



# **AgroSense\_4 Channel Relay**

## **LoRaWAN® Manual**

### **V1.0**

Author: Yuki

Time: 2025.03.28

# Contents

<b>1 Product Description .....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Feature .....	1
1.3 Parameter .....	2
<b>2 Technical route .....</b>	<b>4</b>
2.1 System Framework .....	4
2.2 Regional frequency band .....	5
<b>3 Usage .....</b>	<b>6</b>
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

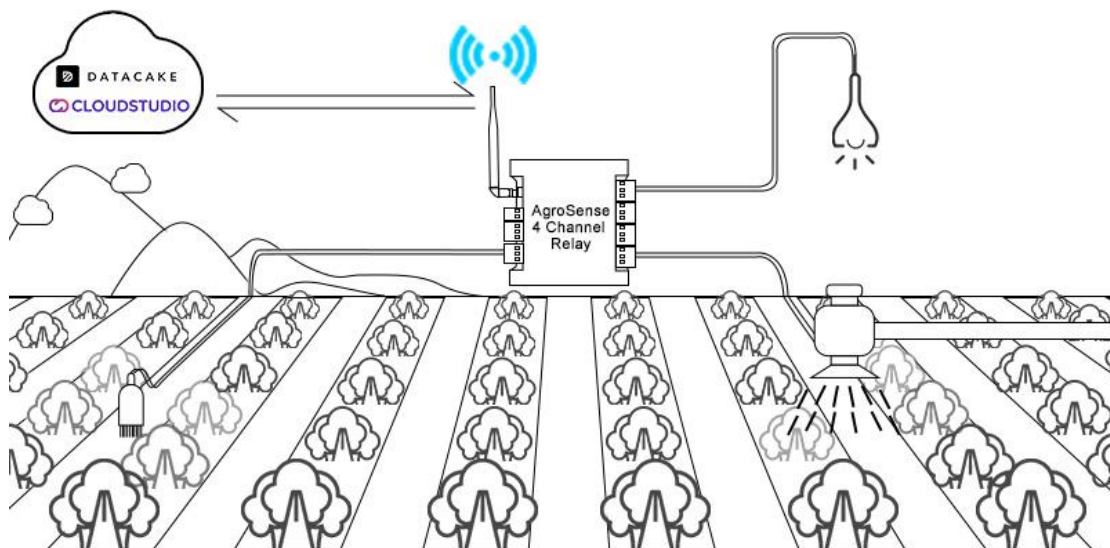
# 1 Product Description

## 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 CloudStudio 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
Humidity Measurement Range	0%-100% RH
Humidity Measurement Accuracy	±2%
Humidity 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 <b>Class C</b>
MAX Transmit Power	22dBm
Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

### 3.Physical Parameters

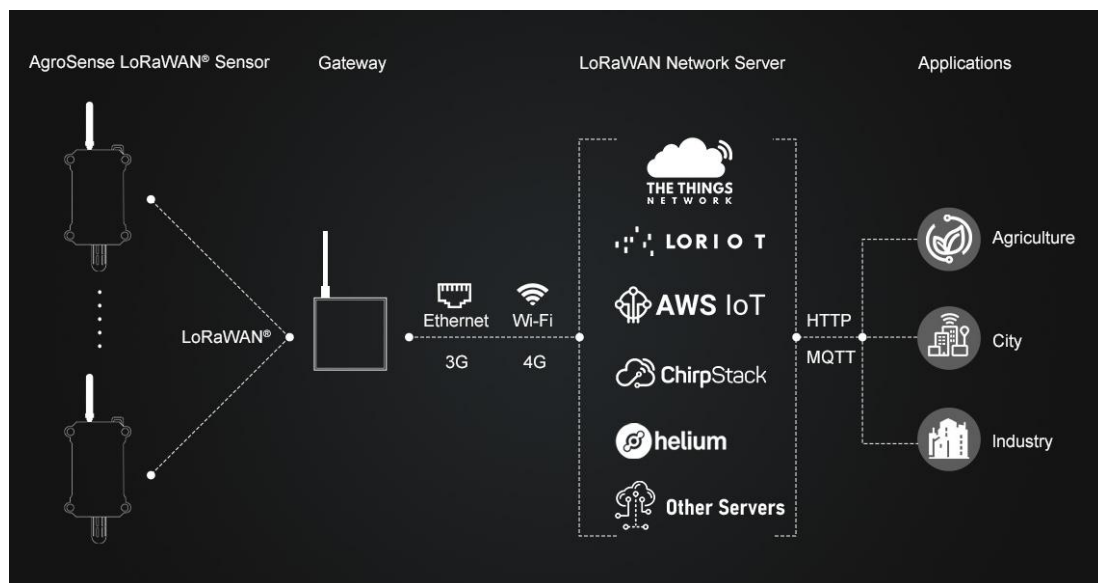
Power Supply	DC 24V
Operating Temperature	-40°C ~85°C
Dimensions	95 × 90 × 40 mm
Mounting	Wall Mounting

## 2 Technical route

### 2.1 System Framework

AgroSense 4\_Channel Relay uses LoRAWAN technology, and its 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).

