AgroSense LoRaWAN®

4-Channel ADC 12 bits Manual

V1.0

Author: Yuki

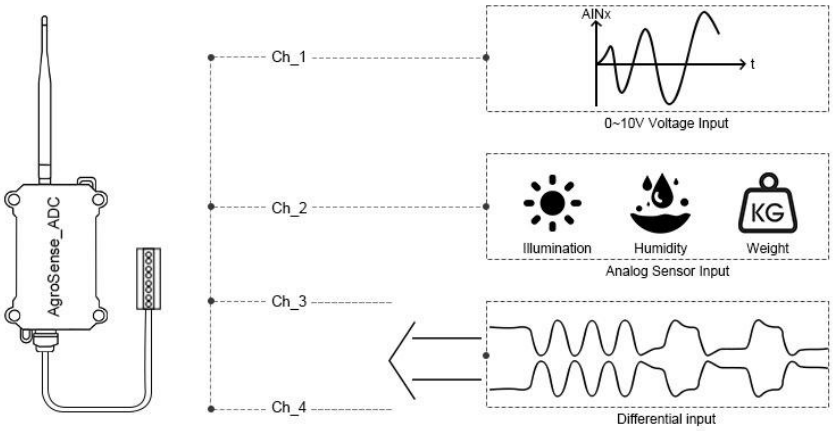
Time: 2025.01.18

Contents

1 Product Description

1.1 Introduction

AgroSense LoRaWAN® 4-Channel ADC 12 bits use ADS1015IDGSR chip, which is capable of connecting up to 4-channel devices and provides high-precision analogue-to-digital conversions (ADCs) of up to 12 bits. Also, the product has been rigorously tested for IP68 waterproof performance to ensure reliability and durability in a variety of harsh environments.The device supports voltage inputs, either directly or from other sensors, for a wide range of IoT applications. The built-in low-power 12-bit delta-sigma analogue-to-digital converter (ADC) provides a high degree of accuracy and stability. In addition, the 3/4 input channels of the AgroSense 4-Channel ADC can be configured in differential input mode, which greatly enhances its immunity to interference and ensures stable operation in complex electrical environments.



The sensor benefits from LoRaWAN , which ensures stability and reliability. It is capable of covering a long transmission range while maintaining low power consumption. Unlike wireline devices, it is battery-powered, reducing the workload and complexity of deployment, design and development for end-users that can work via powering it, and setting the configuration in the cloud server, for LoRaWAN® remote monitoring. It monitors the environmental data users need and report every 1 hour, with downlink users can manually adjust the reporting time.

1.2 Feature

- Includes a **high precision** sensor.
- Compatible with Worldwide **LoRaWAN® Networks**: Support the universal frequency bands EU868/ US915.
- LoRaWAN version: LoRaWAN Specification **1.0.3**.
- **Long Range**: Up to 2 kilometers in the city, up to 10 kilometers in the wilderness, receive sensitivity -137dBm , transmit power up to 21dBm.
- **Ultra-low power** consumption design, traditional AAA alkaline dry battery can be used for one year.
- **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.
- Suitable for **Harsh Environments**: Can work normally under the temperature of -40℃ ~ 85℃, IP68 waterproof, suitable for outdoor use in harsh conditions, high UV, dusty, heavy rain and other bad weather.
- Monitor data and upload **real-time** data regularly.
- Modify the product parameters through **AT commands**.
- Support **downlink** to modify the time interval, motion status on/off, motion status sensitivity.

1.3 Parameter

1. General Parameters

Product Model	AGLWMA01
Measurement Range	12-bit ADC
Measurement Accuracy	1/4096
channel count	4

2.Wireless Parameters

Communication Protocol	Standard LoRaWAN® protocol V1.0.3
Network Access/Operating Mode	OTAA Class A
MAX Transmit Power	21dBm

Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

3.Physical Parameters

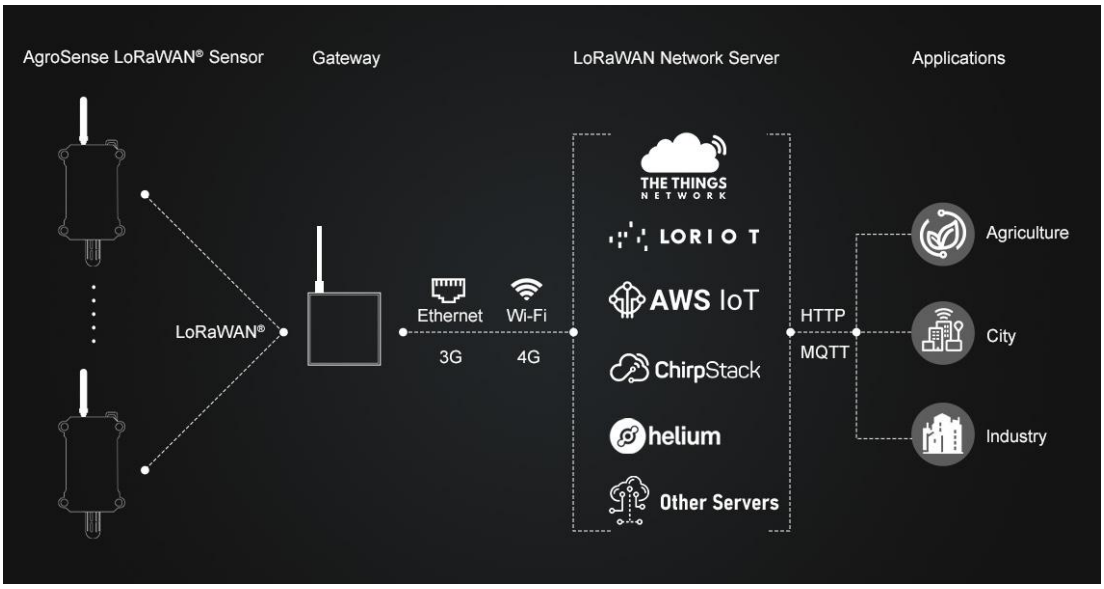
Power Supply	1 x 18650 3.7V Lion batteries
Operating Temperature	-40°C ~85°C
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

2 Technical route

2.1 System Framework

AgroSense 4-Channel ADC 12 bits Sensor 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.



The steps to achieve the detection of ADC is:

1. Collect the ADC data by sensor, and send the data from End Node to Gateway.
2. The Gateway packages node data and transmits it to the Network Server.
3. The Network Server decodes the data and sends it to the Applications.
4. Finally, user can monitor the ADC in the APP.

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

We use The Things Network as our Network Server, we need to configuration the country/ area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak or Datacake.

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 LoRaWAN® 4-Channel ADC 12 bits. (The AgroSense series is applicable)
- Network Server: The Things Network. (Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

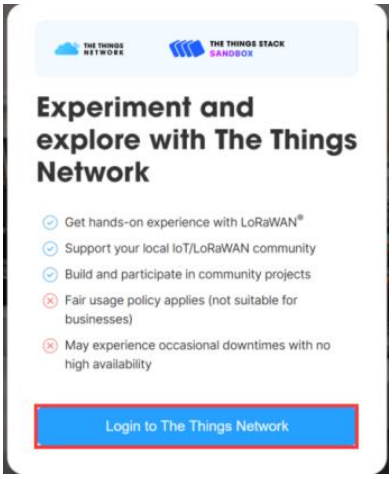
We will use ADC1 as an example to collect the voltage of an adjustable resistor.

