AgroSense_2 Channel Non-invasive AC Current Monitor

LoRaWAN® Manual

V1.1

Author: Yuki

Time: 2025.05.20

目录

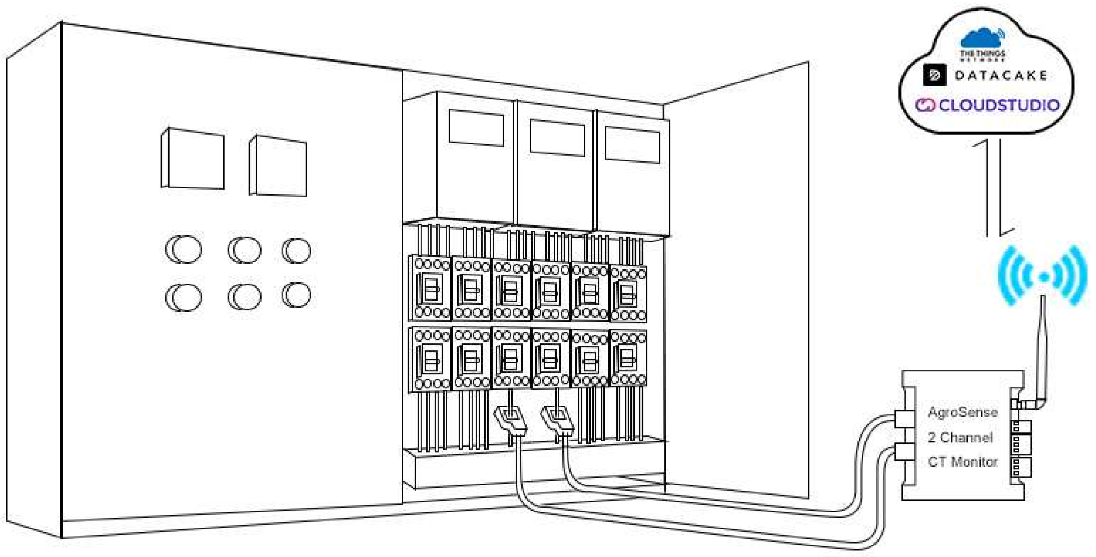
1 Product Description

1.1 Introduction

AgroSense_2 Channel Non-invasive AC Current Monitor integrates two non-intrusive clamp-on current transformers, two 0–5V analog inputs, and a high-precision temperature and humidity sensor into a single compact IoT device. Simply snap the CTs onto conductors—no power interruption or wiring modifications required—and begin monitoring energy consumption (default range: 100A /1V; customizable to other ranges with compatible interfaces) alongside environmental conditions. Reporting data once per minute over LoRaWAN® (up to 5 km line-of-sight), it delivers near-real-time insights for smart agriculture, industrial automation, and intelligent buildings, driving targeted efficiency improvements, cost savings, and optimized energy management.

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 adjust data reporting intervals. Local storage for 3,300+ records ensures critical data is retained during connectivity gaps.



1.2 Feature

- Includes 2-channel independently **CT Monitor.**
- 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.
- 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	AGLW2CT
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
CT Supply	100A/1V
CT channel count	2

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

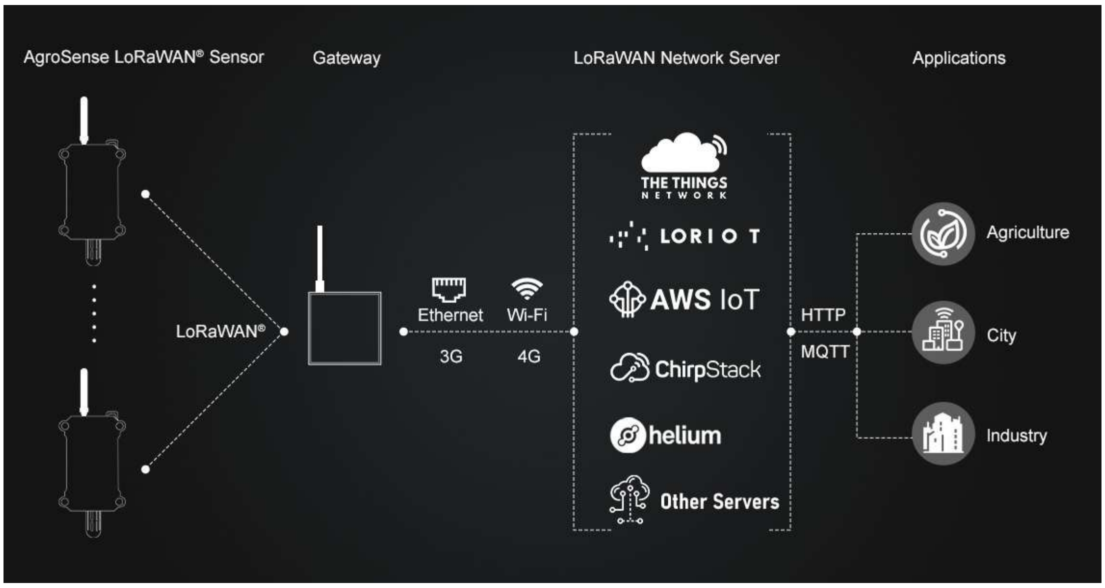
Power Supply	DC 12-24V
Operating Temperature	-40°C ~85°C
Dimensions	95 × 90 × 40 mm
Mounting	Wall Mounting

2 Technical route

2.1 System Framework

AgroSense_2 Channel Non-invasive AC Current Monitor 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 minute.)

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., modify configuration).

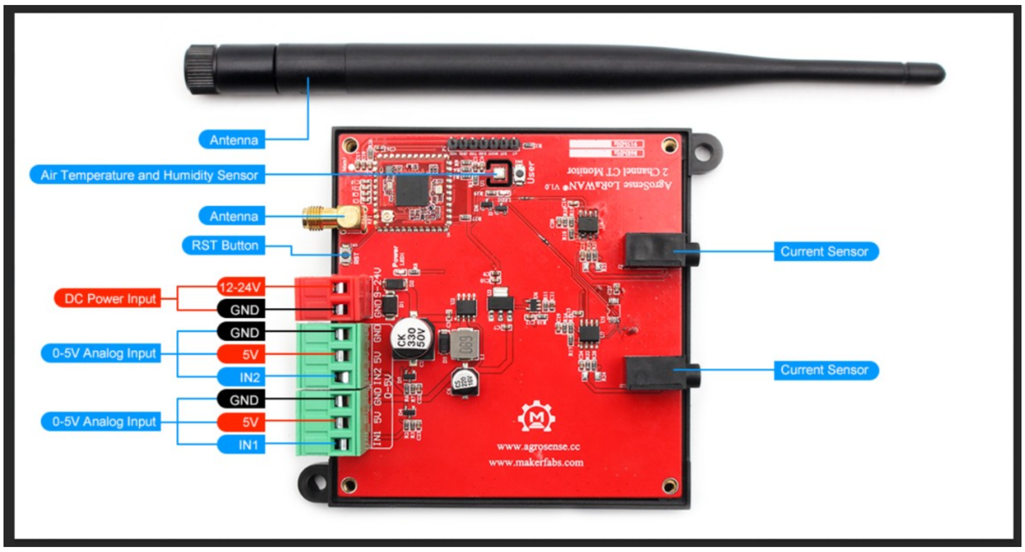
2.2 Regional frequency band

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_2 Channel Non-invasive AC Current Monitor operates from an external DC power source.

Power Specifications	DC 12-24V ±10%
Terminal Definitions	12-24V: Positive terminal (power input)
	GND: Negative terminal (ground)

3.1.2 Dual-Channel 0-5V Analog Input

AgroSense_2 Channel Non-invasive AC Current Monitor 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_2 Channel Non-invasive AC Current Monitor 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 TWO-Channel CT Monitor

AgroSense_2 Channel Non-invasive AC Current Monitor supports two independent, plug-and-play current detection channels, enabling users to begin monitoring energy consumption instantly without any wiring modifications.

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.

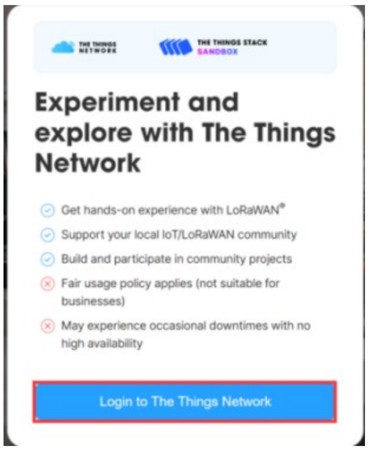
we need to configuration the country/area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

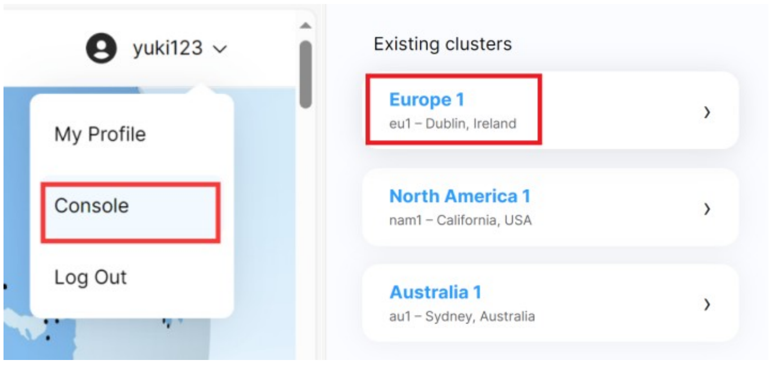
- Open The Things Network in your browser and login it. (Or register an account)



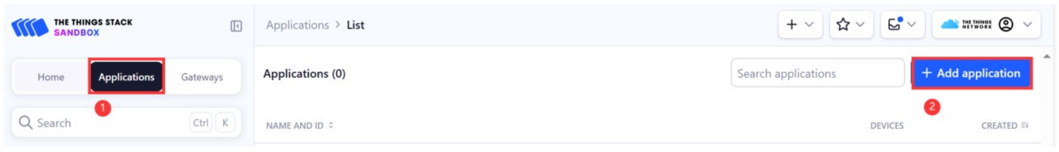
Experiment and explore with The Things Network

- Get hands-on experience with LoRaWAN®
- Support your local IoT/LoRaWAN community
- Build and participate in community projects
- Fair usage policy applies (not suitable for businesses)
- May experience occasional downtimes with no high availability

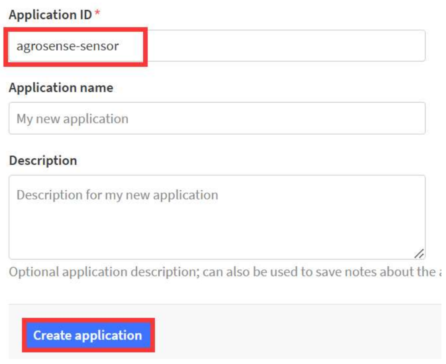
- Click “Console” and select clusters. (we take the European region for example.)