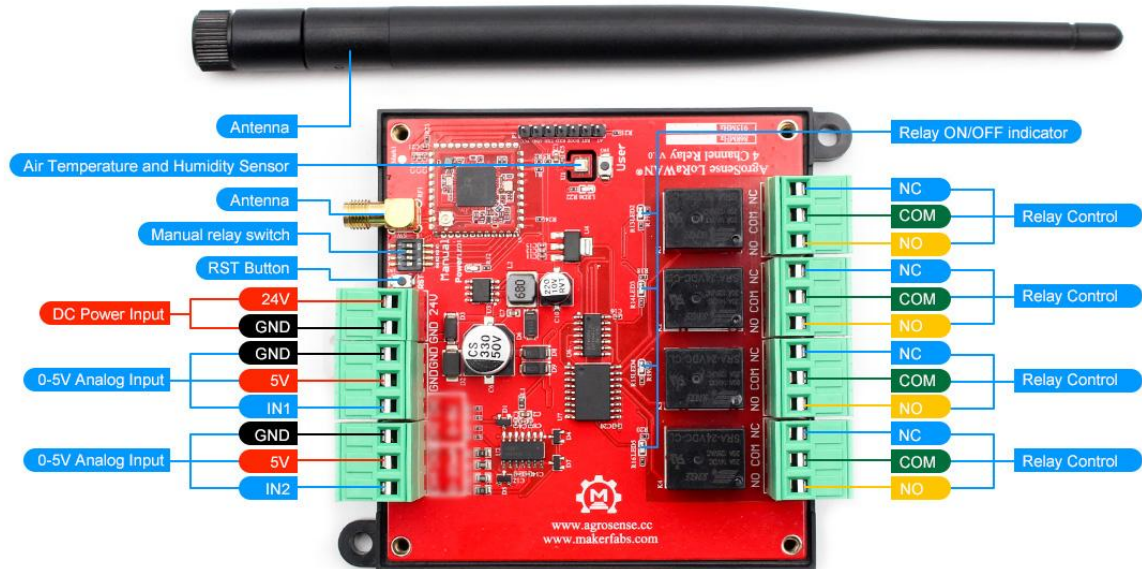
## 2.2 Regional frequency band

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 $\pm 10\%$
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 $\pm 0.2V$ 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	<b>Independent Channel Control:</b> Each relay supports individual programming and real-time status monitoring
	<b>Auto Mode:</b> Receives Downlink commands from cloud platform and Scheduled automation tasks
	<b>Manual Mode:</b> Toggle Manual relay switch for local control
Contact Type:SPDT	<b>COM</b> : Connects to load power
	<b>NO</b> : Closed when relay energized
	<b>NC</b> : 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.

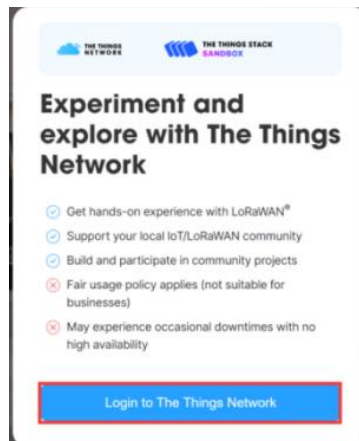
we need to configuration the country/area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

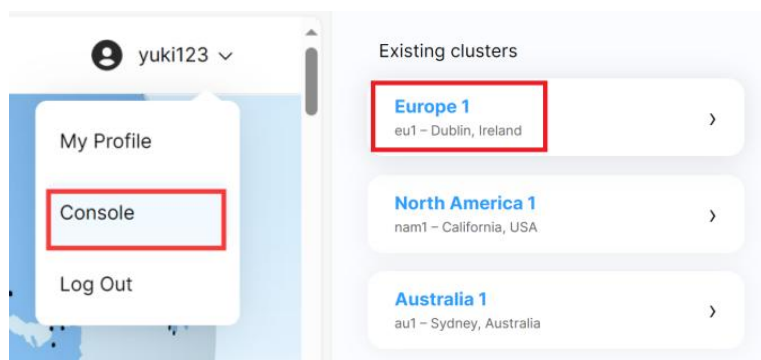
- End Nodes and Gateway: AgroSense\_4 Channel Relay.(The AgroSense series is applicable)
- Network Server: The Things Network. ( Datacake, Lorient, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

### 3.2.1 Network Server configuration

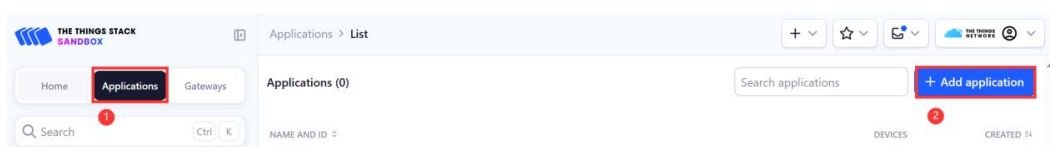
- Open The Things Network in your browser and login it. (Or register an account)



- 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”.

Application ID \*

agrosense-sensor

Application name

My new application

Description

Description for my new application

Optional application description; can also be used to save notes about the

Create application

- Click “+ Register and device”.

End devices

Top end devices Recently active All

No top devices yet

Your most visited and bookmarked end devices will be listed here

+ Register end device

- Following 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.

End device type

Input method ⓘ

☐ Select the end device in the LoRaWAN Device Repository

☒ Enter end device specifics manually 1

Frequency plan ⓘ \*

United States 902-928 MHz, FSB 2 (used by TTN) 2

LoRaWAN version ⓘ \*

LoRaWAN Specification 1.0.3 3

Regional Parameters version ⓘ \*

RP001 Regional Parameters 1.0.3 revision A

Show advanced activation, LoRaWAN class and cluster settings ^ 4

Activation mode ⓘ

☒ Over the air activation (OTAA)

☐ Activation by personalization (ABP)

☐ Define multicast group (ABP & Multicast)

Additional LoRaWAN class capabilities ⓘ

Class C (Continuous) 5

Network defaults ⓘ

☒ Use network's default MAC settings

Cluster settings ⓘ

☐ Skip registration on Join Server

Provisioning information

JoinEUI ⓘ \*

48 FF 00 00 00 00 01 65 Reset

This end device can be registered on the network

DevEUI ⓘ \*

48 E6 63 FF FE 30 01 65 Generate 0/50 used

AppKey ⓘ \*

4A 35 62 6B 95 AB 5B 4D 3F 3B DE 12 71 B1 6F 2A Generate

End device ID ⓘ \*

eui-48e663ffe300165

This value is automatically prefilled using the DevEUI

After registration

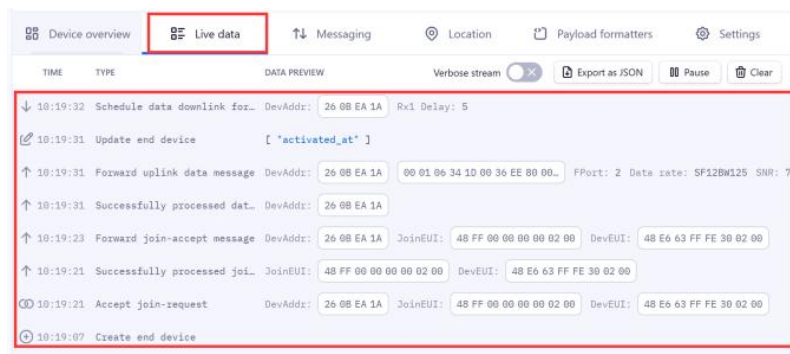
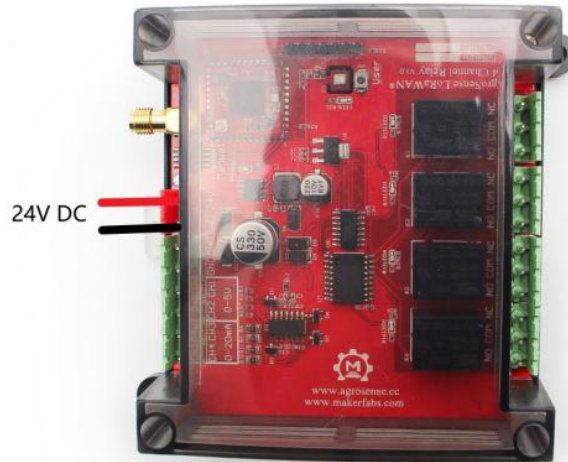
☒ View registered end device