Agrosense	ADC model
IN1	OUT
3V3	VCC
GND	GND

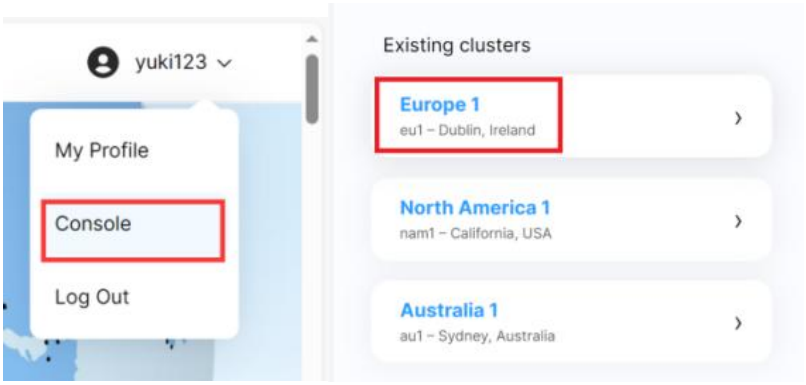
3.1 TTN and ThingSpeak

3.1.1Network Server configuration

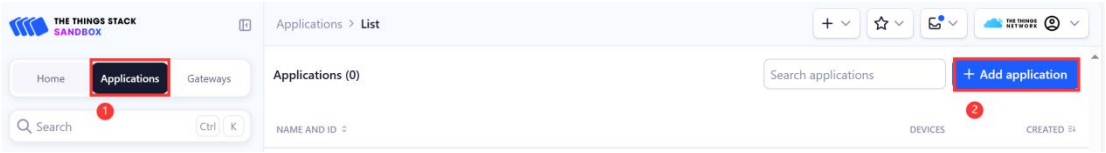
- Open The Things Network in your browser and login it. (Or register an account)



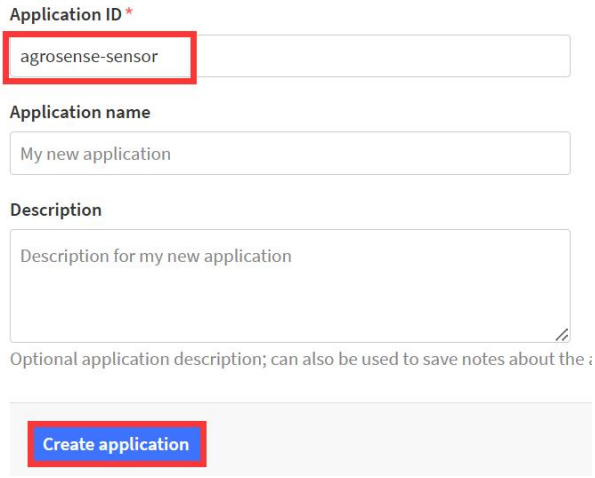
- Click “Console” and select clusters. (we take the European region for example.)



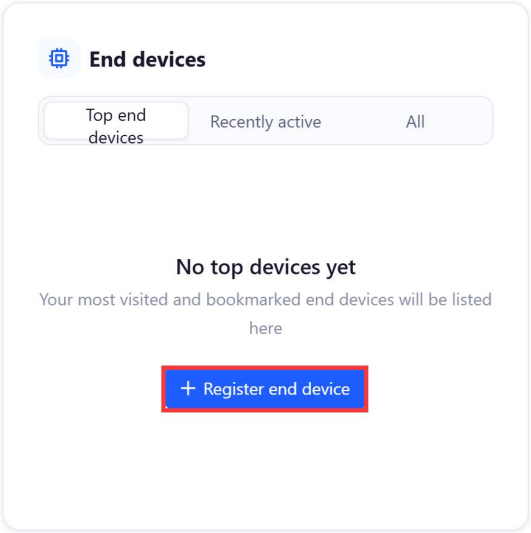
- Click “Go to applications” --> “+ Create application”.



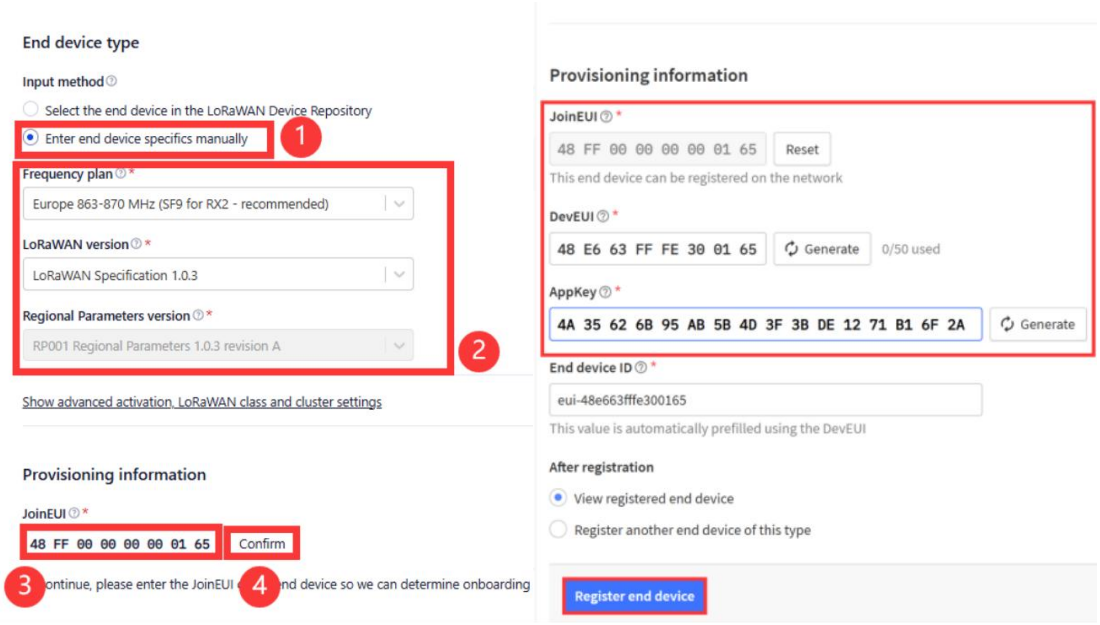
- Write the Application ID and click “Create application”.



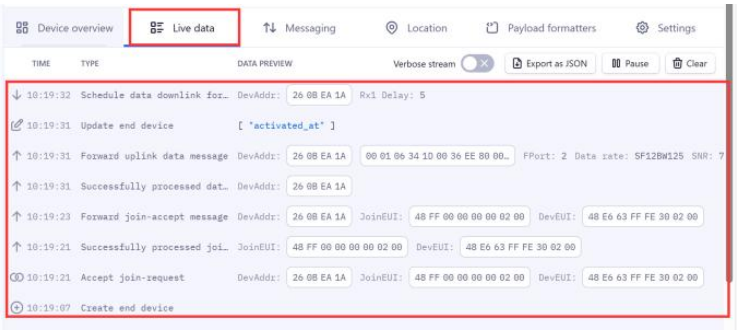
- Click “+ Register and device”.



- Fllowing the steps, 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.

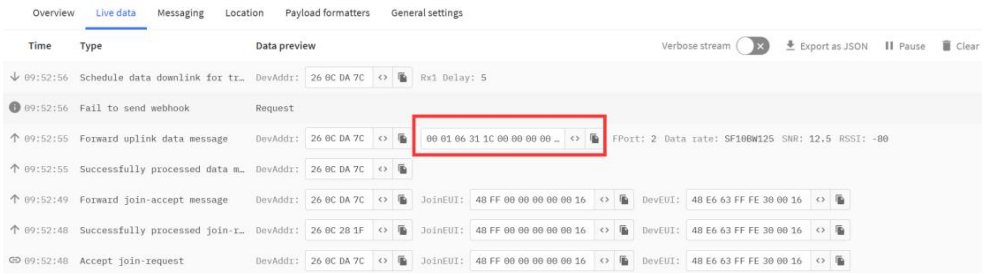


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



3.1.2 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	Battery voltage		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 battery 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 battery voltage is 3.3V.
byte 3	ADC_0 bits 8 to 15		ADC value in millivolts For example, if the value is 0x0CE4 = 3300, then the value is 3300mV.
byte 4	ADC_0 bits 0 to 7		
byte 5	ADC_1 bits 8 to 15		ADC value in millivolts

byte 6	ADC_1 bits 0 to 7		For example, if the value is 0x0CE4 = 3300, then the value is 3300mV.
byte 7	ADC_2 bits 8 to 15		ADC value in millivolts For example, if the value is 0x0CE4 = 3300, then the value is 3300mV.
byte 8	ADC_2 bits 0 to 7		
byte 9	ADC_3 bits 8 to 15		ADC value in millivolts For example, if the value is 0x0CE4 = 3300, then the value is 3300mV.
byte 10	ADC_3 bits 0 to 7		
byte 11	ADC_Differentialbits 6 to 15		ADC value in millivolts For example, if the value is 0x0CE4 = 3300, then the value is 3300mV.
byte 12	ADC_Differentialbits 0 to 7		
byte 13	data transmission interval bits 24 to 31	0-0xFFFFFFFF	The time interval for data transmission has been increased by a factor of 1000. The unit is seconds.
byte 14	data transmission interval bits 16 to 23		
byte 15	data transmission interval bits 8 to 15		
byte 16	data transmission interval bits 0 to 7		
byte 17	The data validity flag	0/1	0 is invalid, 1 is valid.
Fport 1	Change the data sending interval		
Fport 5	Upload the quantity of the latest local logged data		
Fport 6	Turn on and off the corresponding ADC channels		
Fport 7	Set the third and fourth channels to differential input		0 is enable ADC3 and ADC4 , 1 is enable Differentialbits

Example: 0x00, 0x01, 0x28, 0x0C, 0xE4, 0x0C, 0xC4, 0x0A, 0xE4, 0x0A, 0xAA, 0x0B, 0xB4, 0x00, 0x36, 0xEE, 0x80, 0x01

Data parsing：

Battery voltage is 4V.

