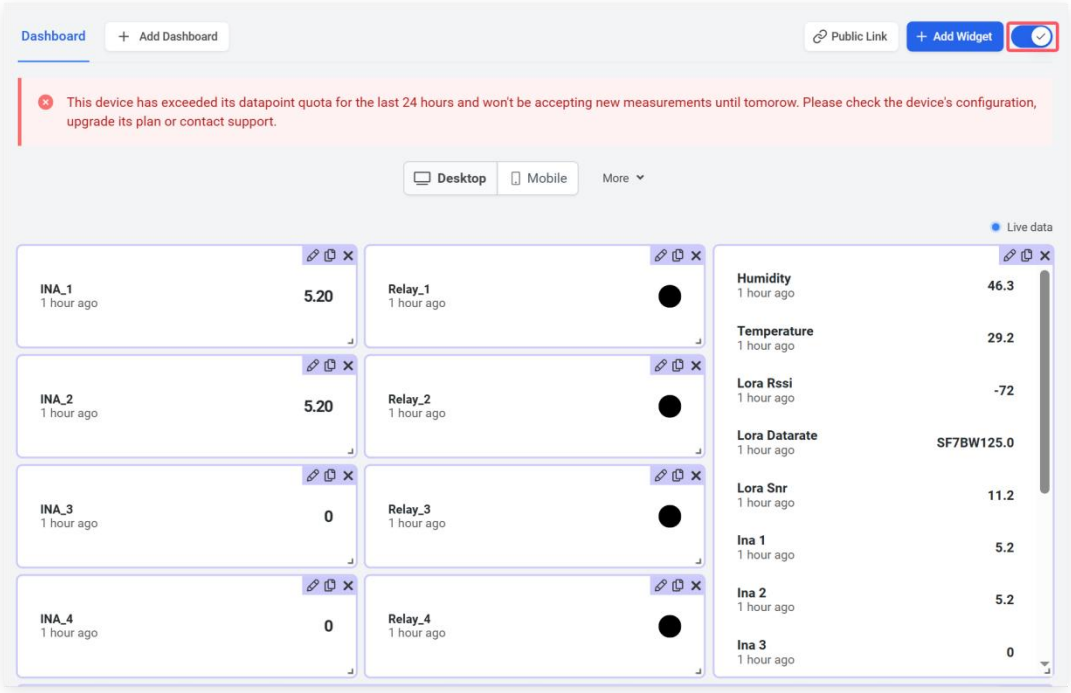




15、Select Device Fields, check “Fields” and click “Save”.



16、Click the switch to save, and you can see the data visually.



3.3.1 Downlink

The downlink has the following functions:

- Modification time interva (Fport1)

Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport2)

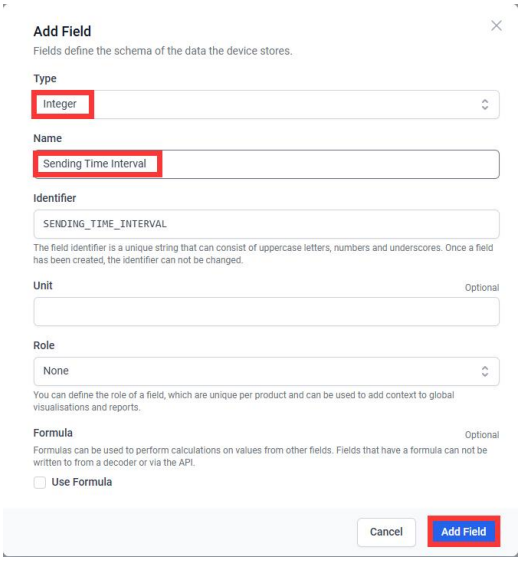
Users can view previous data based on this feature.

- Change the Relay ON/OFF (Fport6-9)

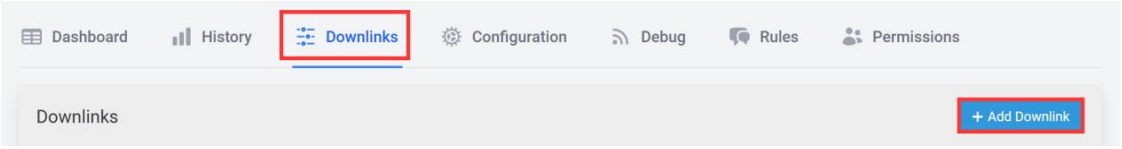
Relay Channel	Fport	Explanation
Relay K1	6	0 is OFF, 1 is ON.
Relay K2	7	
Relay K3	8	
Relay K4	9	

Modify the time interval :

1 、 If you need to change time Interval (Default 60 minutes), you can click "Configuration-->Fields-->+Add Field"



2、Click "Downlink-->Add Downlink".