☐ Register another end device of this type

Register end device

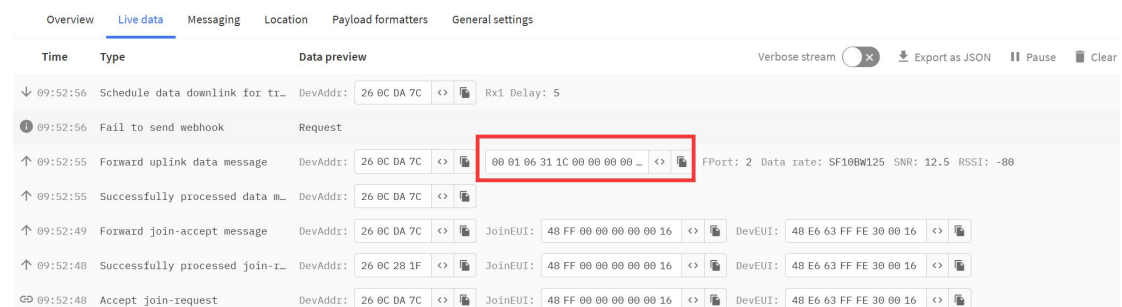


- Connects to 24v DC power supply and press RES button, you can see the device is connected successfully in the TTN.



### 3.2.2 Decoder

- Now, we need to decode the data.



**AgroSense\_4 Channel Relay LoRaWAN®**

Data length	Data description	Value range	Explanation
<b>byte 0</b>	Data packet sequence number high 8 bits	0-0xFFFF	Counting starts from 0 and increments, resetting back to 0 after reaching 65535
<b>byte 1</b>	Data packet sequence number low 8 bits		
<b>byte 2</b>	Relay Status_K1	0/1	0 is invalid, 1 is valid.
<b>byte 3</b>	Relay Status_K2		
<b>byte 4</b>	Relay Status_K3		
<b>byte 5</b>	Relay Status_K4		
<b>byte 6</b>	Humidity 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.
<b>byte 7</b>	Humidity sensor bits 0 to 7		
<b>byte 8</b>	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°C.
<b>byte 9</b>	Temperature sensor bits 0 to 7		
<b>byte 10</b>	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.
<b>byte 11</b>	ADC_1 bits 0 to 7		
<b>byte 12</b>	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.
<b>byte 13</b>	ADC_2 bits 0 to 7		
<b>byte 14</b>	NC		
<b>byte 15</b>	NC		
<b>byte 16</b>	NC		
<b>byte 17</b>	NC		
<b>byte 18</b>	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.
<b>byte 19</b>	data transmission interval bits 16 to 23		
<b>byte 20</b>	data transmission interval bits 8 to 15		

### AgroSense\_4 Channel Relay LoRaWAN®

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

Humidity value is 51.2%.

Temperature value is 22.2°C.

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];
}
```

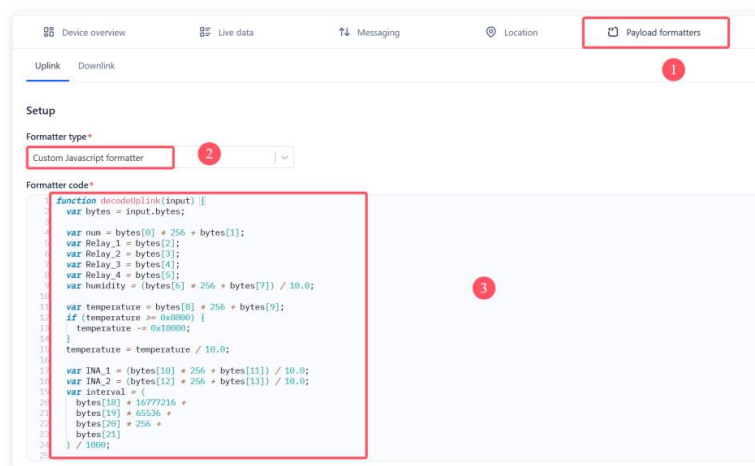
```

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.

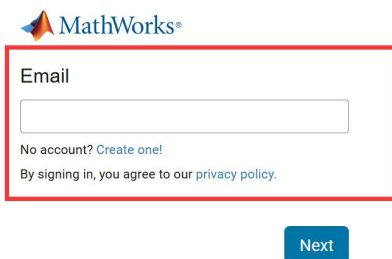


Save changes

### 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)



MathWorks®

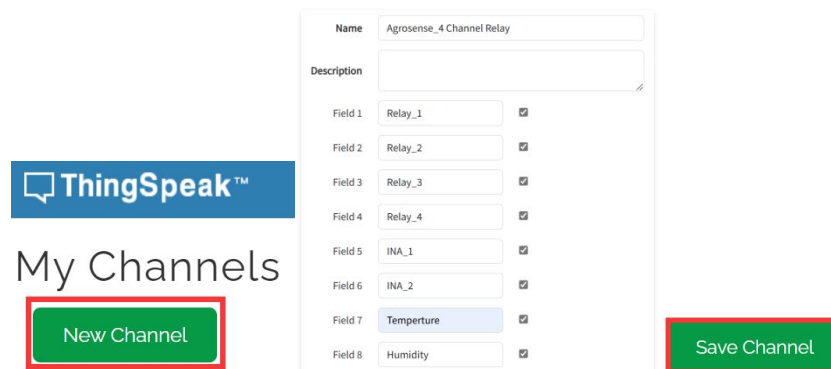
Email

No account? [Create one!](#)

By signing in, you agree to our [privacy policy](#).