ADC_0 is 3300mV.

ADC_1 is 3268mV.

ADC_2 is 2788mV.

ADC_3 is 2730mV.

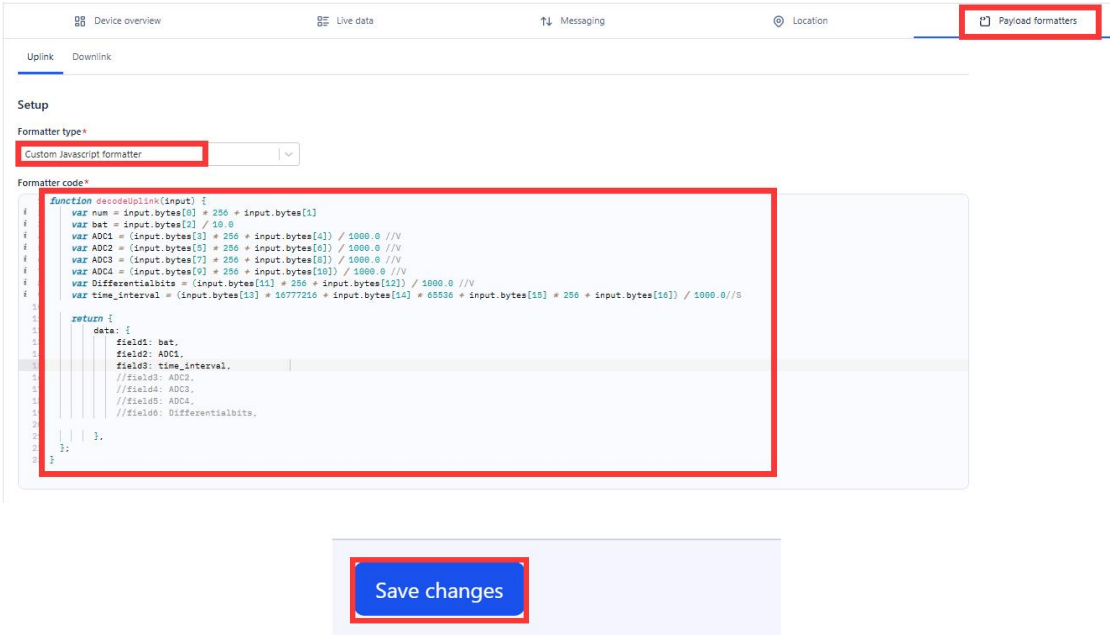
ADC_Differentialbits is 2996mV.

Data transmission interval value is 3600s.

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

```
function decodeUplink(input) {
    var num = input.bytes[0] * 256 + input.bytes[1]
    var bat = input.bytes[2] / 10.0
    var ADC1 = (input.bytes[3] * 256 + input.bytes[4]) / 1000.0 //V
    var ADC2 = (input.bytes[5] * 256 + input.bytes[6]) / 1000.0 //V
    var ADC3 = (input.bytes[7] * 256 + input.bytes[8]) / 1000.0 //V
    var ADC4 = (input.bytes[9] * 256 + input.bytes[10]) / 1000.0 //V
    var Differentialbits = (input.bytes[11] * 256 + input.bytes[12]) / 1000.0 //V
    var time_interval = (input.bytes[13] * 16777216 + input.bytes[14] * 65536 + input.bytes[15] * 256 + input.bytes[16]) / 1000.0//S

    return {
        data: {
            field1: bat,
            field2: ADC1,
            Field3: time_interval,
        },
  };
}
```

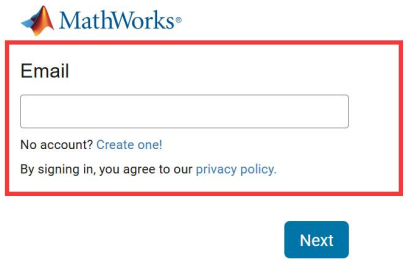
- Select “Payload formatters” and follow the steps.



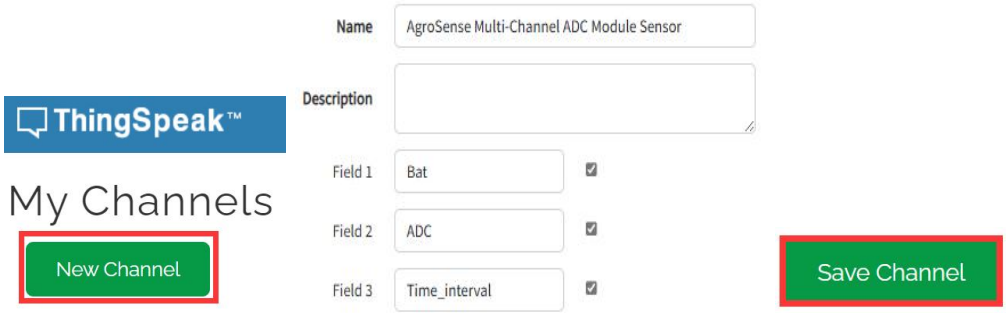
3.1.3 Application Server configuration

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.

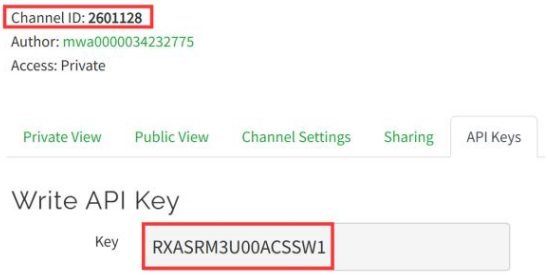
- Login to the ThingSpeak. (Or register an account)



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

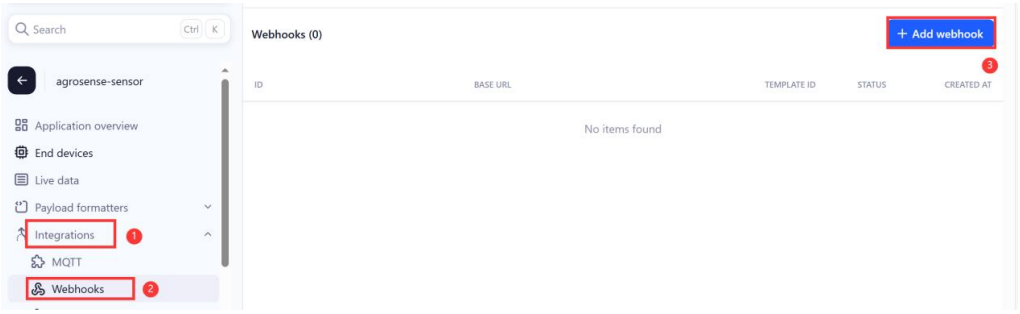


- After successful creation, copy the Channel ID and API Key.

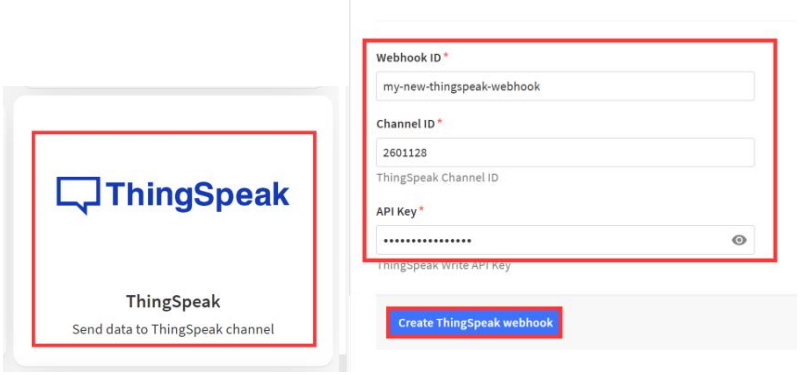


3.1.4 Connect the Network Server and Application Server

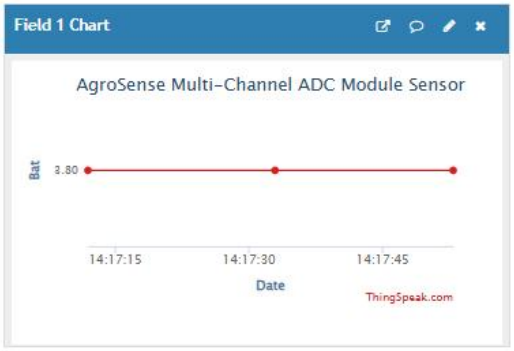
- In the TTN, click “integrations” --> “Webhooks” --> “+ Add webhook”.