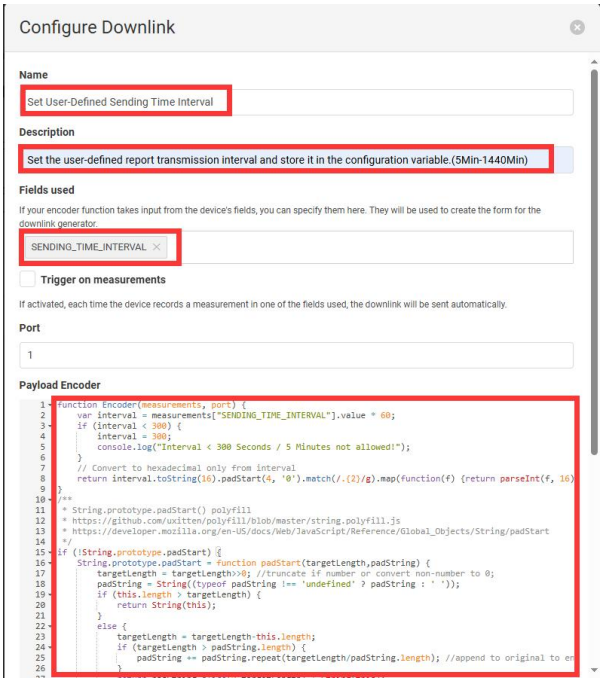


Enter name、description、fields used and payload encoder respectively.

Name: Set User-Defined Sending Time Interval

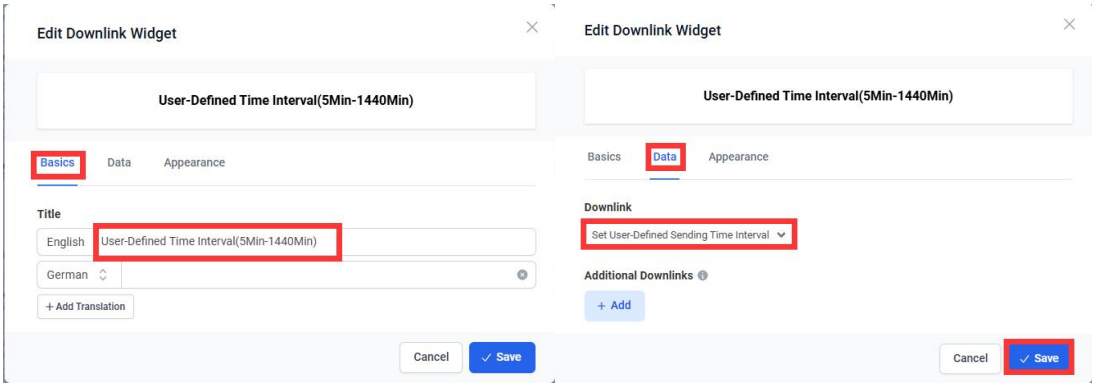
Description: Set the user-defined report transmission interval and store it in the configuration variable.(5Min-1440Min)

Payload Encoder: copy in Github.

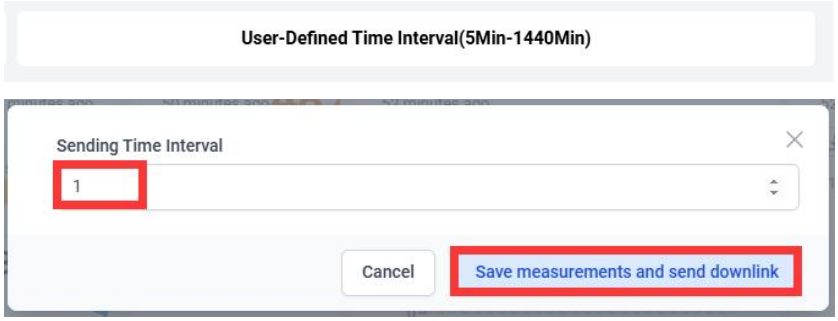


3、Click “Dashboard-->switch-->+ Add Widget”.

Select “Downlink” and setting as follow image.