Next

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



ThingSpeak™

My Channels

New Channel

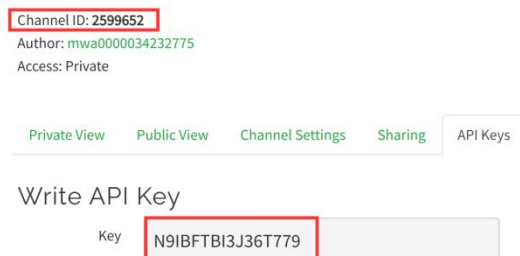
Name: Agrosense\_4 Channel Relay

Description:

Field 1	Relay_1	<input checked="" type="checkbox"/>
Field 2	Relay_2	<input checked="" type="checkbox"/>
Field 3	Relay_3	<input checked="" type="checkbox"/>
Field 4	Relay_4	<input checked="" type="checkbox"/>
Field 5	INA_1	<input checked="" type="checkbox"/>
Field 6	INA_2	<input checked="" type="checkbox"/>
Field 7	Temperature	<input checked="" type="checkbox"/>
Field 8	Humidity	<input checked="" type="checkbox"/>

Save Channel

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



Channel ID: 2599652

Author: mwa000034232775

Access: Private

Private View Public View Channel Settings Sharing API Keys

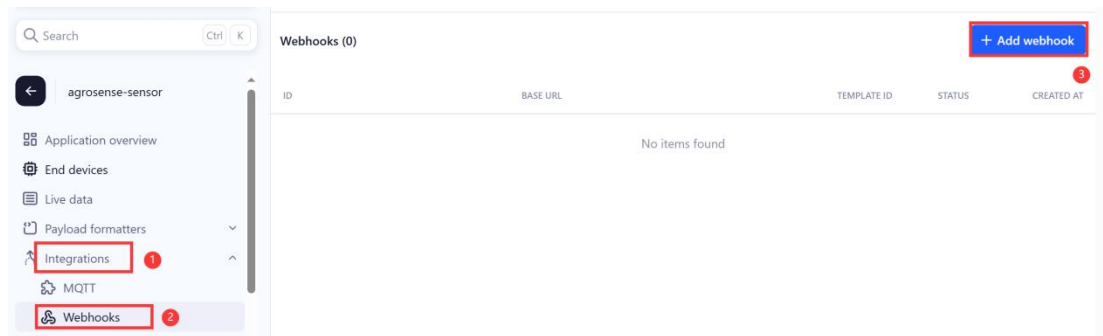
Write API Key

Key: N9IBFTBI3J36T779

### 3.2.4 Connect the Network Server and Application Server

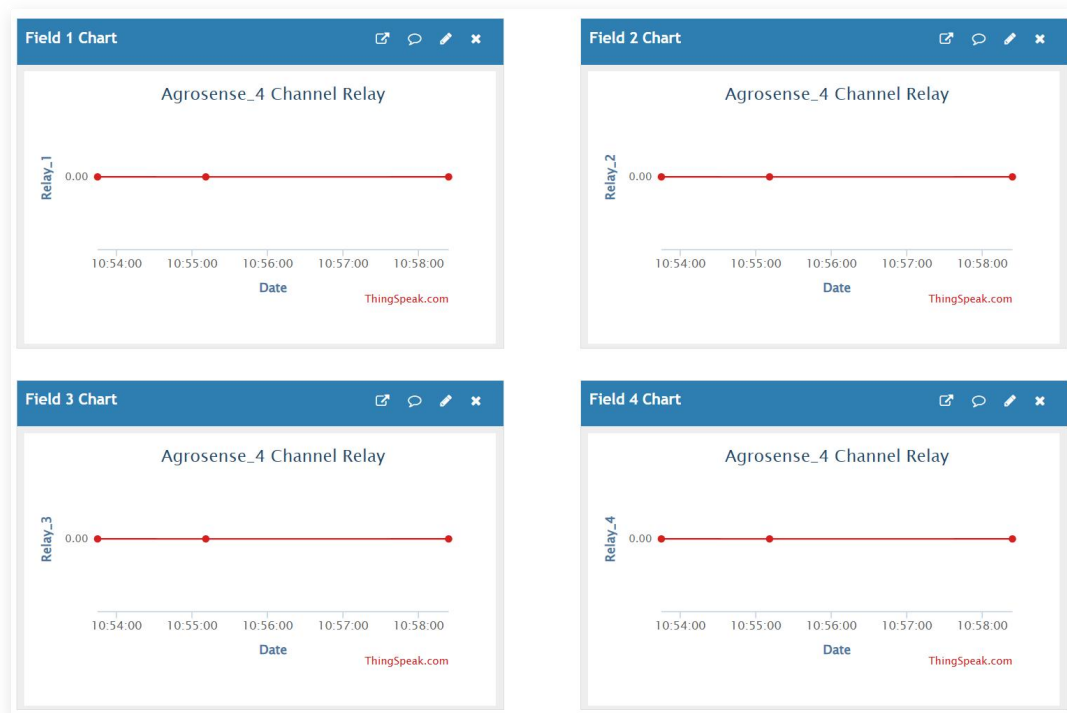
- In the TTN, click “integrations” --> “Webhooks” --> “+ Add webhook”.