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



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



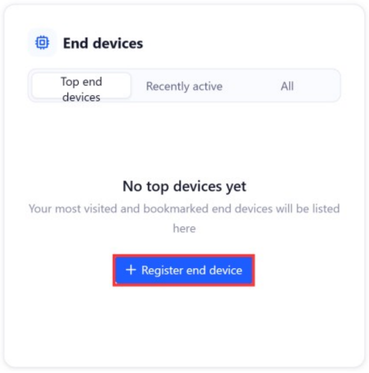
Application ID *
agrosense-sensor

Application name
My new application

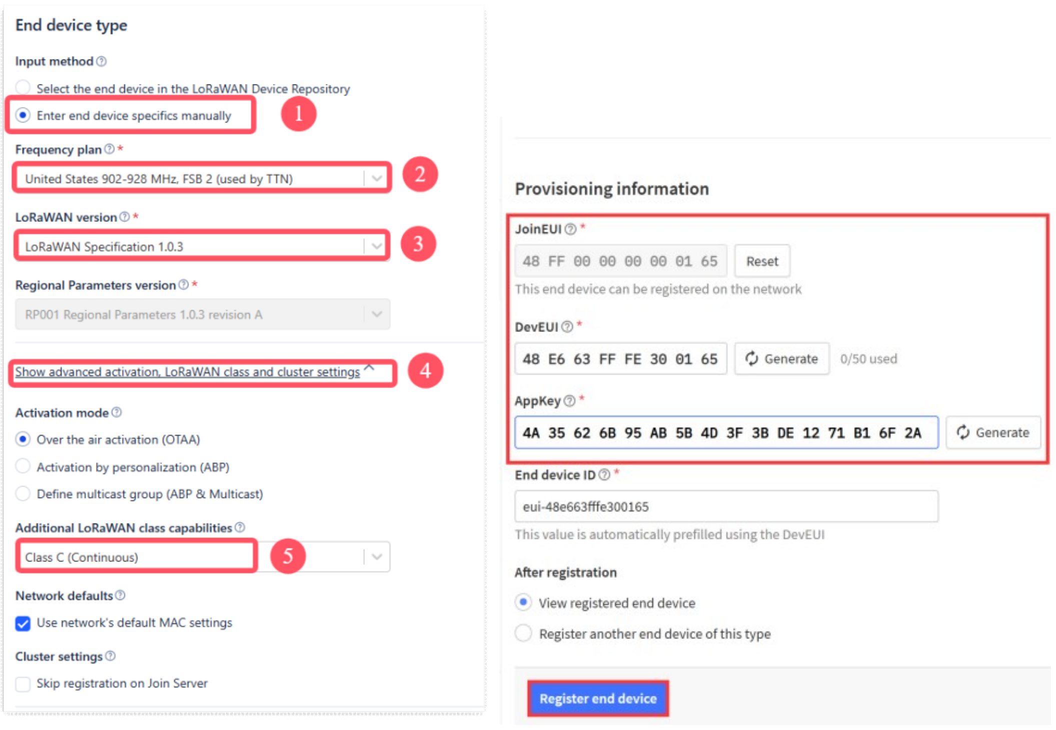
Description
Description for my new application

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

- 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.



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

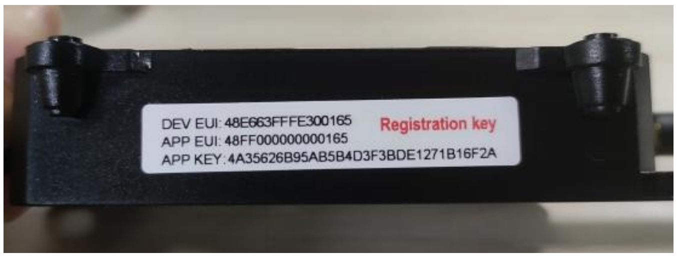
DevEUI *
48 E6 63 FF FE 30 01 65 Generate 0/50 used

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

End device ID *
eui-48e663fffe300165
This value is automatically prefilled using the DevEUI

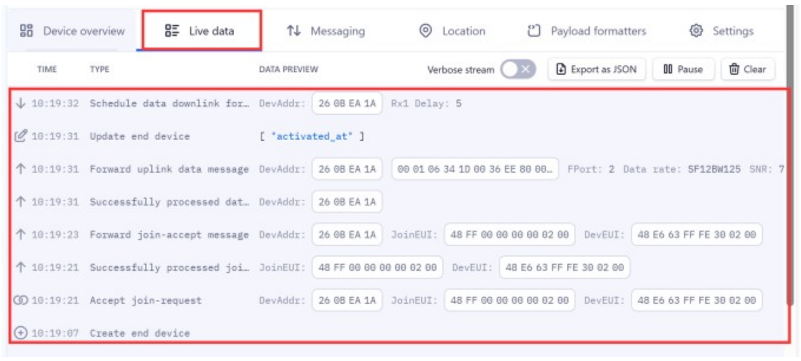
After registration
View registered end device
Register another end device of this type

Register end device



DEV EUI: 48E663FFFE300165
APP EUI: 48FF000000000165
APP KEY: 4A35626B95AB5B4D3F3BDE1271B16F2A

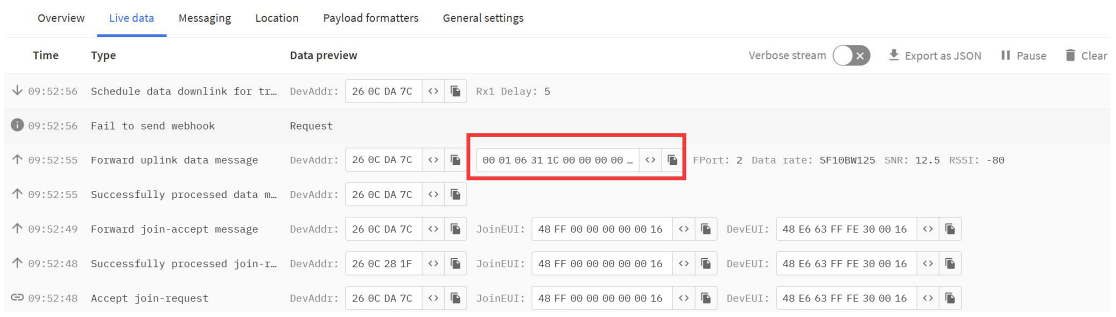
- Connects to 12-24VDC power supply and press RES button, you can see the device is connected successfully in the TTN.



```
TIME      TYPE                              DATA PREVIEW
10:19:32  Schedule data downlink for…       DevAddr: 26 0B EA 1A  Rx1 Delay: 5
10:19:31  Update end device                 [ "activated_at" ]
10:19:31  Forward uplink data message       DevAddr: 26 0B EA 1A  00 01 06 34 1D 00 36 EE 80 00…  FPort: 2  Data rate: SF12BW125  SNR: 7
10:19:31  Successfully processed dat…       DevAddr: 26 0B EA 1A
10:19:23  Forward join-accept message       DevAddr: 26 0B EA 1A  JoinEUI: 48 FF 00 00 00 00 02 00  DevEUI: 48 E6 63 FF FE 30 02 00
10:19:21  Successfully processed joi…       JoinEUI: 48 FF 00 00 00 00 02 00  DevEUI: 48 E6 63 FF FE 30 02 00
10:19:21  Accept join-request               DevAddr: 26 0B EA 1A  JoinEUI: 48 FF 00 00 00 00 02 00  DevEUI: 48 E6 63 FF FE 30 02 00
10:19:07  Create end device
```

3.2.2 Decoder

- Now, we need to decoder the data.



```
Time      Type                              Data preview
09:52:56  Schedule data downlink for tr…    DevAddr: 26 0C DA 7C  Rx1 Delay: 5
09:52:56  Fail to send webhook              Request
09:52:55  Forward uplink data message       DevAddr: 26 0C DA 7C  00 01 06 31 1C 00 00 00 00 …  FPort: 2  Data rate: SF10BW125  SNR: 12.5  RSSI: -80
09:52:55  Successfully processed data m…    DevAddr: 26 0C DA 7C
09:52:49  Forward join-accept message       DevAddr: 26 0C DA 7C  JoinEUI: 48 FF 00 00 00 00 00 16  DevEUI: 48 E6 63 FF FE 30 00 16
09:52:48  Successfully processed join-r…    DevAddr: 26 0C 28 1F  JoinEUI: 48 FF 00 00 00 00 00 16  DevEUI: 48 E6 63 FF FE 30 00 16
09:52:48  Accept join-request               DevAddr: 26 0C DA 7C  JoinEUI: 48 FF 00 00 00 00 00 16  DevEUI: 48 E6 63 FF FE 30 00 16
```