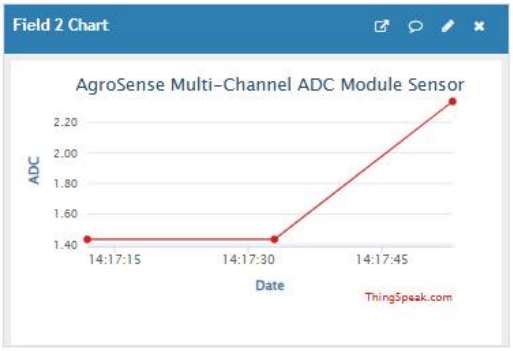


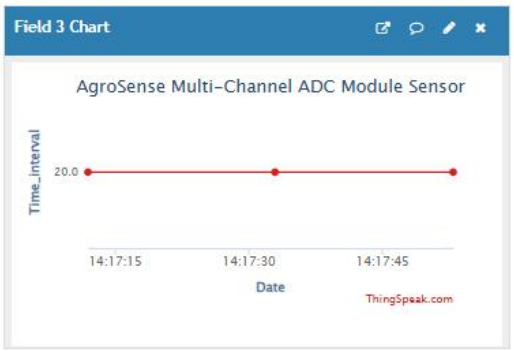
- Select “ThingSpeak”, Fill in the Webhook ID and paste the Channel ID and API Key, click “Create ThingSpeak Webhook”.



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







3.1.5 Downlink

The downlink has four 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.

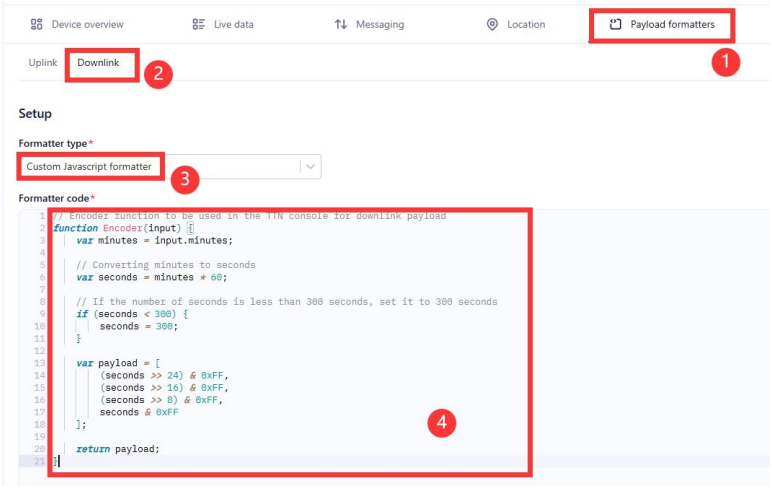
- Turn on or off the corresponding ADC channel (Fport6, the default is all on);

Differentialbits	ADC4	ADC3	ADC2	ADC1	Note
0	0	0	0	1	0x01 =enable the ADC1 channel
0	0	0	1	0	0x02 =enable the ADC2 channel
0	0	1	0	0	0x04 =enable the ADC3 channel
0	1	0	0	0	0x08 =enable the ADC4 channel
1	0	0	0	0	0x10 =enable the Differentialbits channel, At this point, FPort7 must be set to 1
0	0	0	0	0	0x00 =Turn off all channels
1	1	1	1	1	0x1F =enable all channels

- Set the third and fourth channels as differential inputs (Fport7)

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.

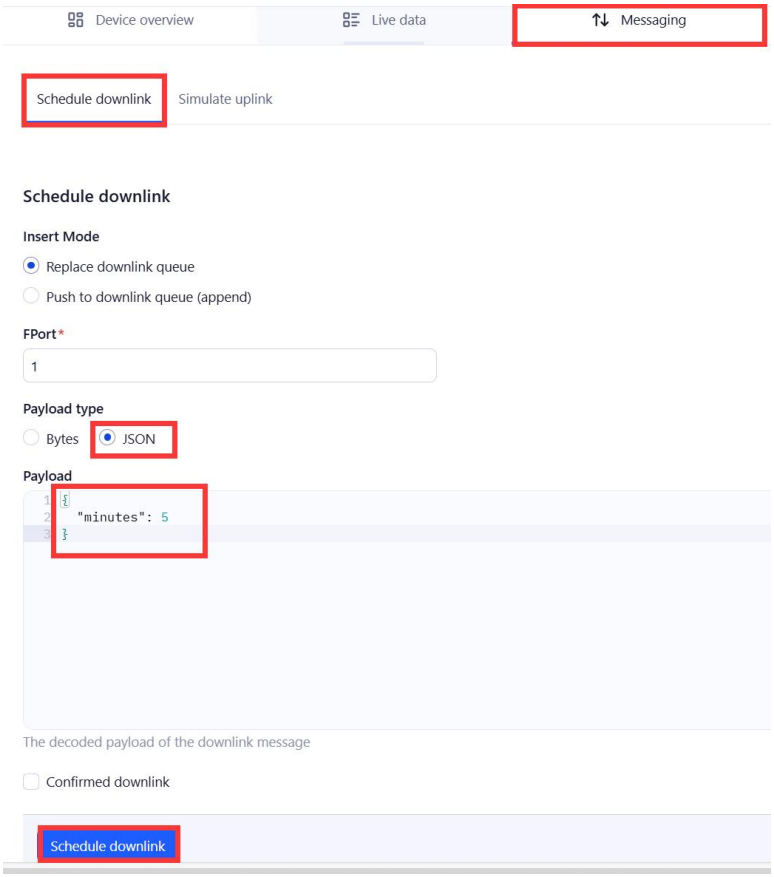


2、Click "Save changes".

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

Note: you must use this format:

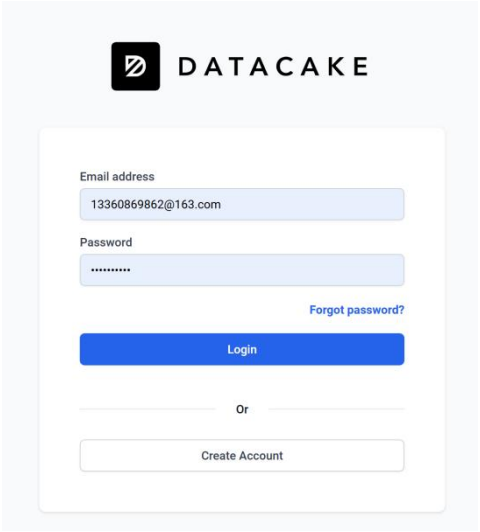
```
{
   "minutes": 5
}
```



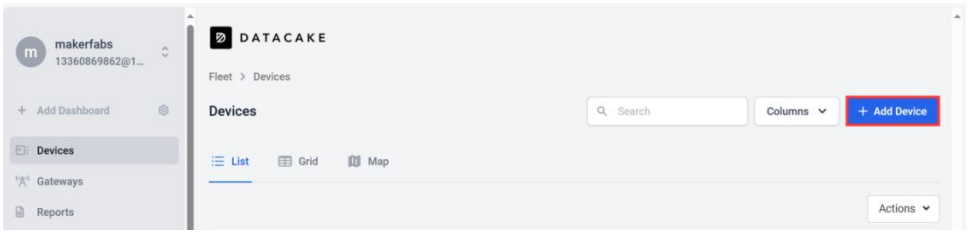
4、The modified interval will be updated after the next data upload.