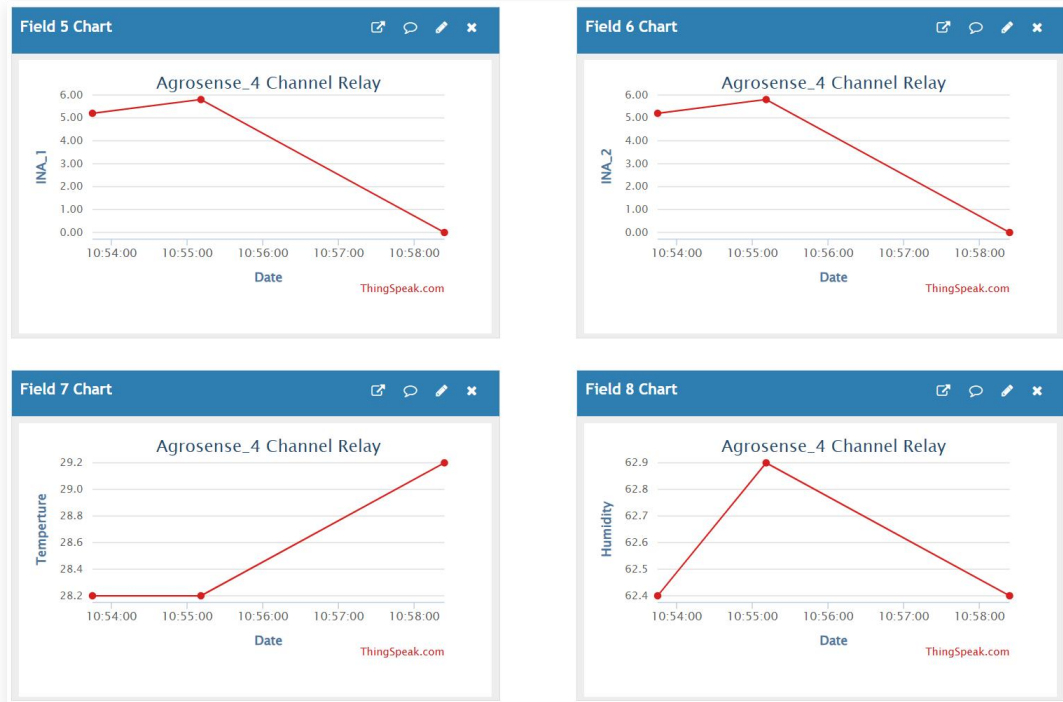




- Select “ThingSpeak”, Fill in the Webhook ID and paste the Channel ID and API Key, click “Create ThingSpeak Webhook”.

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





### 3.2.5 Downlink

The downlink has two functions:

- Modification time interval (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](#).

Device overview Live data Messaging Location Payload formatters

Uplink Downlink

Setup

Formatter type\*  
Custom Javascript formatter

Formatter code\*

```

1 // Encoder function to be used in the TTN console for downlink payload
2 function Encoder(input) {
3   var minutes = input.minutes;
4   // Converting minutes to seconds
5   var seconds = minutes * 60;
6   // If the number of seconds is less than 300 seconds, set it to 300 seconds
7   if (seconds < 300) {
8     seconds = 300;
9   }
10  var payload = [
11    (seconds >> 24) & 0xFF,
12    (seconds >> 16) & 0xFF,
13    (seconds >> 8) & 0xFF,
14    seconds & 0xFF
15  ];
16  return payload;
17 }

```

Save changes

2、Click “Save changes”.

3、Click “Messaging-->Schedule downlink”.

**Note:** you must use this format:

```

{
  "minutes": 5
}

```

Device overview Live data Messaging

Schedule downlink Simulate uplink

Schedule downlink

Insert Mode

☒ Replace downlink queue

☐ Push to downlink queue (append)

FPort\*

1

Payload type

☐ Bytes ☒ JSON

Payload

```

{
  "minutes": 5
}

```

The decoded payload of the downlink message

☐ Confirmed downlink

Schedule downlink

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

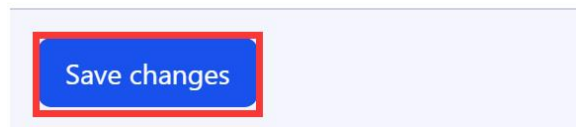
### Change the Relay ON/OFF:

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

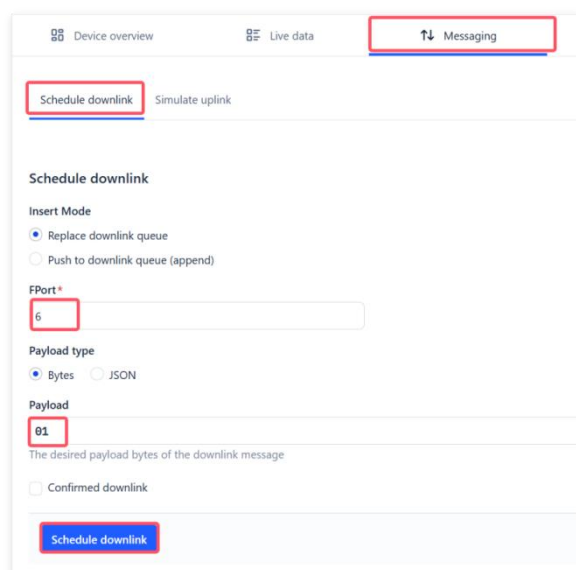
Formatter code you can find in [Github](#).



2、Click “Save changes”.



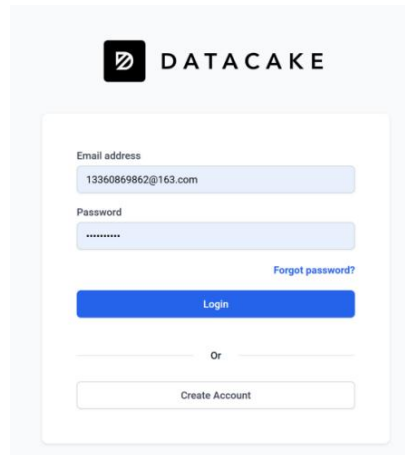
2、Click “Messaging-->Schedule downlink”. The relay will ON/OFF immediately after the modification. ( 00 is OFF, 01 is ON )



### 3.3 Usage with Datacake

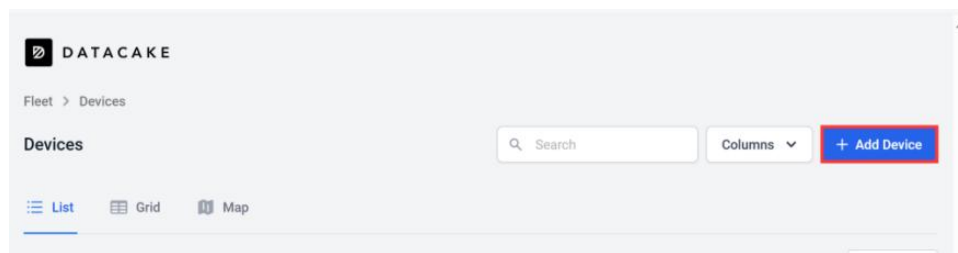
In this phase, we use DataCake(<https://datacake.co/>) as the data server & console.