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	ADC_1 bits 8 to 15		The value is amplified by a factor of 10. To get the actual value, divide it by 10. For example, if the value is 0x21 (33), the actual voltage is 3.3 V
byte 3	ADC_1 bits 0 to 7		
byte 4	ADC_2 bits 8 to 15		The value is amplified by a factor of 10. To get the actual value, divide it by 10. For example, if the value is 0x21 (33), the actual voltage is 3.3 V
byte 5	ADC_2 bits 0 to 7		
byte 6	Current_1 bits 8 to 15		This value is obtained after magnifying the data by 1000 times.
byte 7	Current_1 bits 0 to 7		

byte 8	Current_2 bits 8 to 15		This value is obtained after magnifying the data by 1000 times.
byte 9	Current_2 bits 0 to 7		
byte 10	Humidity sensor bits 8 to 15		The value is amplified by a factor of 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 11	Humidity sensor bits 0 to 7		
byte 12	Temperature sensor bits 8 to 15		The value is amplified by a factor of 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 Temperature value obtained is 0x00000285= 645. After converting and dividing by 10, the actual Temperature is 6.45℃.
byte 13	Temperature sensor bits 0 to 7		
byte 14	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 15	data transmission interval bits 16 to 23		
byte 16	data transmission interval bits 8 to 15		
byte 17	data transmission interval bits 0 to 7		
Downlink			
Fport 1	Change the data sending interval		
Fport 5	Upload the quantity of the latest local logged data		

Example:0x00 0x8C 0x00 0x08 0x00 0x20 0x01 0x40 0x00 0x40 0x01 0xCC 0x01 0x15 0x00 0x00 0xEA 0x60

Data parsing:

ADC1 value is 1.0v

ADC2 value is 3.2v

Current_1 is 0.5v

Current_2 is 0.1v

Humility value is 46.1%

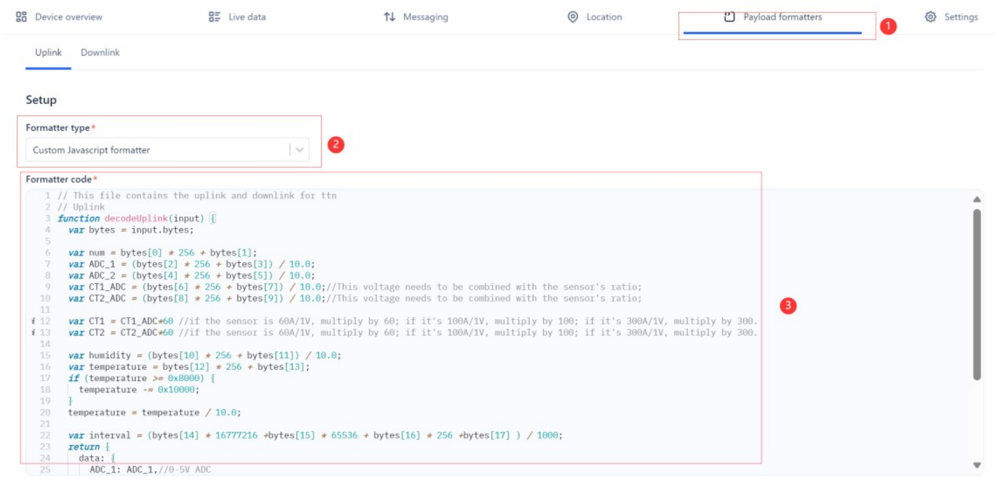
Temperature value is 27.8℃

Data transmission interval value is 60s

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

```
// This file contains the uplink and downlink for ttn
// Uplink
function decodeUplink(input) {
    var bytes = input.bytes;

    var num = bytes[0] * 256 + bytes[1];
    var ADC_1 = (bytes[2] * 256 + bytes[3]) / 10.0;
    var ADC_2 = (bytes[4] * 256 + bytes[5]) / 10.0;
    var CT1_ADC = (bytes[6] * 256 + bytes[7]) / 10.0;//This voltage needs to be combined with the sensor's ratio;
    var CT2_ADC = (bytes[8] * 256 + bytes[9]) / 10.0;//This voltage needs to be combined with the sensor's ratio;

    var CT1 = CT1_ADC*60 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V, multiply by 300.
    var CT2 = CT2_ADC*60 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V, multiply by 300.

    var humidity = (bytes[10] * 256 + bytes[11]) / 10.0;
    var temperature = bytes[12] * 256 + bytes[13];
    if (temperature >= 0x8000) {
        temperature -= 0x10000;
    }
    temperature = temperature / 10.0;

    var interval = (bytes[14] * 16777216 +bytes[15] * 65536 + bytes[16] * 256 +bytes[17] ) / 1000;
    return {
        data: {
            ADC_1: ADC_1,//0-5V ADC
            ADC_2: ADC_2,//0-5V ADC
            CT1: CT1,//A
            CT2: CT2,//A
            temperature: temperature,
            humidity: humidity,
            interval: interval
        },
        warnings: [],
        errors: []
    };
}
```

- Select “Payload formatters” and follow the steps.



Formatter type*
Custom Javascript formatter

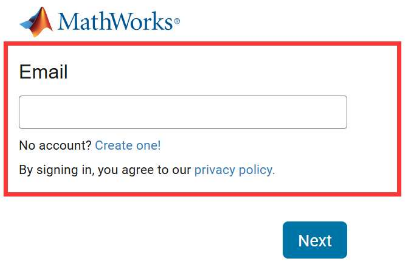
Formatter code*

```
// This file contains the uplink and downlink for ttn
// Uplink
function decodeUplink(input) {
  var bytes = input.bytes;

  var num = bytes[0] * 256 + bytes[1];
  var ADC_1 = (bytes[2] * 256 + bytes[3]) / 10.0;
  var ADC_2 = (bytes[4] * 256 + bytes[5]) / 10.0;
  var CT1_ADC = (bytes[6] * 256 + bytes[7]) / 10.0;//This voltage needs to be combined with the sensor's ratio;
  var CT2_ADC = (bytes[8] * 256 + bytes[9]) / 10.0;//This voltage needs to be combined with the sensor's ratio;

  var CT1 = CT1_ADC*60 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V, multiply by 300.
  var CT2 = CT2_ADC*60 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V, multiply by 300.

  var humidity = (bytes[10] * 256 + bytes[11]) / 10.0;
  var temperature = bytes[12] * 256 + bytes[13];
  if (temperature >= 0x8000) {
    temperature -= 0x10000;
  }
  temperature = temperature / 10.0;

  var interval = (bytes[14] * 16777216 +bytes[15] * 65536 + bytes[16] * 256 +bytes[17] ) / 1000;
  return {
    data: {
      ADC_1: ADC_1,//0-5V ADC
```

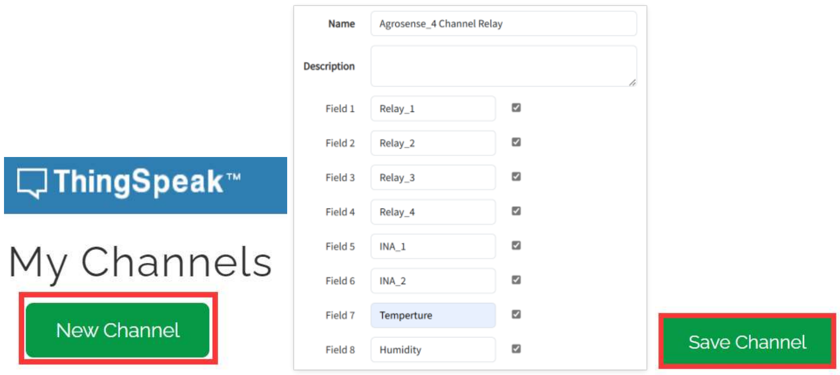
3.2.3 Application Server configuration

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)

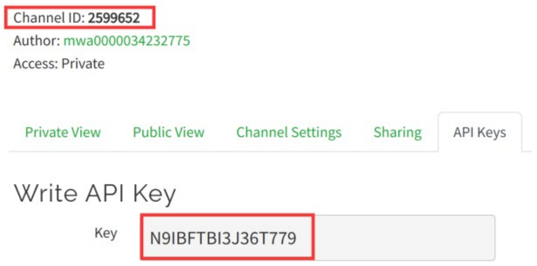


- Click "New Channel", fill in the Channel name and field names and click "Save Channel".



Name	Agrosense_4 Channel Relay
Description	
Field 1	Relay_1
Field 2	Relay_2
Field 3	Relay_3
Field 4	Relay_4
Field 5	INA_1
Field 6	INA_2
Field 7	Temperture
Field 8	Humidity