3.2 Datacake

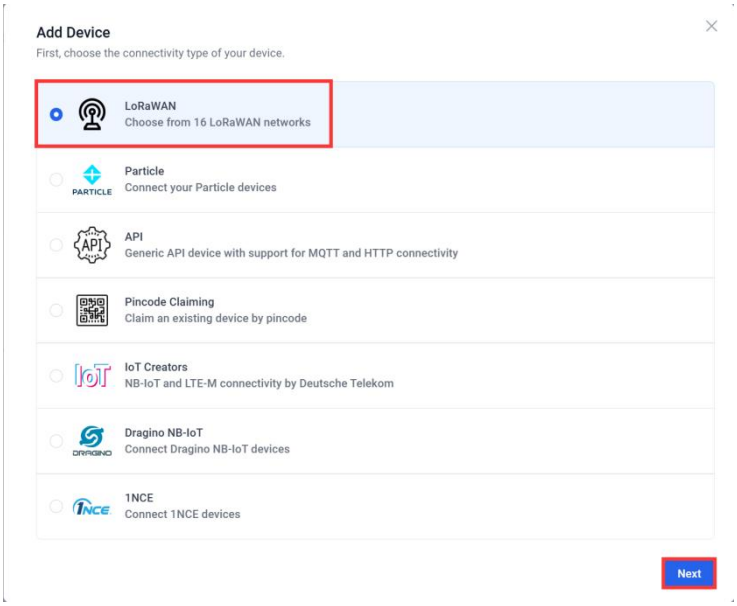
1、Login datacake or Create Account



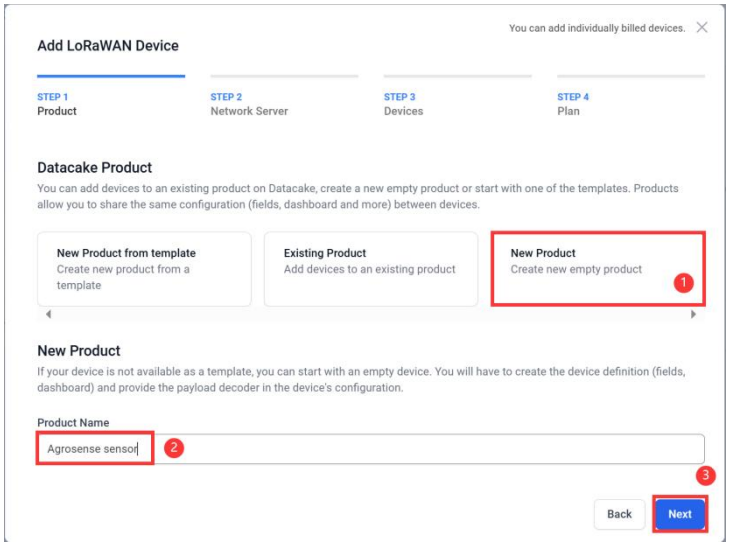
2、Click “Add Device”



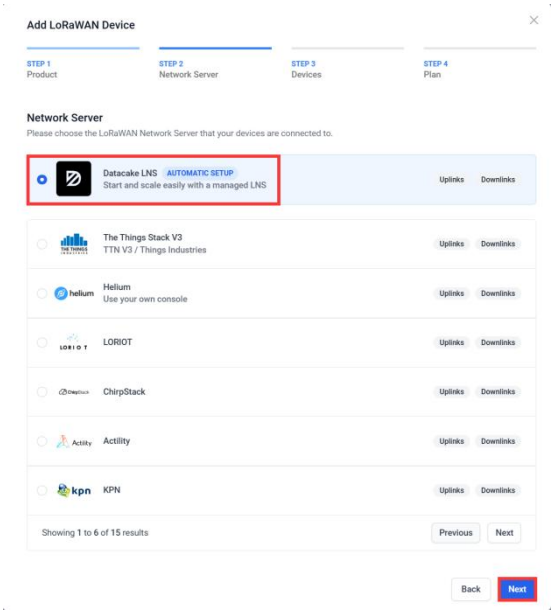
3、Select LoRaWAN and click “Next”



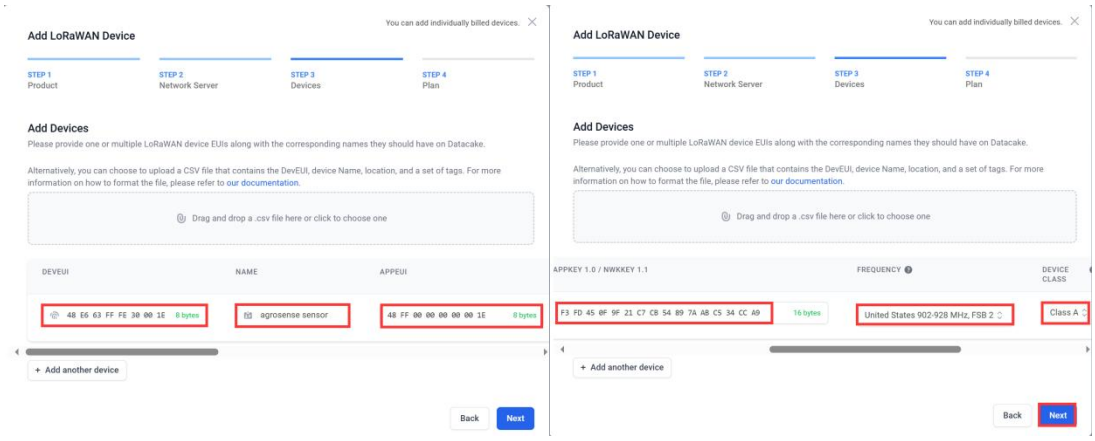
4、Select a Product based on your needs, take "Create new empty product" as an example.



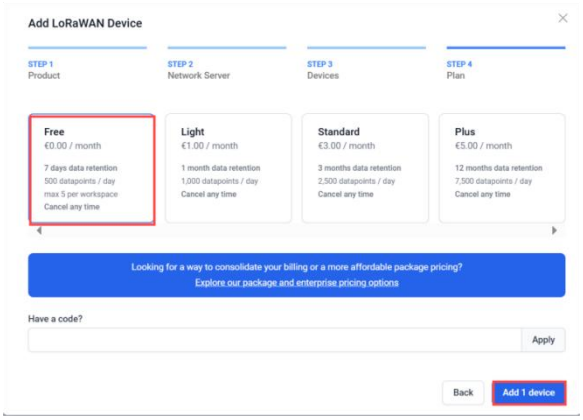
5、Select “Datacake LNS”



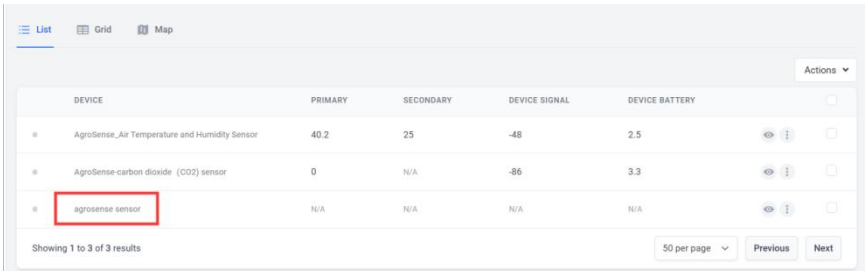
6、Enter DEVEUI、APPEUI、APPKEY、FREQUENCY(take 915 for example) and DEVICE CLASS.



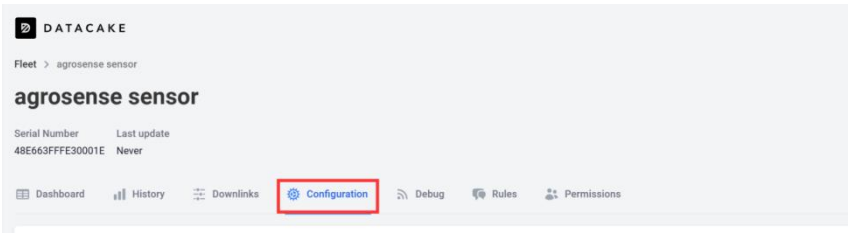
7、Choose the type according to your needs, and click “Add 1 device”.