#### 1、Login datacake or Create Account



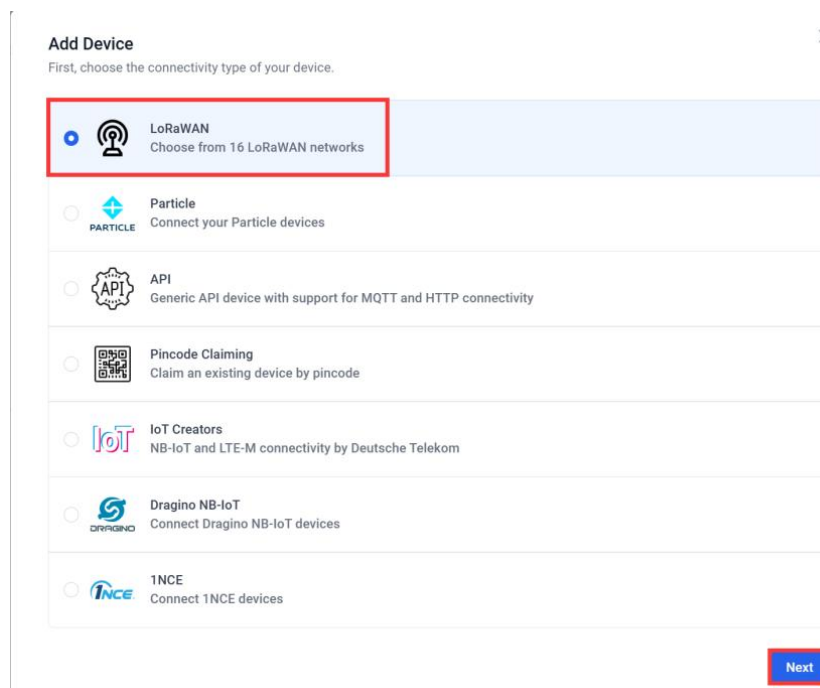
The image shows the DataCake login and registration interface. At the top is the DataCake logo. Below it is a form with two input fields: 'Email address' (containing '13360869862@163.com') and 'Password' (masked with dots). There is a 'Forgot password?' link next to the password field. Below the password field is a blue 'Login' button. Underneath the login button is an 'Or' separator, followed by a 'Create Account' button.

#### 2、Click “Add Device”



The image shows the DataCake dashboard. At the top left is the DataCake logo. Below it is a breadcrumb 'Fleet > Devices'. The main heading is 'Devices'. To the right of the heading are a search bar, a 'Columns' dropdown, and a blue '+ Add Device' button which is highlighted with a red box. Below the heading are three tabs: 'List' (selected), 'Grid', and 'Map'.

#### 3、Select LoRaWAN and click “Next”



The image shows the 'Add Device' dialog box. At the top is the title 'Add Device' and a subtitle 'First, choose the connectivity type of your device.' Below this is a list of connectivity options, each with a radio button, an icon, and a description. The first option, 'LoRaWAN', is selected and highlighted with a red box. The other options are 'Particle', 'API', 'Pincode Claiming', 'IoT Creators', 'Dragino NB-IoT', and '1NCE'. At the bottom right of the dialog is a blue 'Next' button, also highlighted with a red box.



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.

**DATA CAKE**

Fleet > Devices

Devices

Search Columns + Add Device

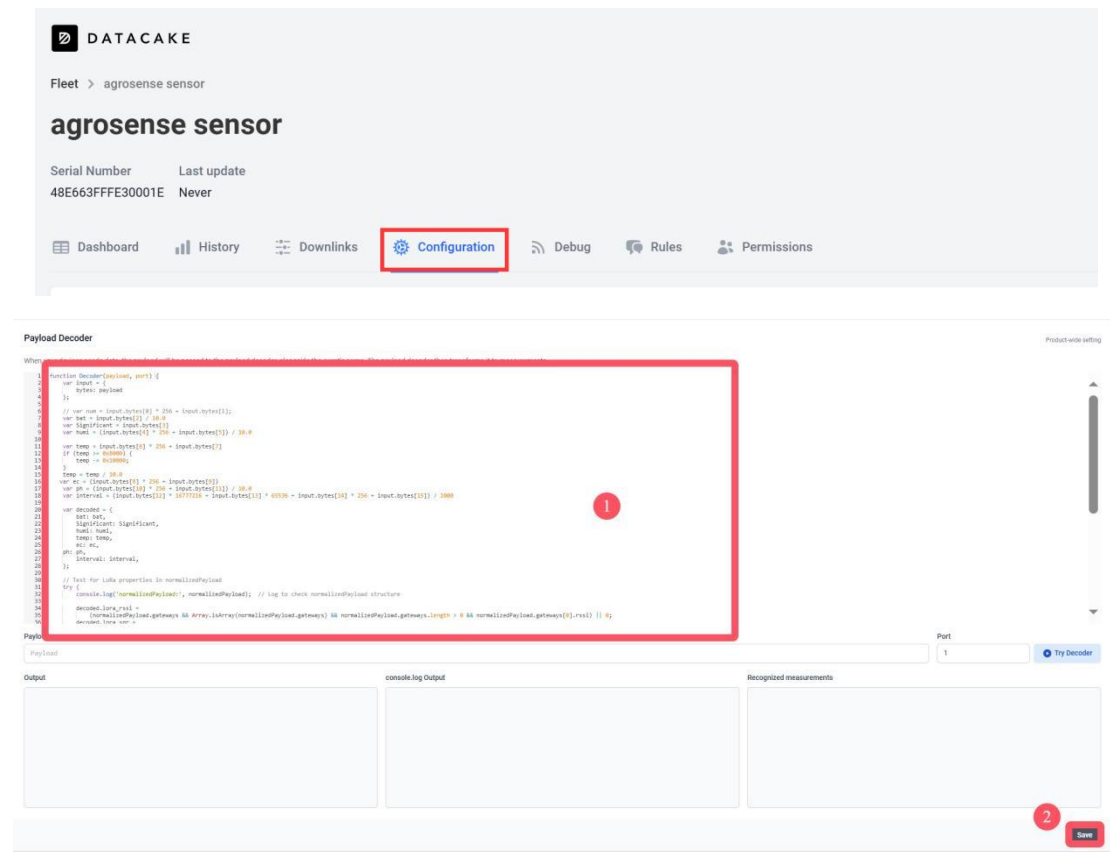
List Grid Map

DEVICE	PRIMARY	SECONDARY	DEVICE SIGNAL	DEVICE BATTERY	Actions
AgroSense_Air Temperature and Humidity Sensor	40.2	25	-48	2.5	👁️ ⋮ ⌵
AgroSense-carbon dioxide (CO2) sensor	0	N/A	-86	3.3	👁️ ⋮ ⌵
agrosense sensor	N/A	N/A	N/A	N/A	👁️ ⋮ ⌵

Showing 1 to 3 of 3 results

50 per page Previous Next

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)

### Fields

Fields describe the data the device will store.

+ Add Field

**Add Field** ✕

Fields define the schema of the data the device stores.

**Type**

Boolean

**Name**

Relay 1

**Identifier**

RELAY\_1

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

**Unit** Optional

**Role**

None

You can define the role of a field, which are unique per product and can be used to add context to global visualisations and reports.