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

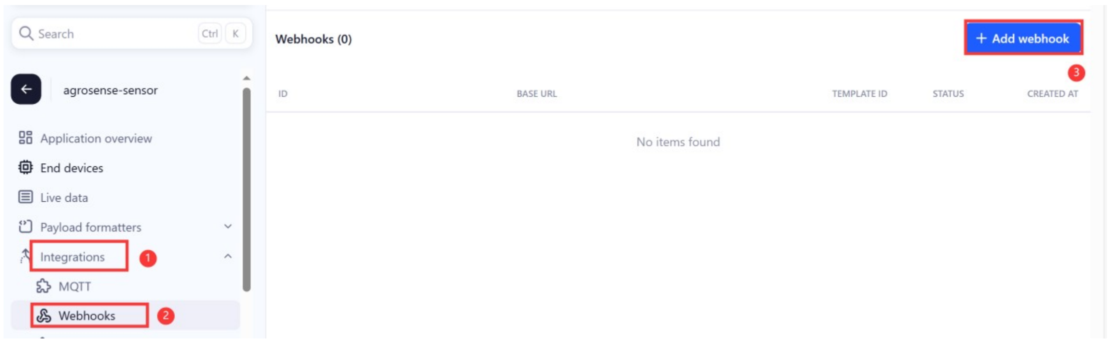


Channel ID: 2599652
Author: mwa0000034232775
Access: Private

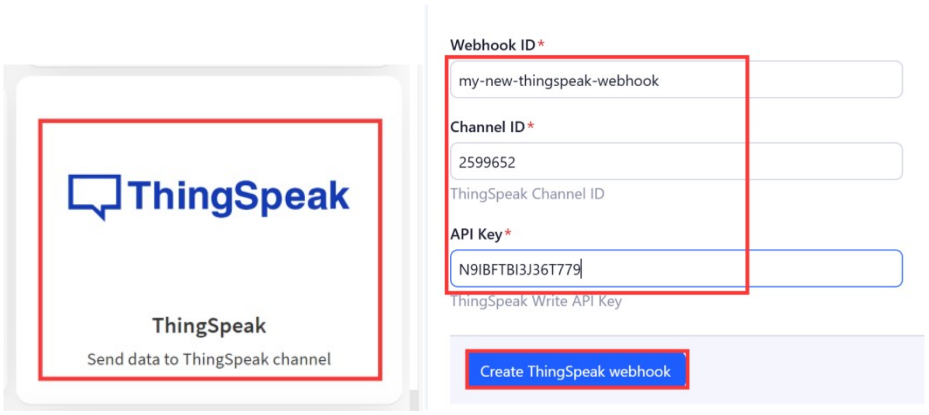
Write API Key
Key N9IBFTBI3J36T779

3.2.4 Connect the Network Server and Application Server

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

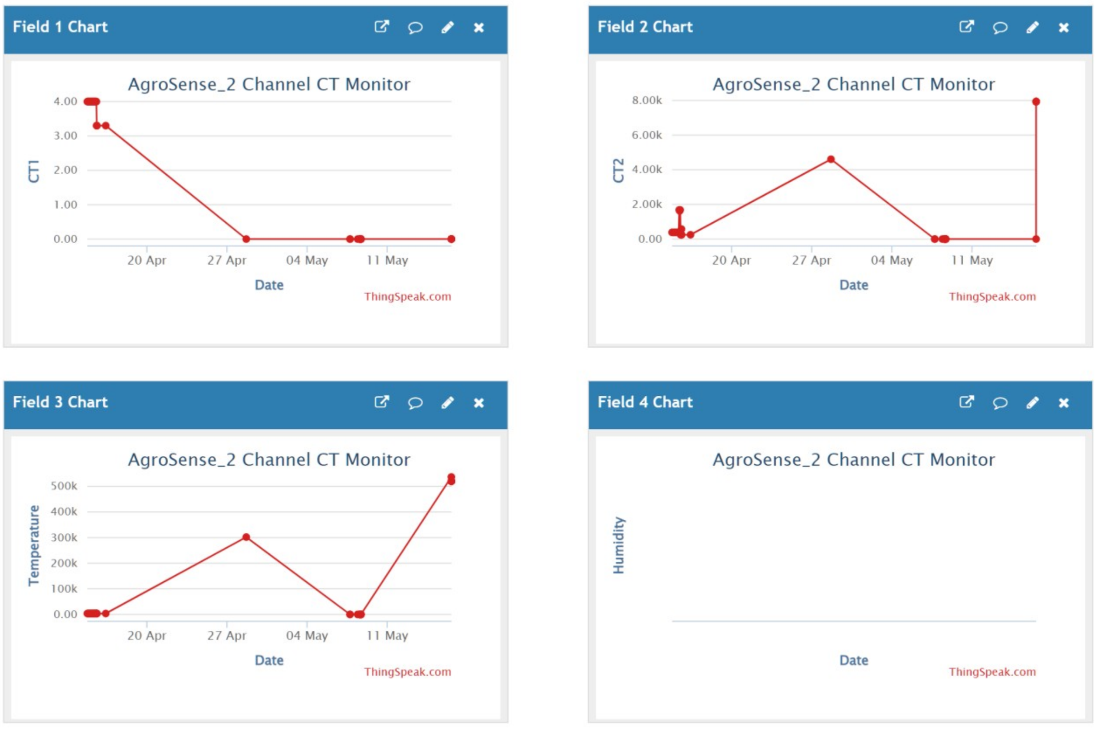


- Select "ThingSpeak", Fill in the Webhook ID and paste the Channel ID and API Key, click "Create ThingSpeak Webhook".



Webhook ID*
my-new-thingspeak-webhook
Channel ID*
2599652
ThingSpeak Channel ID
API Key*
N9IBFTBI3J36T779
ThingSpeak Write API Key

- 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)

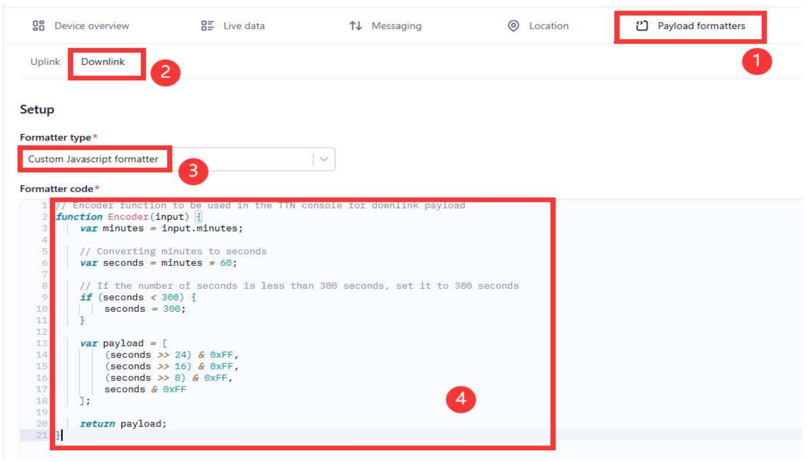
Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport5)

Modify the time interval :

1、 If you need to change time Interval (Default 1 minutes), you can click “Payload formatters-->Downlink” and follow the steps.

Formatter code you can find in Github.



Setup

Formatter type*

Custom Javascript formatter

Formatter code*

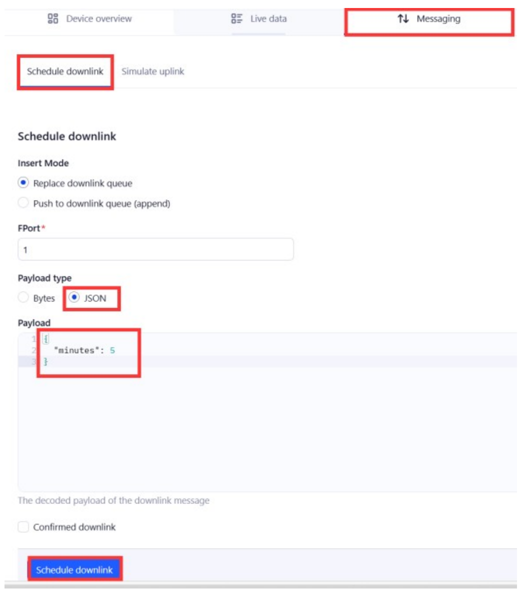
```
// Encoder function to be used in the TTN console for downlink payload
function Encoder(input) {
  var minutes = input.minutes;

  // Converting minutes to seconds
  var seconds = minutes * 60;

  // If the number of seconds is less than 300 seconds, set it to 300 seconds
  if (seconds < 300) {
    seconds = 300;
  }

  var payload = [
    (seconds >> 24) & 0xFF,
    (seconds >> 16) & 0xFF,
    (seconds >> 8) & 0xFF,
    seconds & 0xFF
  ];

  return payload;
}
```

2、Click “Save changes”.

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

Note: you must use this format:

```
{
   "minutes": 5
}
```



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

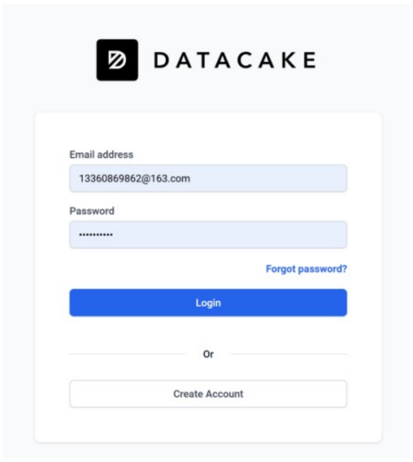
Schedule downlink

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

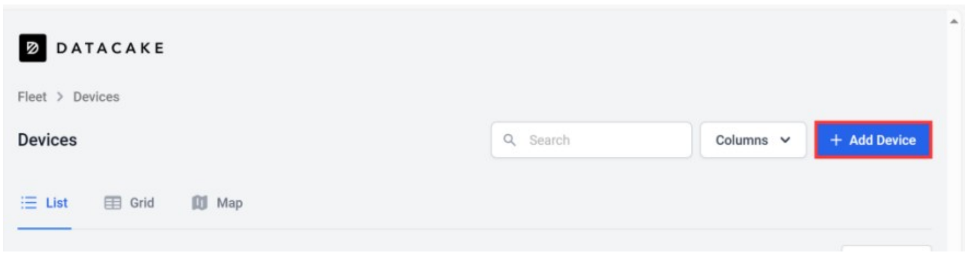
3.3 Usage with Datacake

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

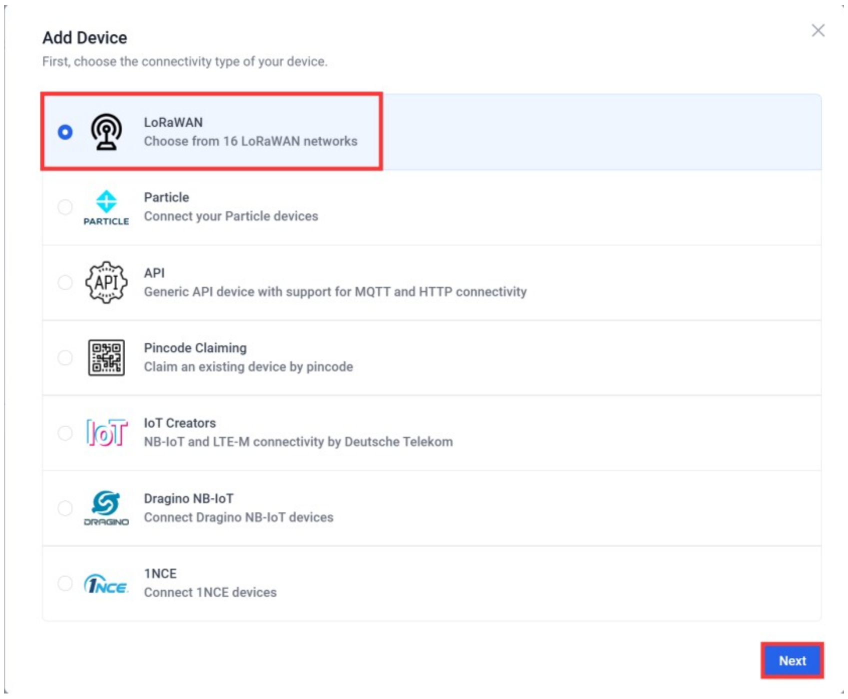
1、Login datacake or Create Account



2、Click “Add Device”