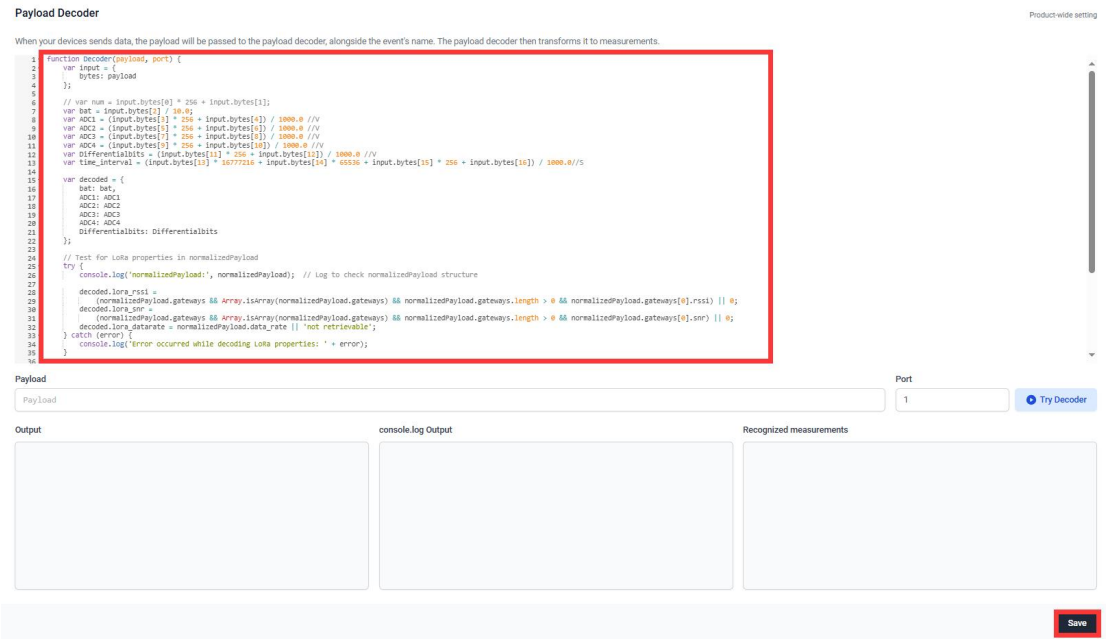


8、Click to go to the device you just added.