**Formula** Optional

Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

☐ Use Formula

Cancel
Add Field

NAME	IDENTIFIER	TYPE
Humidity	HUMIDITY	Float
Ina 4	INA_4	Integer
Ina 2	INA_2	Float
Ina 1	INA_1	Float
Ina 3	INA_3	Integer
Lora Snr	LORA_SNR	Float
Lora Datarate	LORA_DATARATE	String
Lora Rssi	LORA_RSSI	Integer
Interval	INTERVAL	Integer
Relay 3	RELAY_3	Boolean
Relay 4	RELAY_4	Boolean
Relay 1	RELAY_1	Boolean
Relay 2	RELAY_2	Boolean
Temperature	TEMPERATURE	Float

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

data the sensor is currently reading.

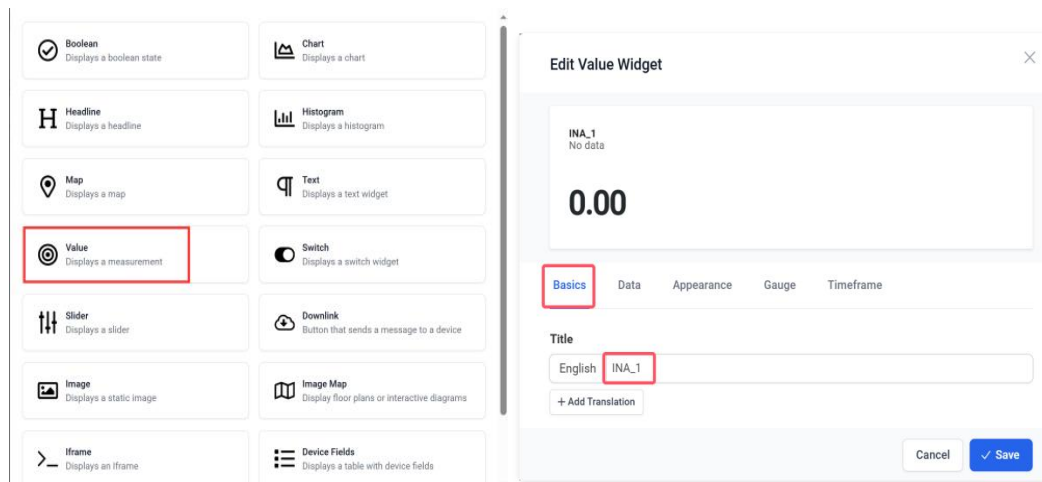
NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
Humidity	HUMIDITY	Float	N/A	46.3	17 minutes ago
Ina 4	INA_4	Integer	N/A	0	17 minutes ago
Ina 2	INA_2	Float	N/A	5.2	17 minutes ago
Ina 1	INA_1	Float	N/A	5.2	17 minutes ago
Ina 3	INA_3	Integer	N/A	0	17 minutes ago
Lora Snr	LORA_SNR	Float	N/A	11.2	17 minutes 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.



**Edit Value Widget**

INA\_1  
49 minutes ago

**5.20**

Basics **Data** Appearance Gauge Timeframe

Field  
Field  
Ina 1

Unit  
English  
+ Add Translation Sync Translations With Other Widgets

Decimal Places  
2

Cancel Save

14、Select “Boolean” and set Title, Field as well as the status color. Relay\_2, Relay\_3, Relay\_4 as the same way.

**Boolean**  
Displays a boolean state

**Chart**  
Displays a chart

**Headline**  
Displays a headline

**Histogram**  
Displays a histogram

**Map**  
Displays a map

**Text**  
Displays a text widget

**Value**  
Displays a measurement

**Switch**  
Displays a switch widget

**Slider**  
Displays a slider

**Downlink**  
Button that sends a message to a device

**Image**  
Displays a static image

**Image Map**  
Displays floor plans or interactive diagrams

**Iframe**  
Displays an iframe

**Device Fields**  
Displays a table with device fields

**Online status**  
Displays the online status of the device

**SOS**  
Displays an alarm with a reset button

**Set Value**

**Measurement List**

**Edit Boolean Widget**

Relay\_1  
No data

Basics **Data** Appearance

Title  
English Relay\_1  
+ Add Translation

Icon  
Search

Cancel Save

**Edit Boolean Widget**

Relay\_1  
26 minutes ago

Basics **Data** **Appearance**

Display On Text  
English  
+ Add Translation Sync Translations With Other Widgets

Display Off Text  
English  
+ Add Translation Sync Translations With Other Widgets

Widget color based on state

Hide background

Hide last update

Display On Color  
#ff0000

Display Off Color  
#000000

Text Color

Highlight Color

Cancel Save



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”

**Add Field**

Fields define the schema of the data the device stores.

Type: Integer

Name: Sending Time Interval

Identifier: SENDING\_TIME\_INTERVAL

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

Unit: Optional

Role: None

You can define the role of a field, which are unique per product and can be used to add context to global visualisations and reports.

Formula: Optional

Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

☐ Use Formula

Cancel Add Field

2、 Click “Downlink-->Add Downlink”.

Dashboard History **Downlinks** Configuration Debug Rules Permissions

Downlinks + 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](#).

**Configure Downlink**

Name  
Set User-Defined Sending Time Interval

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

Fields used  
If your encoder function takes input from the device's fields, you can specify them here. They will be used to create the form for the downlink generator.  
SENDING\_TIME\_INTERVAL

Trigger on measurements  
☐

Port  
1

Payload Encoder  

```

1 function Encoder(measurements, port) {
2   var interval = measurements["SENDING_TIME_INTERVAL"].value * 60;
3   if (interval < 300) {
4     interval = 300;
5     console.log("Interval < 300 Seconds / 5 Minutes not allowed!");
6   }
7   // Convert to hexadecimal only from interval
8   return interval.toString(16).padStart(4, '0').match(/.{2}/g).map(function(f) { return parseInt(f, 16);
9   });
10  }
11  // String.prototype.padStart() polyfill
12  * https://github.com/unix/polyfill/blob/master/string/polyfill.js
13  * https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/padStart
14  */
15  if (!String.prototype.padStart) {
16    String.prototype.padStart = function padStart(targetLength,padString) {
17      targetLength = targetLength||0; //truncate if number or convert non-number to 0;
18      padString = String((typeof padString !== 'undefined' ? padString : ' '));
19      if (this.length > targetLength) {
20        return String(this);
21      }
22      else {
23        targetLength = targetLength-this.length;
24        if (targetLength > padString.length) {
25          padString += padString.repeat(targetLength/padString.length); //append to original to en
26        }

```

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

Select “Downlink” and setting as follow image.