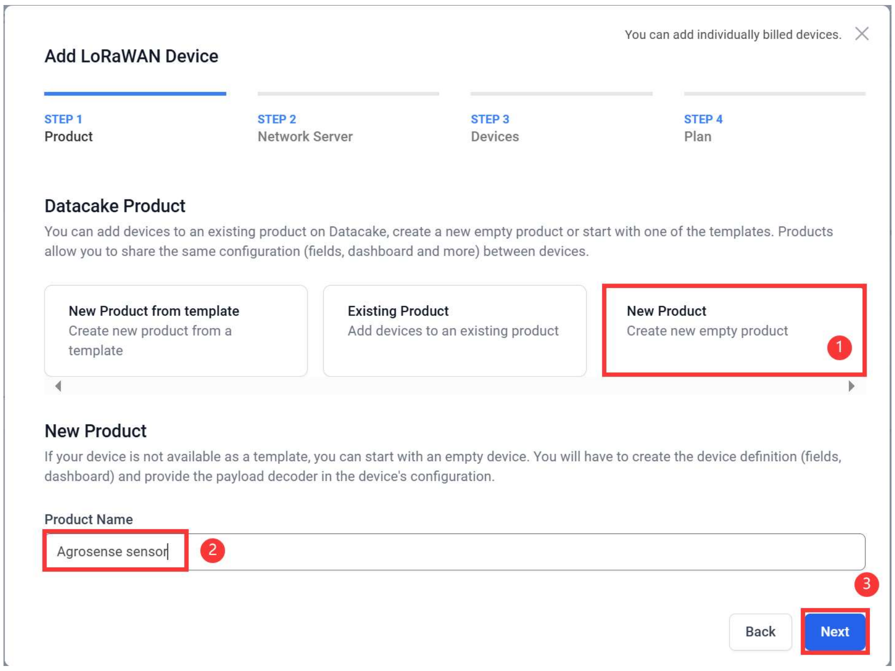
3、Select LoRaWAN and click “Next”


Add Device

First, choose the connectivity type of your device.

- LoRaWAN — Choose from 16 LoRaWAN networks
- Particle — Connect your Particle devices
- API — Generic API device with support for MQTT and HTTP connectivity
- Pincode Claiming — Claim an existing device by pincode
- IoT Creators — NB-IoT and LTE-M connectivity by Deutsche Telekom
- Dragino NB-IoT — Connect Dragino NB-IoT devices
- 1NCE — Connect 1NCE devices

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



Add LoRaWAN Device

You can add individually billed devices.

STEP 1 Product | STEP 2 Network Server | STEP 3 Devices | STEP 4 Plan

Datacake Product

You can add devices to an existing product on Datacake, create a new empty product or start with one of the templates. Products allow you to share the same configuration (fields, dashboard and more) between devices.

New Product from template — Create new product from a template

Existing Product — Add devices to an existing product

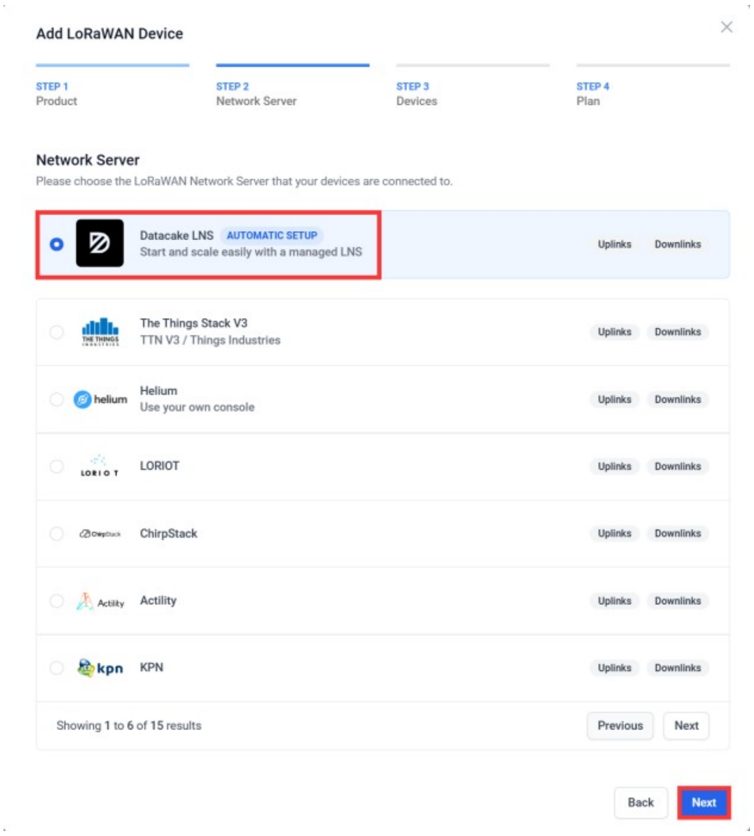
New Product — Create new empty product

New Product

If your device is not available as a template, you can start with an empty device. You will have to create the device definition (fields, dashboard) and provide the payload decoder in the device's configuration.

Product Name: Agrosense sensor

5、Select “Datacake LNS”



Add LoRaWAN Device

STEP 1 Product | STEP 2 Network Server | STEP 3 Devices | STEP 4 Plan

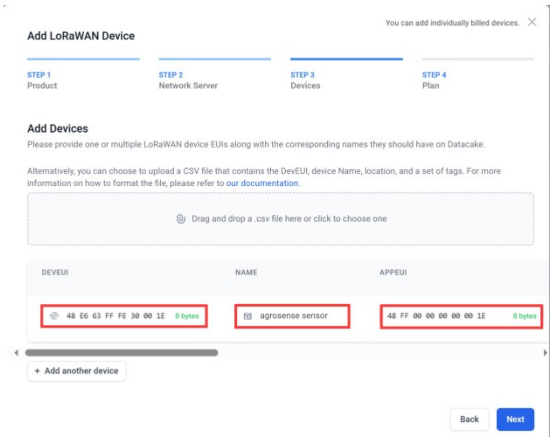
Network Server

Please choose the LoRaWAN Network Server that your devices are connected to.

Datacake LNS AUTOMATIC SETUP	Start and scale easily with a managed LNS	Uplinks Downlinks
The Things Stack V3	TTN V3 / Things Industries	Uplinks Downlinks
Helium	Use your own console	Uplinks Downlinks
LORIOT		Uplinks Downlinks
ChirpStack		Uplinks Downlinks
Actility		Uplinks Downlinks
KPN		Uplinks Downlinks

Showing 1 to 6 of 15 results

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



You can add individually billed devices.

Add LoRaWAN Device

STEP 1 Product · STEP 2 Network Server · STEP 3 Devices · STEP 4 Plan

Add Devices

Please provide one or multiple LoRaWAN device EUIs along with the corresponding names they should have on Datacake.

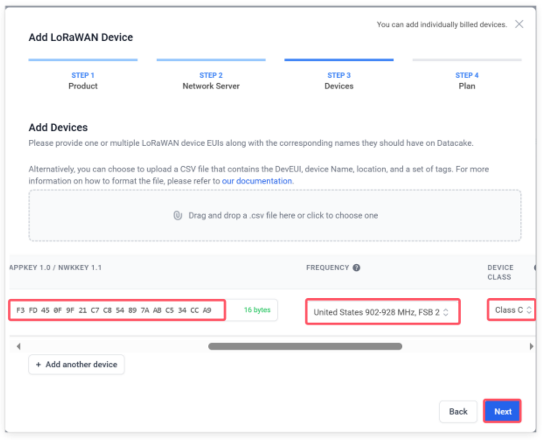
Alternatively, you can choose to upload a CSV file that contains the DevEUI, device Name, location, and a set of tags. For more information on how to format the file, please refer to our documentation.

Drag and drop a .csv file here or click to choose one

DEVEUI	NAME	APPEUI
48 E6 63 FF FE 30 00 1E 8 bytes	agrosense sensor	48 FF 00 00 00 00 00 1E 8 bytes

+ Add another device

Back Next



You can add individually billed devices.

Add LoRaWAN Device

STEP 1 Product · STEP 2 Network Server · STEP 3 Devices · STEP 4 Plan

Add Devices

Please provide one or multiple LoRaWAN device EUIs along with the corresponding names they should have on Datacake.

Alternatively, you can choose to upload a CSV file that contains the DevEUI, device Name, location, and a set of tags. For more information on how to format the file, please refer to our documentation.