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

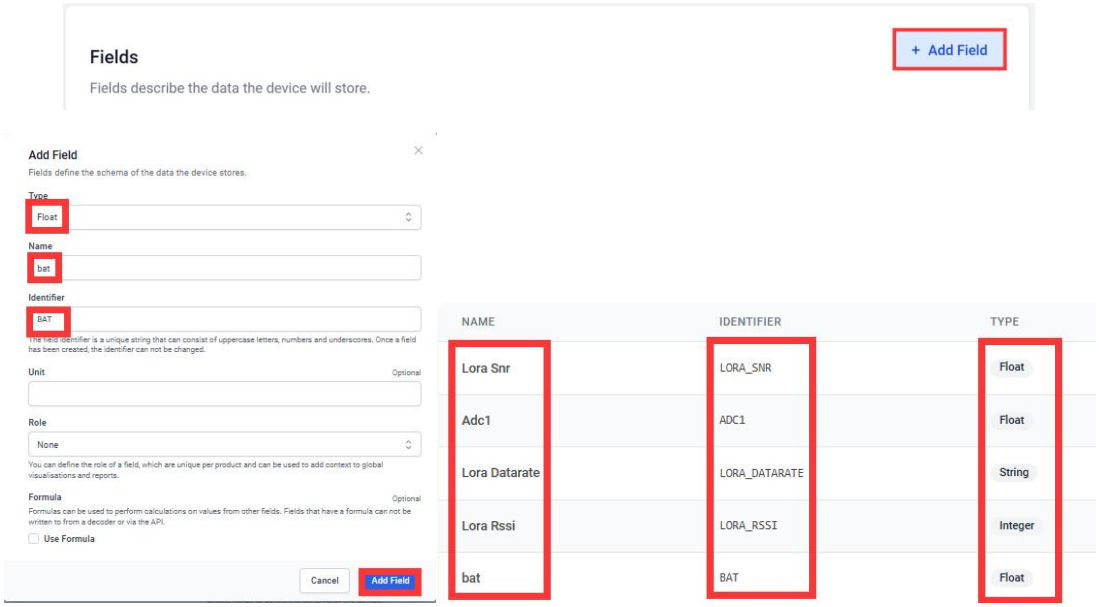




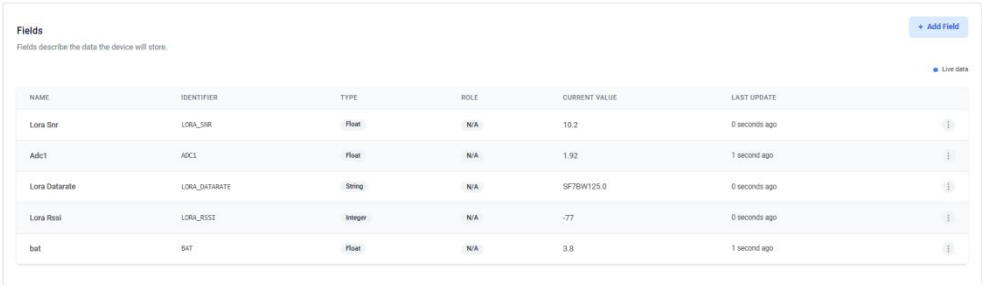
```
function Decoder(payload, port) {
    var input = {
        bytes: payload
    };

    // var num = input.bytes[0] * 256 + input.bytes[1];
    var bat = input.bytes[2] / 10.0;
    var ADC1 = (input.bytes[3] * 256 + input.bytes[4]) / 1000.0 //V
    var ADC2 = (input.bytes[5] * 256 + input.bytes[6]) / 1000.0 //V
    var ADC3 = (input.bytes[7] * 256 + input.bytes[8]) / 1000.0 //V
    var ADC4 = (input.bytes[9] * 256 + input.bytes[10]) / 1000.0 //V
    var Differentialbits = (input.bytes[11] * 256 + input.bytes[12]) / 1000.0 //V
    var time_interval = (input.bytes[13] * 16777216 + input.bytes[14] * 65536 + input.bytes[15] * 256 + input.bytes[16]) / 1000.0//S

    var decoded = {
        bat: bat,
        ADC1: ADC1
        ADC2: ADC2
        ADC3: ADC3
        ADC4: ADC4
        Differentialbits: Differentialbits
    };

    // 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 =
            (normalizedPayload.gateways && Array.isArray(normalizedPayload.gateways) && normalizedPayload.gateways.length > 0 && normalizedPayload.gateways[0].snr) || 0;
        decoded.lora_datarate = normalizedPayload.data_rate || 'not retrievable';
    } catch (error) {
        console.log('Error occurred while decoding LoRa properties: ' + error);
    }

    return [
        { field: "bat", value: decoded.bat },
        { field: "ADC1", value: decoded.ADC1 },
        //{ field: "ADC2", value: decoded.ADC2 },
        //{ field: "ADC3", value: decoded.ADC3 },
```

```
        //{ field: "ADC4", value: decoded.ADC4 },
        //{ field: " Differentialbits", value: decoded. Differentialbits },
        { 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)

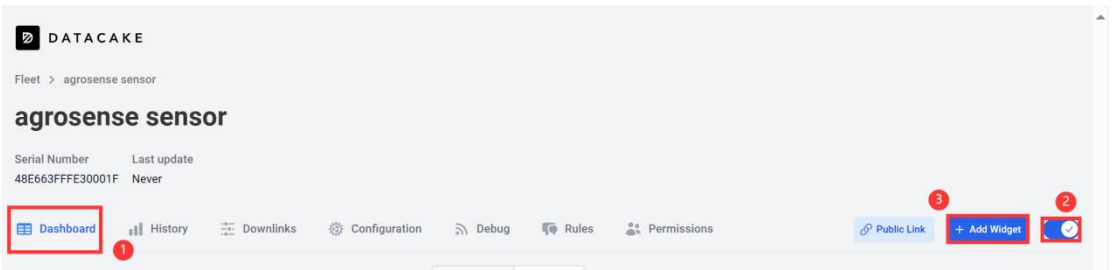


11、Press RST button, wait until the sensor connects to the gateway successfully, you will see the data the sensor is currently reading.