**Edit Downlink Widget**

User-Defined Time Interval(5Min-1440Min)

Basics Data Appearance

Title  
English User-Defined Time Interval(5Min-1440Min)  
German  
+ Add Translation

Cancel Save

**Edit Downlink Widget**

User-Defined Time Interval(5Min-1440Min)

Basics Data Appearance

Downlink  
Set User-Defined Sending Time Interval

Additional Downlinks  
+ Add

Cancel Save

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

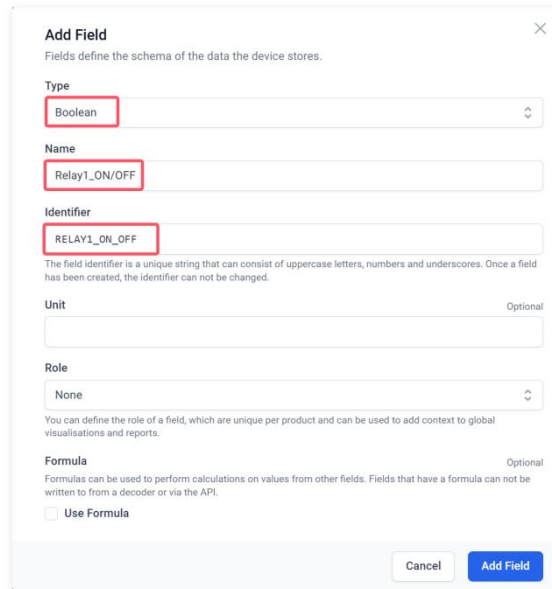
**User-Defined Time Interval(5Min-1440Min)**

Sending Time Interval  
1

Cancel Save measurements and send downlink

## Change the Relay ON/OFF:

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



**Add Field**  
Fields define the schema of the data the device stores.

Type: Boolean

Name: Relay1\_ON/OFF

Identifier: RELAY1\_ON\_OFF

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

Unit: Optional

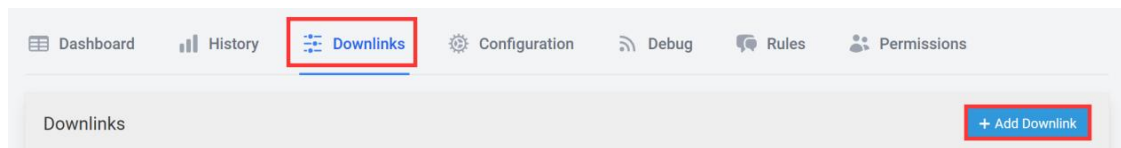
Role: None

You can define the role of a field, which are unique per product and can be used to add context to global visualisations and reports.

Formula: Optional  
Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.  
☐ Use Formula

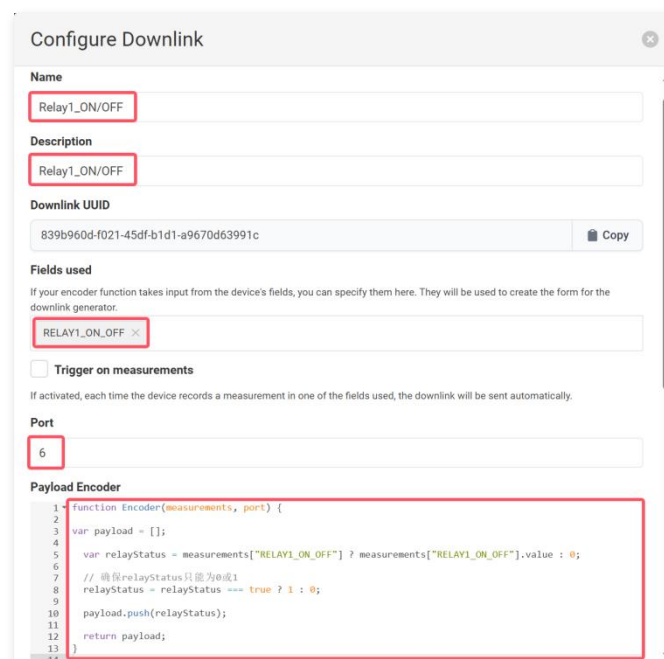
Cancel Add Field

2、Click “Downlink-->Add Downlink”.



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

Payload Encoder: copy in [Github](#).



**Configure Downlink**

Name: Relay1\_ON/OFF

Description: Relay1\_ON/OFF

Downlink UUID: 839b960d-f021-45df-b1d1-a9670d63991c Copy

Fields used  
If your encoder function takes input from the device's fields, you can specify them here. They will be used to create the form for the downlink generator.  
RELAY1\_ON\_OFF X

☐ Trigger on measurements  
If activated, each time the device records a measurement in one of the fields used, the downlink will be sent automatically.

Port: 6

Payload Encoder

```

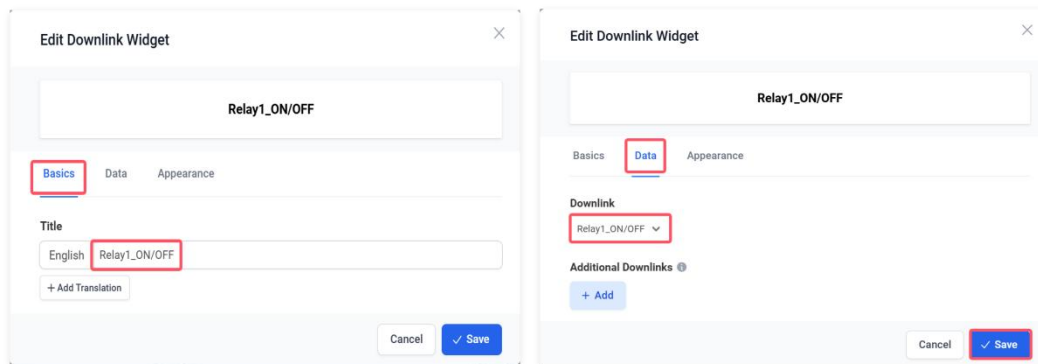
1 function Encoder(measurements, port) {
2   var payload = [];
3   var relayStatus = measurements["RELAY1_ON_OFF"] ? measurements["RELAY1_ON_OFF"].value : 0;
4   // 确保relayStatus只能为0或1
5   relayStatus = relayStatus === true ? 1 : 0;
6   payload.push(relayStatus);
7   return payload;
8 }

```



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.)