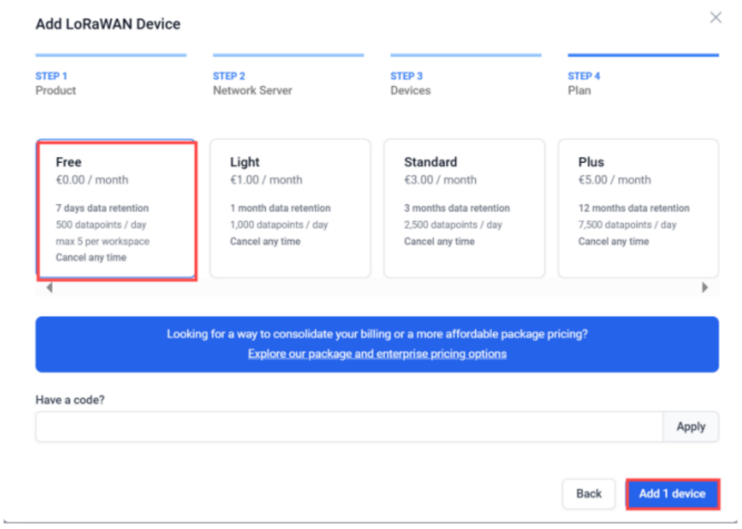
Drag and drop a .csv file here or click to choose one

APPKEY 1.0 / NWKKEY 1.1	FREQUENCY	DEVICE CLASS
F3 FD 45 0F 9F 21 C7 C8 54 89 7A AB C5 34 CC A9 16 bytes	United States 902-928 MHz, FSB 2	Class C

+ Add another device

Back Next

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



Add LoRaWAN Device

STEP 1 Product · STEP 2 Network Server · STEP 3 Devices · STEP 4 Plan

Free	Light	Standard	Plus
€0.00 / month	€1.00 / month	€3.00 / month	€5.00 / month
7 days data retention	1 month data retention	3 months data retention	12 months data retention
500 datapoints / day	1,000 datapoints / day	2,500 datapoints / day	7,500 datapoints / day
max 5 per workspace	Cancel any time	Cancel any time	Cancel any time
Cancel any time			

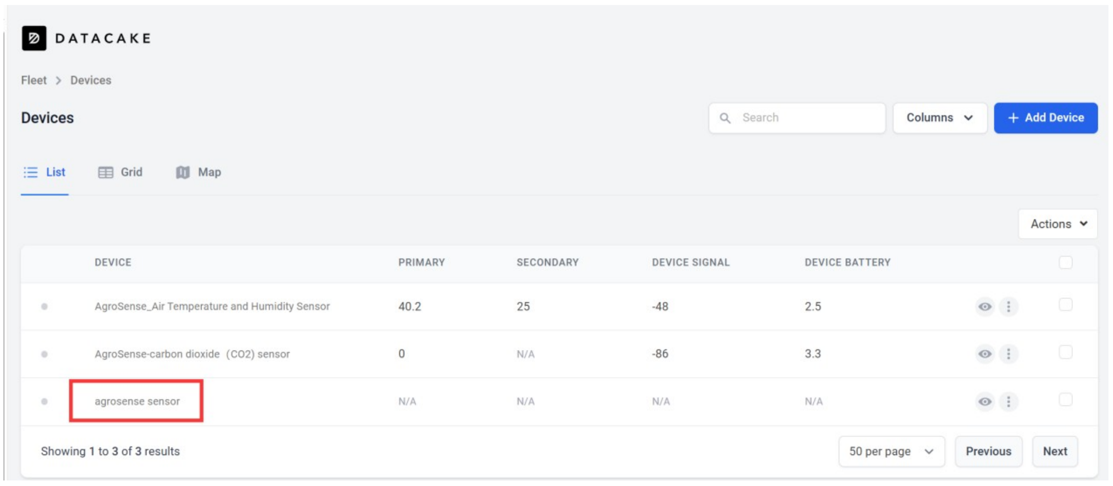
Looking for a way to consolidate your billing or a more affordable package pricing?
Explore our package and enterprise pricing options

Have a code?

Apply

Back Add 1 device

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



DATACAKE

Fleet > Devices

Devices

Search · Columns · + Add Device

List · Grid · Map

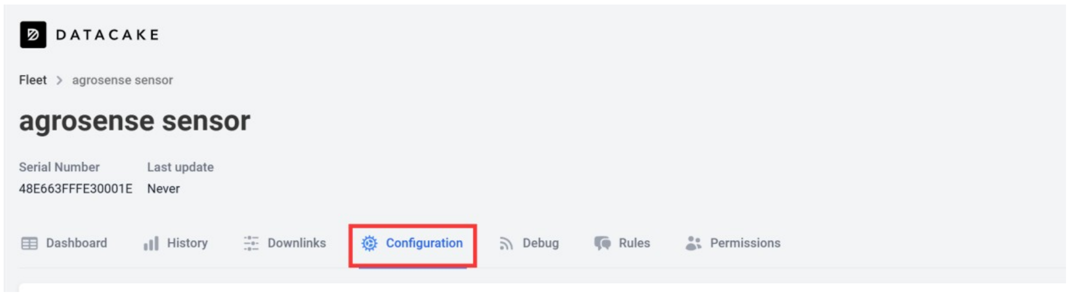
Actions

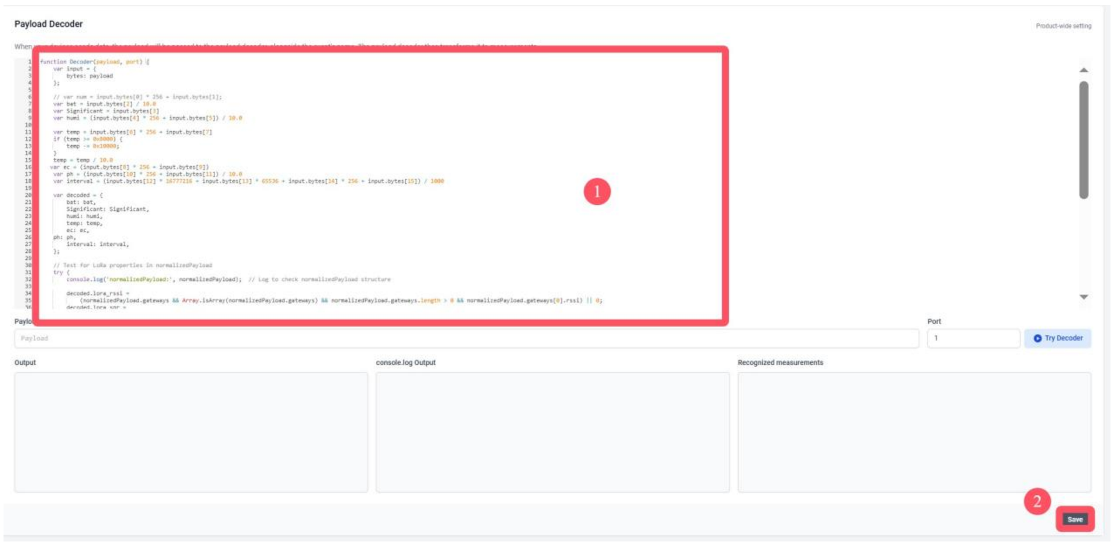
DEVICE	PRIMARY	SECONDARY	DEVICE SIGNAL	DEVICE BATTERY
AgroSense_Air Temperature and Humidity Sensor	40.2	25	-48	2.5
AgroSense-carbon dioxide (CO2) sensor	0	N/A	-86	3.3
agrosense sensor	N/A	N/A	N/A	N/A

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)



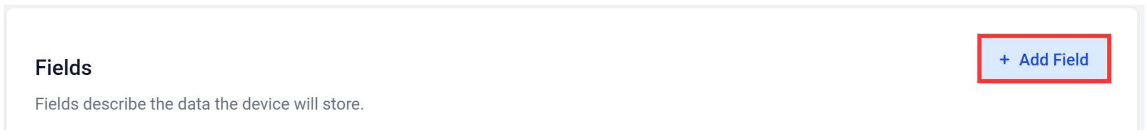


```
function Decoder(payload, port) {
    var input = {
        bytes: payload
    };

    // var num = input.bytes[0] * 256 + input.bytes[1];
    var bat = input.bytes[2] / 10.0
    var Significant = input.bytes[3]
    var humi = (input.bytes[4] * 256 + input.bytes[5]) / 10.0

    var temp = input.bytes[6] * 256 + input.bytes[7]
    if (temp >= 0x8000) {
        temp -= 0x10000;
    }
    temp = temp / 10.0
   var ec = (input.bytes[8] * 256 + input.bytes[9])
    var ph = (input.bytes[10] * 256 + input.bytes[11]) / 10.0
    var interval = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 + input.bytes[15]) / 1000

    var decoded = {
        bat: bat,
        Significant: Significant,
        humi: humi,
        temp: temp,
        ec: ec,
    ph: ph,
        interval: interval,
    };

    // Test for LoRa properties in normalizedPayload
    try {
        console.log('normalizedPayload:', normalizedPayload);  // Log to check normalizedPayload structure

        decoded.lora_rssi =
            (normalizedPayload.gateways && Array.isArray(normalizedPayload.gateways) && normalizedPayload.gateways.length > 0 && normalizedPayload.gateways[0].rssi) || 0;
        decoded.lora_snr =
```