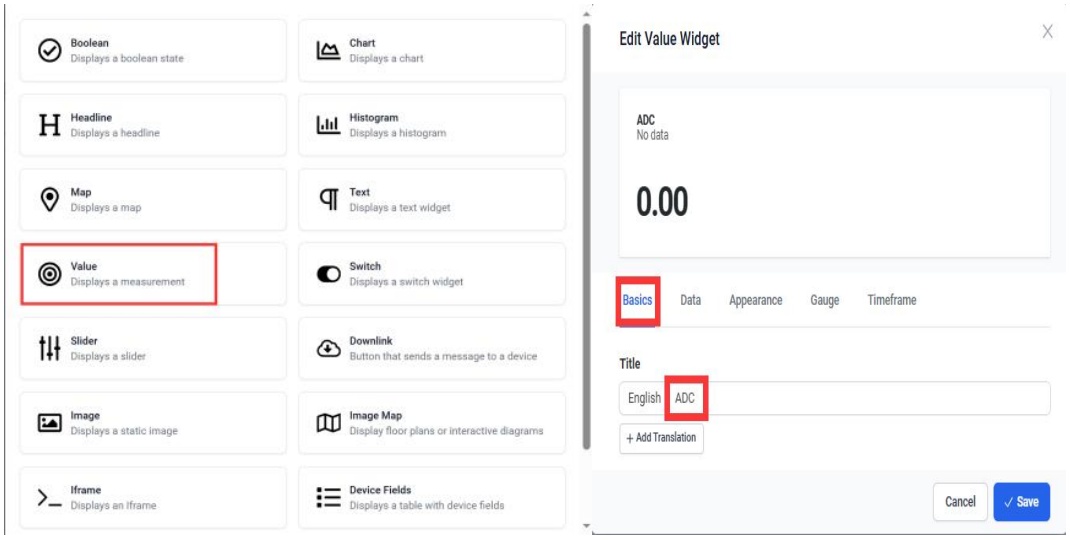


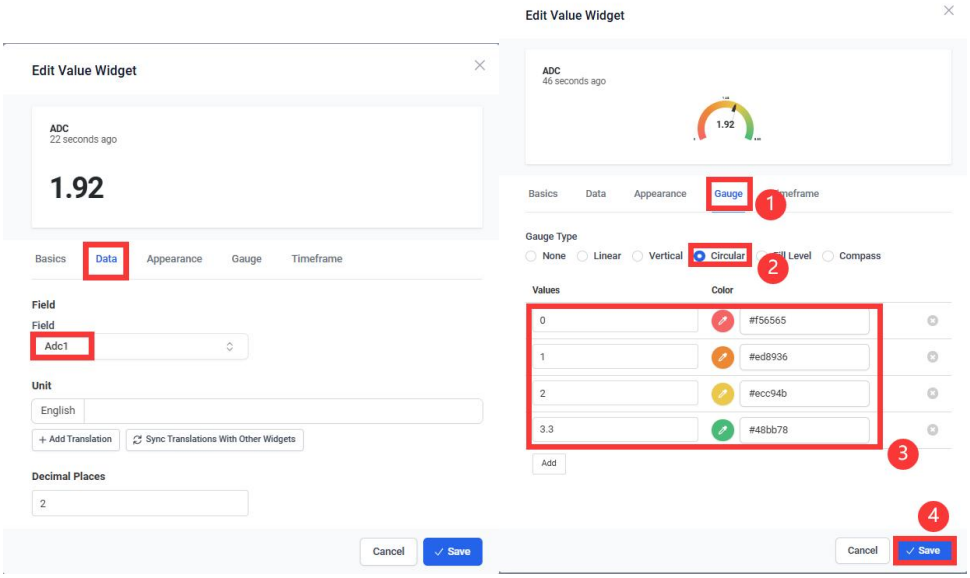
12、To get a better look at the data, we can add widget.

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

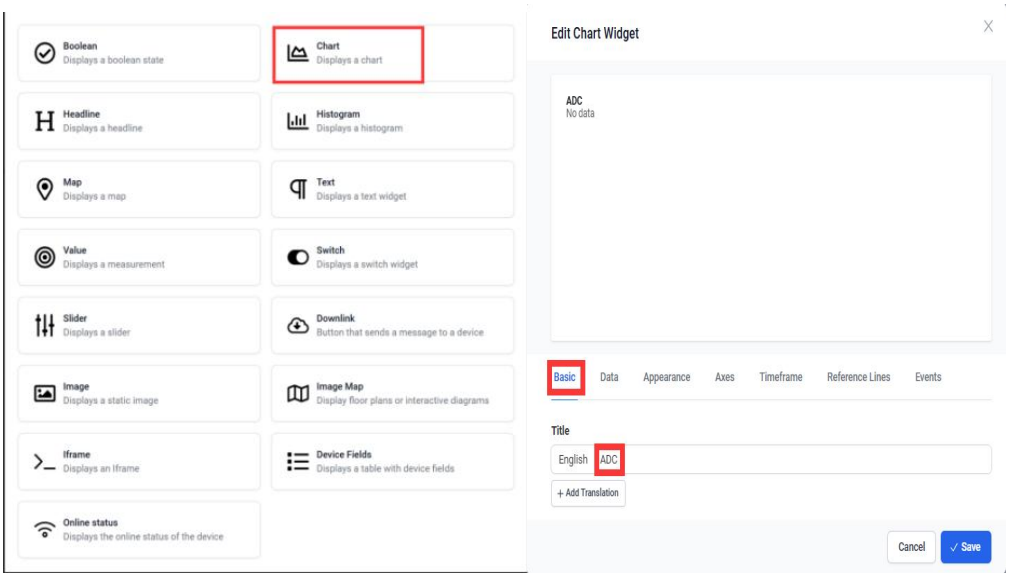


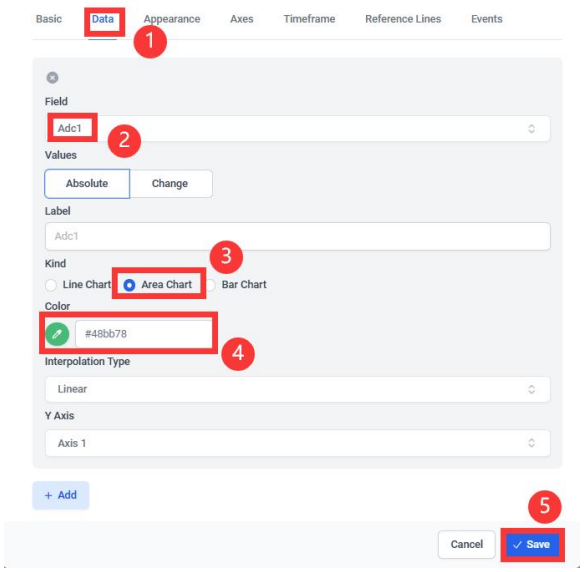
13、Select “Value” and set Title, Field and presentation form as well as the interval color.



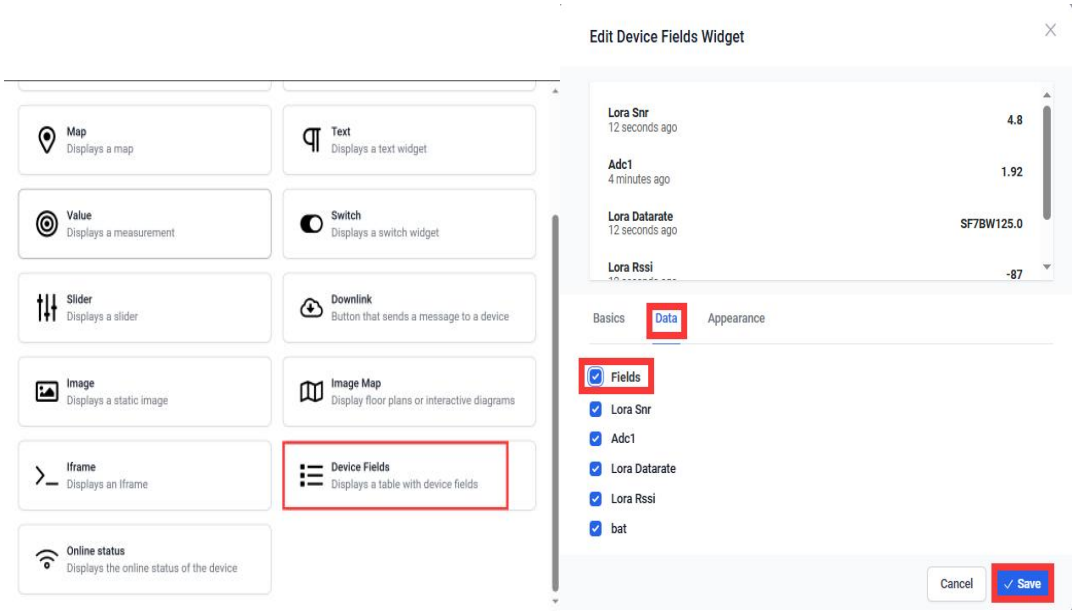


14、Select Chart and set Title, Field, Kind, Line Thickness and click “save”.



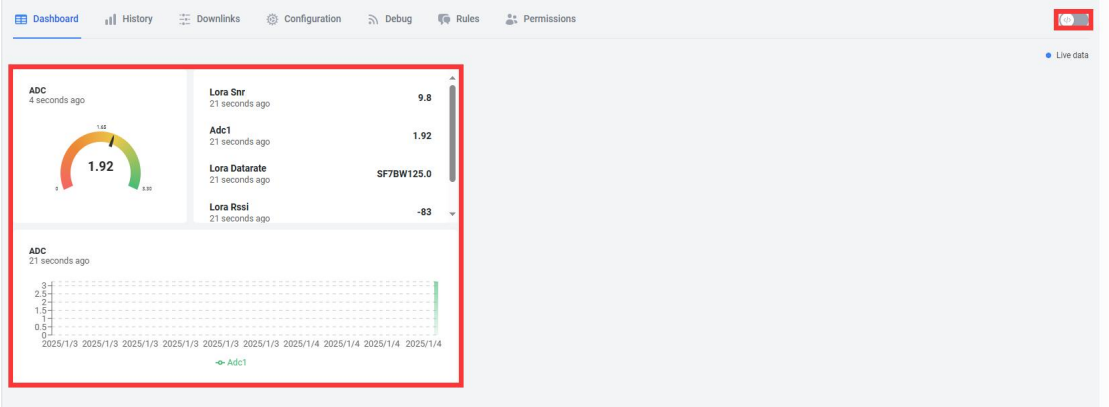


15、Select Device Fields, check “Fields” and click “Save”.



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

17、The steps for humidity are the same as above, and you can add your own.



3.2.1 Downlink

The downlink has four 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)

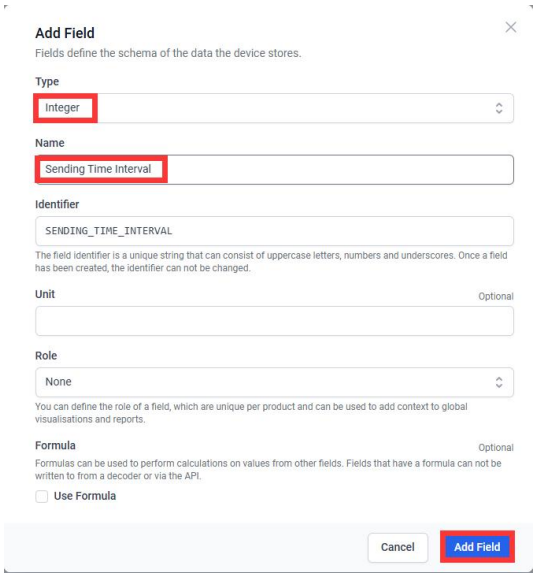
Users can view previous data based on this feature.

- Turn on or off the corresponding ADC channel (Fport6, the default is all on);

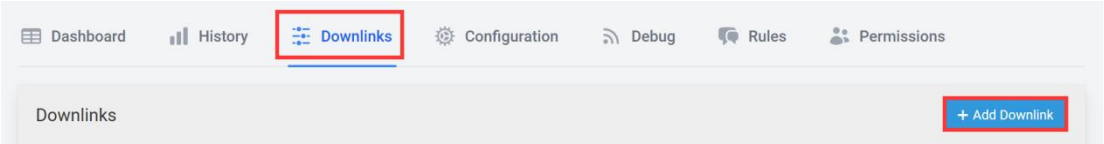
Differentialbits	ADC4	ADC3	ADC2	ADC1	Note
0	0	0	0	1	0x01 =enable the ADC1 channel
0	0	0	1	0	0x02 =enable the ADC2 channel
0	0	1	0	0	0x04 =enable the ADC3 channel
0	1	0	0	0	0x08 =enable the ADC4 channel
1	0	0	0	0	0x10 =enable the Differentialbits channel, At this point, FPort7 must be set to 1
0	0	0	0	0	0x00 =Turn off all channels
1	1	1	1	1	0x1F =enable all channels

- Set the third and fourth channels as differential inputs (Fport7)

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



2、Click ”Downlink-->Add Downlink”.

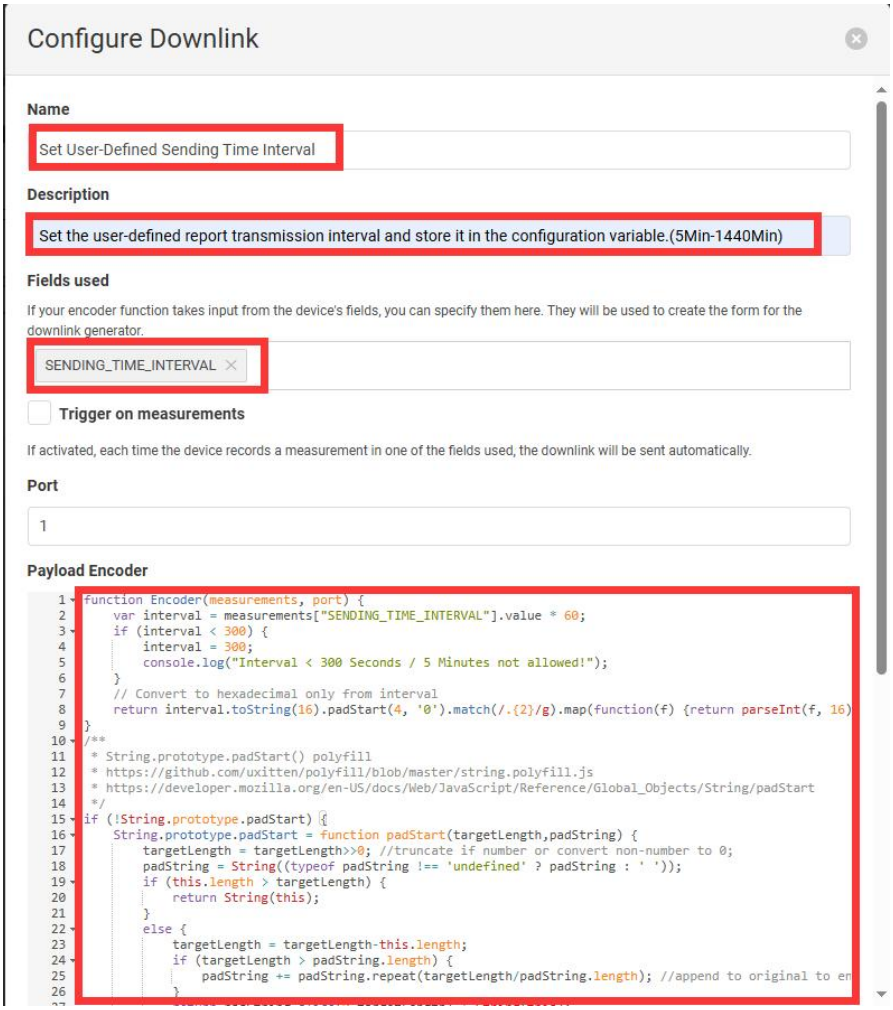


Enter name、description、fields used and payload encoder respectively.

Name: Set User-Defined Sending Time Interval

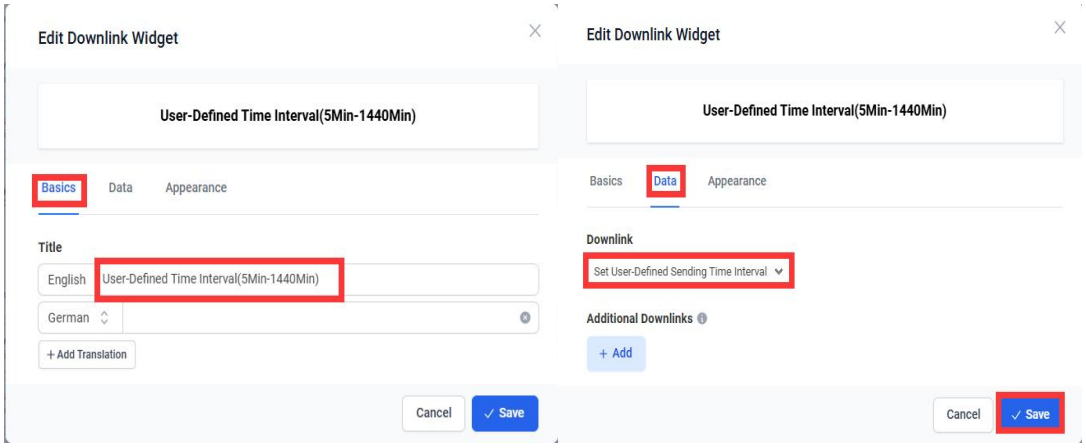
Description: Set the user-defined report transmission interval and store it in the configuration variable.(5Min-1440Min)

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.