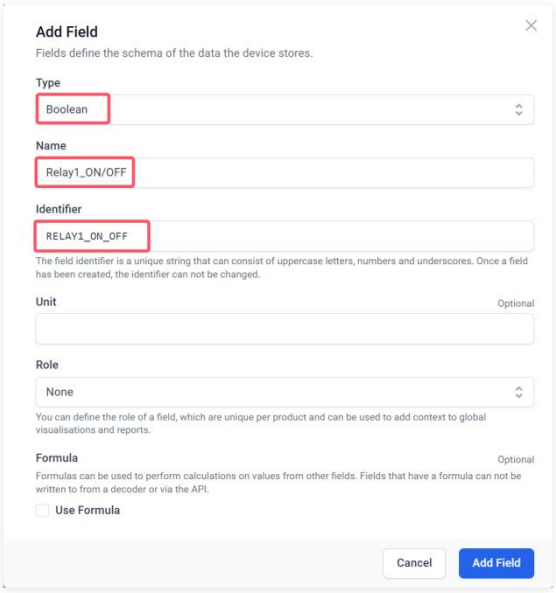


4、Click the switch to save, and you can click to change your time Interval.

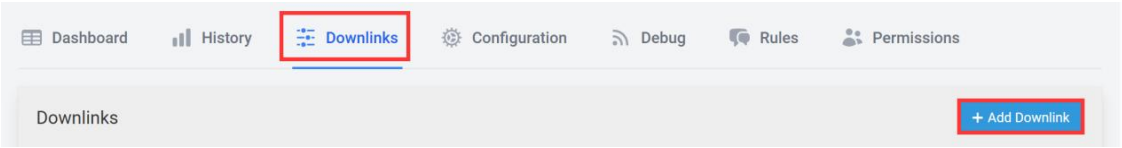


Change the Relay ON/OFF:

1、Click “Configuration-->Fields-->+Add Field”.

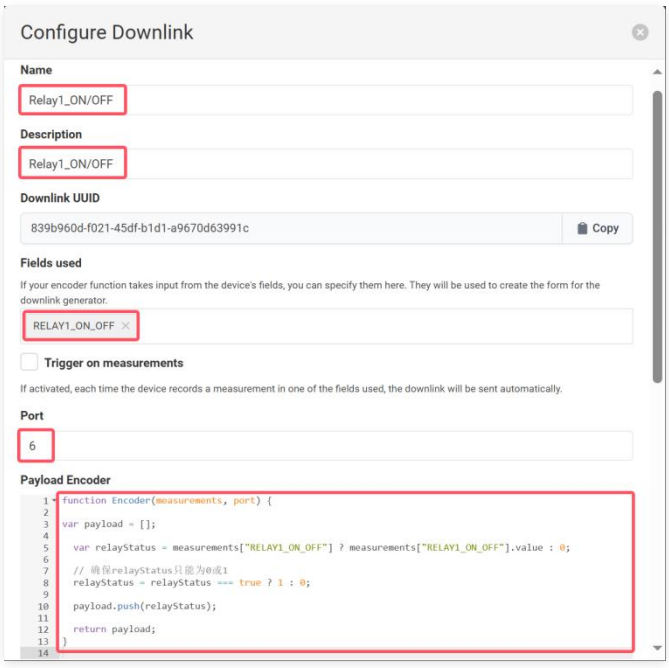


2、Click ”Downlink-->Add Downlink”.



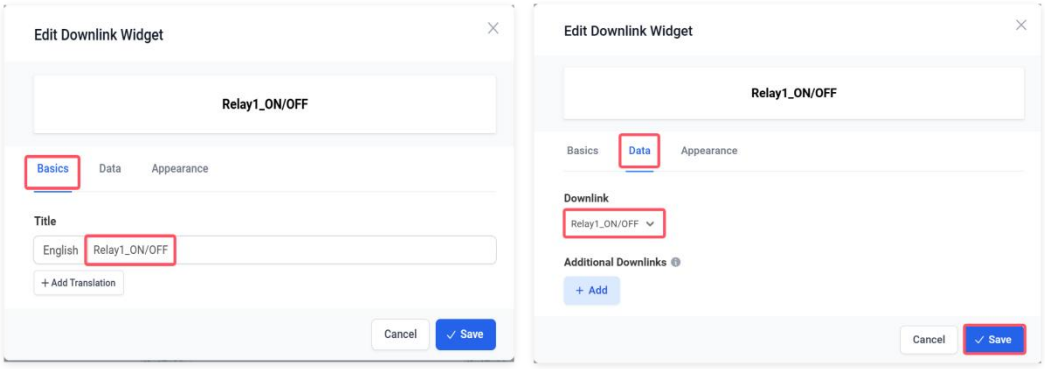
Enter name、description、fields used and payload encoder respectively.

Payload Encoder: copy in Github.



3、Click “Dashboard-->switch-->+ Add Widget”.

Select “Downlink” and setting as follow image.



4、Click the switch to save, and you can click to change Relay1 ON/OFF.

(Check the box to turn it on, otherwise it's off.)