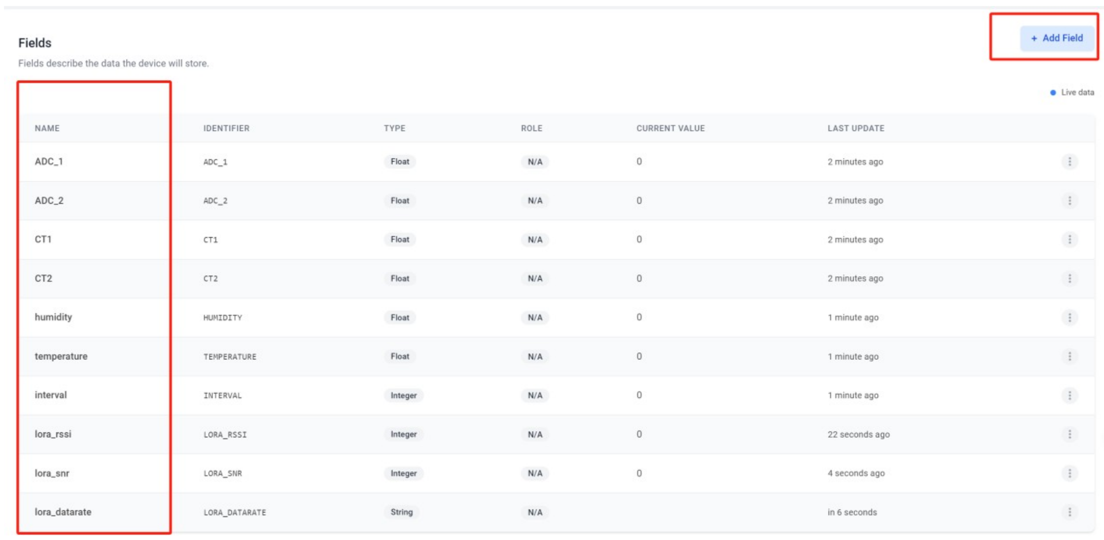
```
function Decoder(payload, port) {

        var input = { bytes: payload};

        var num = input.bytes[0] * 256 + input.bytes[1]

        var ADC_1 =    (input.bytes[2] * 256 + input.bytes[3]) / 10.0

        var ADC_2 =    (input.bytes[4] * 256 + input.bytes[5]) / 10.0

        var CT1_ADC = (input.bytes[6] * 256 + input.bytes[7]) / 1000.0      //This voltage needs to be combined with
the sensor's ratio

        var CT2_ADC = (input.bytes[8] * 256 + input.bytes[9]) / 1000.0      //This voltage needs to be combined with
the sensor's ratio

    var CT1 = CT1_ADC*100 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V,
multiply by 300.

    var CT2 = CT2_ADC*100 //if the sensor is 60A/1V, multiply by 60; if it's 100A/1V, multiply by 100; if it's 300A/1V,
multiply by 300.

    var humidity = (input.bytes[10] * 256 + input.bytes[11]) / 10.0;

    var temperature = input.bytes[12] * 256 + input.bytes[13];

    if (temperature >= 0x8000) {
      temperature -= 0x10000;
   }
   temperature = temperature / 10.0;
      var interval = (input.bytes[14]* 16777216 + input.bytes[15]* 65536 + input.bytes[16] * 256 + input.bytes[17])
/ 1000
      var decoded =
      {
            ADC_1:ADC_1,
            ADC_2:ADC_2,
            CT1:CT1,
            CT2:CT2,
             temperature:temperature,
             humidity:humidity,
             interval:interval,
      };
      // Test for LoRa properties in normalizedPayload
 try {
   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;
   }
   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: "ADC_1", value: decoded.ADC_1 },

    { field: "ADC_2", value: decoded.ADC_2 },

    { field: "CT1", value: decoded.CT1 },

    { field: "CT2", value: decoded.CT2 },

    { 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.


Fields

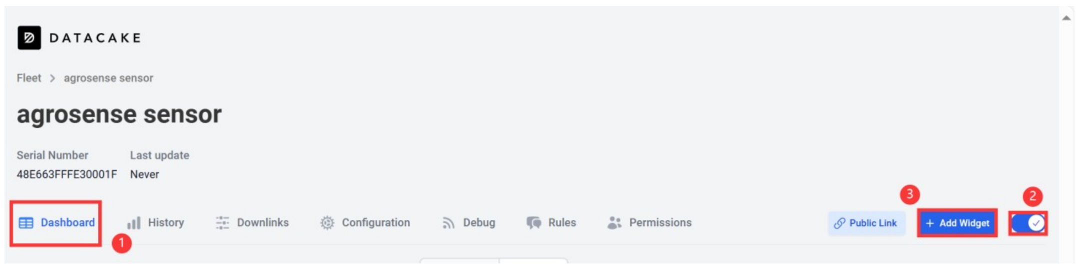
Fields describe the data the device will store.

NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
ADC_1	ADC_1	Float	N/A	0	2 minutes ago
ADC_2	ADC_2	Float	N/A	0	2 minutes ago
CT1	CT1	Float	N/A	0	2 minutes ago
CT2	CT2	Float	N/A	0	2 minutes ago
humidity	HUMIDITY	Float	N/A	0	1 minute ago
temperature	TEMPERATURE	Float	N/A	0	1 minute ago
interval	INTERVAL	Integer	N/A	0	1 minute ago
lora_rssi	LORA_RSSI	Integer	N/A	0	22 seconds ago
lora_snr	LORA_SNR	Integer	N/A	0	4 seconds ago
lora_datarate	LORA_DATARATE	String	N/A		in 6 seconds

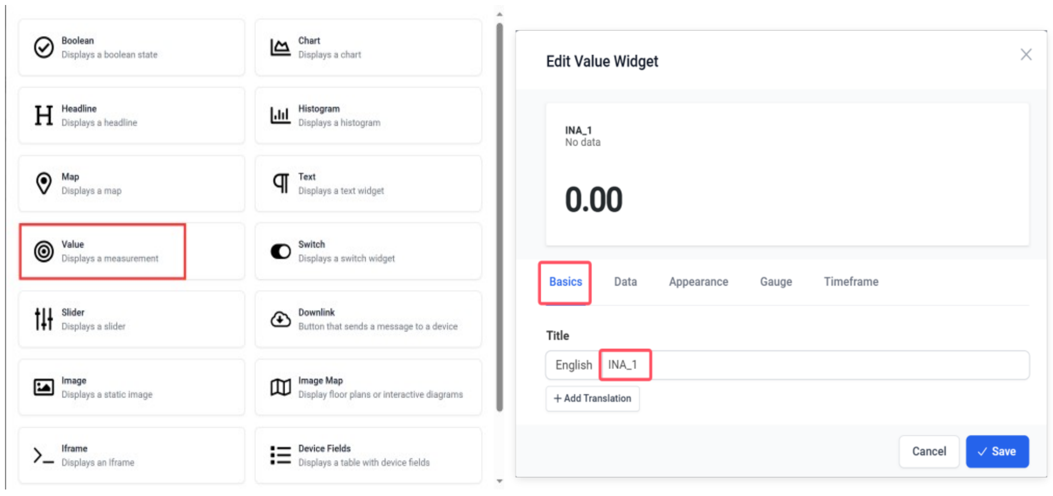
11、 Press RST button, wait until the sensor connects to the gateway successfully, you will see the data the sensor is currently reading.

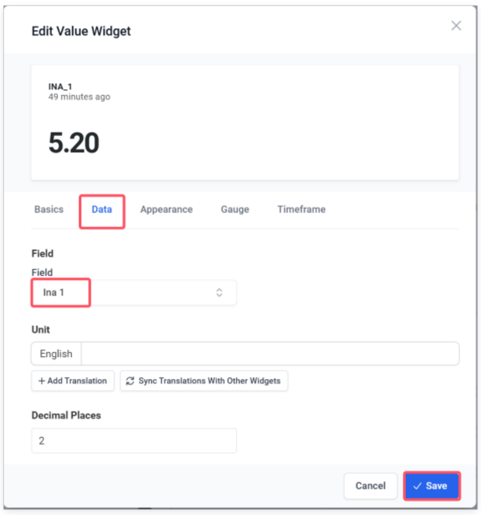
12、 To get a better look at the data, we can add widget.

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



13、Select “Value” and set Title, Field.ADC_1, ADC_2, CT1,CT2,Temperature, Humidityas the same way.





Edit Value Widget

INA_1
49 minutes ago

5.20

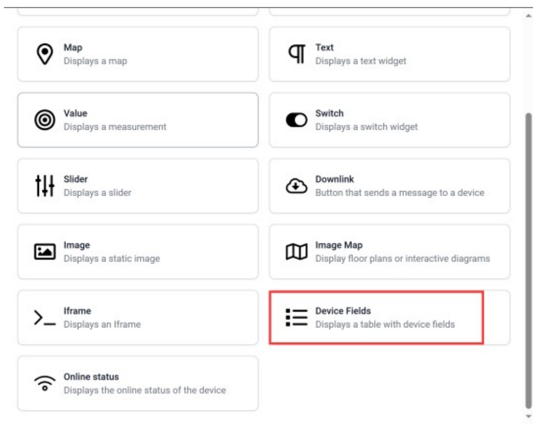
Field

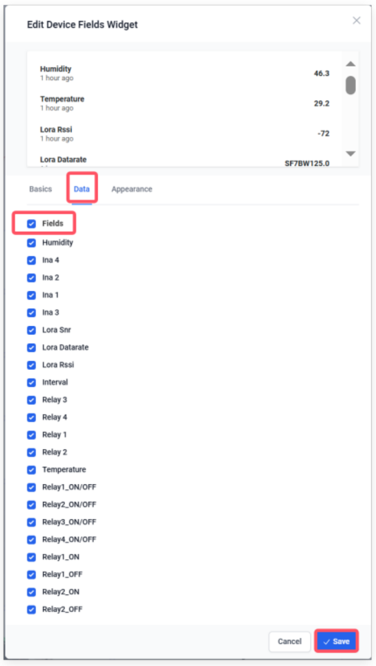
Field: Ina 1

Unit: English

Decimal Places: 2

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



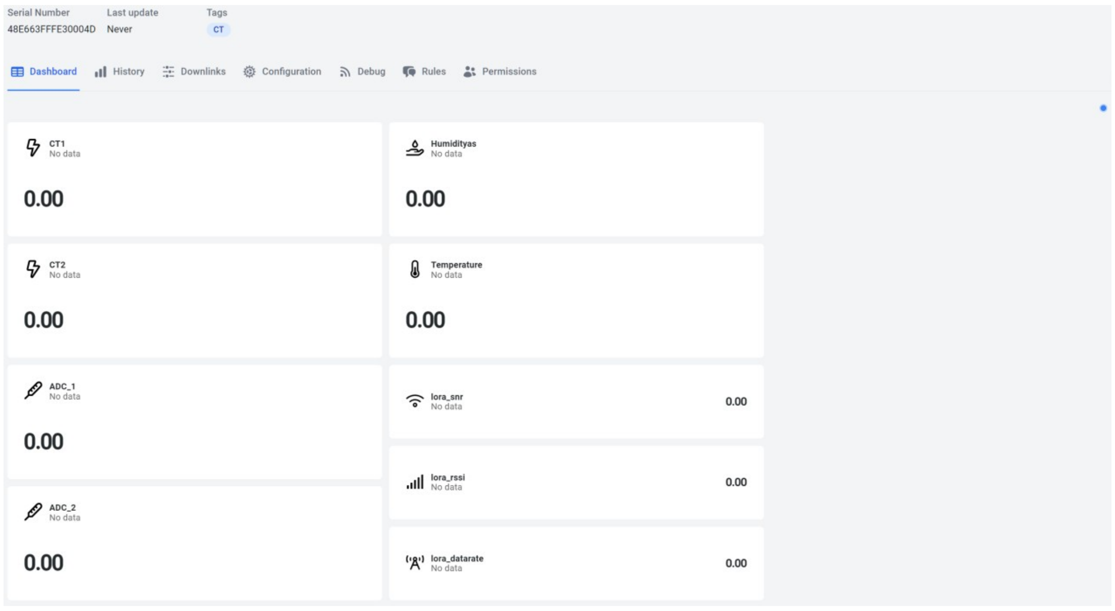


Edit Device Fields Widget

Humidity 1 hour ago	46.3
Temperature 1 hour ago	29.2
Lora Rssi 1 hour ago	-72
Lora Datarate	SF7BW125.0

- ✓ Fields
- ✓ Humidity
- ✓ Ina 4
- ✓ Ina 2
- ✓ Ina 1
- ✓ Ina 3
- ✓ Lora Snr
- ✓ Lora Datarate
- ✓ Lora Rssi
- ✓ Interval
- ✓ Relay 3
- ✓ Relay 4
- ✓ Relay 1
- ✓ Relay 2
- ✓ Temperature
- ✓ Relay1_ON/OFF
- ✓ Relay2_ON/OFF
- ✓ Relay3_ON/OFF
- ✓ Relay4_ON/OFF
- ✓ Relay1_ON
- ✓ Relay1_OFF
- ✓ Relay2_ON
- ✓ Relay2_OFF

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


Serial Number 48E663FFFE30004D · Last update Never · Tags CT

CT1 No data 0.00
Humidityas No data 0.00
CT2 No data 0.00
Temperature No data 0.00
ADC_1 No data 0.00
lora_snr No data 0.00
lora_rssi No data 0.00
ADC_2 No data 0.00
lora_datarate No data 0.00

3.3.1 Downlink

The downlink has the following functions:

- Modification time interva (Fport1)

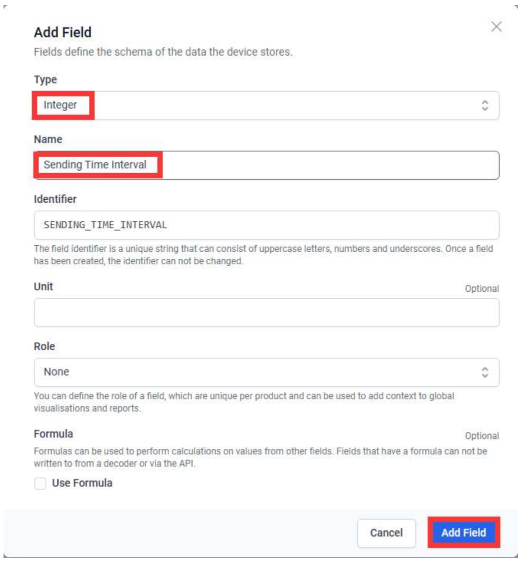
Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport5)

Users can view previous data based on this feature.

Modify the time interval :

1 、 If you need to change time Interval (Default 1 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

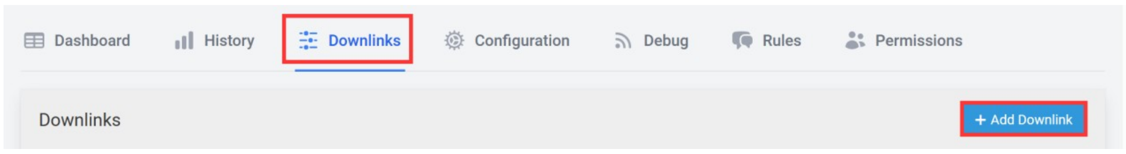
You can define the role of a field, which are unique per product and can be used to add context to global visualisations and reports.

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

2、Click ”Downlink-->Add Downlink”.

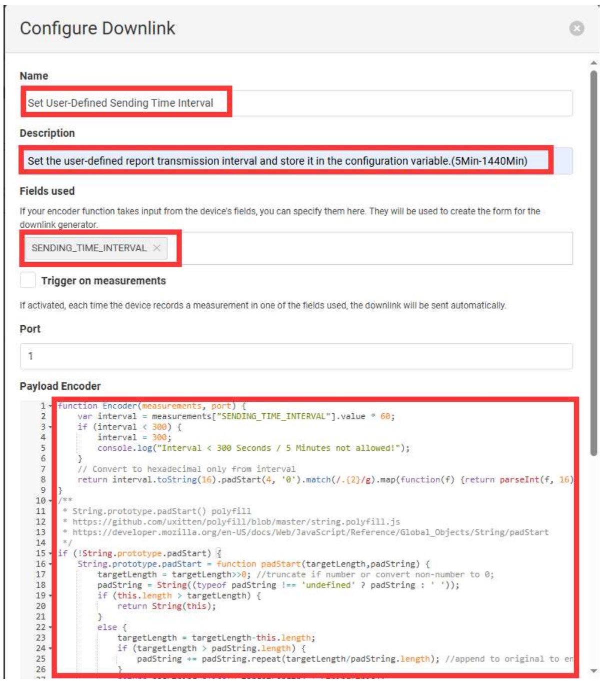


Enter name、 description、 fields used and payload encoder respectively.

Name: Set User-Defined Sending Time Interval

Description: Set the user-defined report transmission interval and store it in the configuration variable.(10S -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

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

Port

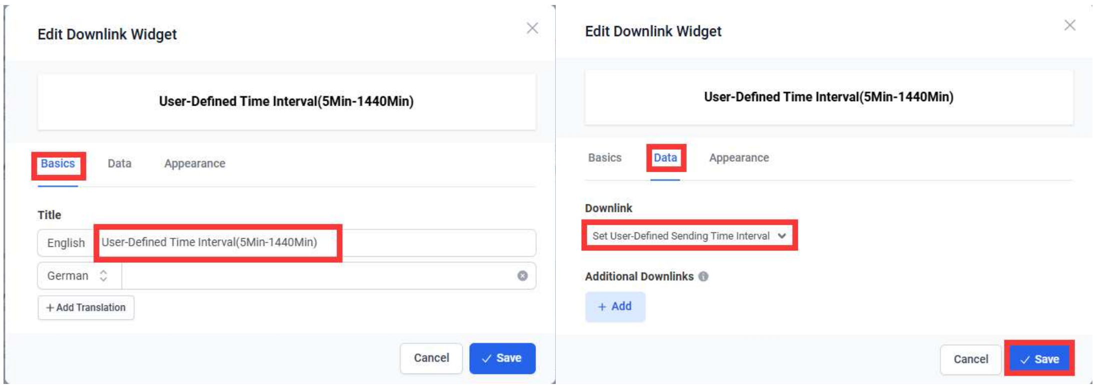
1

Payload Encoder

```
function Encoder(measurements, port) {
    var interval = measurements["SENDING_TIME_INTERVAL"].value * 60;
    if (interval < 300) {
        interval = 300;
        console.log("Interval < 300 Seconds / 5 Minutes not allowed!");
    }
    // Convert to hexadecimal only from interval
    return interval.toString(16).padStart(4, '0').match(/.{2}/g).map(function(f) {return parseInt(f, 16)
}
/**
 * String.prototype.padStart() polyfill
 * https://github.com/uxitten/polyfill/blob/master/string.polyfill.js
 * https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/padStart
 */
if (!String.prototype.padStart) {
    String.prototype.padStart = function padStart(targetLength,padString) {
        targetLength = targetLength>>0; //truncate if number or convert non-number to 0;
        padString = String((typeof padString !== 'undefined' ? padString : ' '));
        if (this.length > targetLength) {
            return String(this);
        }
        else {
            targetLength = targetLength-this.length;
            if (targetLength > padString.length) {
                padString += padString.repeat(targetLength/padString.length); //append to original to en
            }
```

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

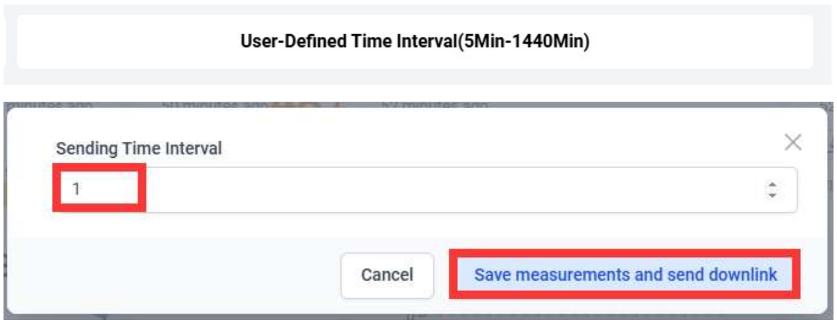
Downlink: Set User-Defined Sending Time Interval

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