



AgroSense LoRaWAN®

4-Channel ADC 12 bits Manual

V1.1

Author: Yuki

Time: 2025.05.09

| Date | Versions | Description | Author |
|------------|----------|--------------------------------|--------|
| 2025.01.18 | V1.0 | Introduction to Use & Function | Yuki |
| 2025.05.09 | V1.1 | Add Wiring diagram | Yuki |

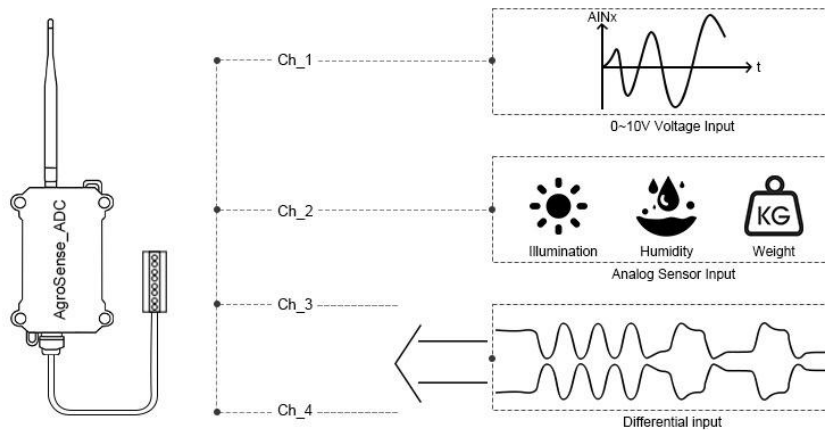
Contents

| | |
|---|----------|
| 1 Product Description | 1 |
| 1.1 Introduction | 1 |
| 1.2 Feature | 2 |
| 1.3 Parameter | 2 |
| 2 Technical route | 4 |
| 2.1 System Framework | 4 |
| 2.2 Regional frequency band | 4 |
| 3 Usage | 6 |
| 3.1 TTN and ThingSpeak | 7 |
| 3.1.1 Network Server configuration | 7 |
| 3.1.2 Decoder | 9 |
| 3.1.3 Application Server configuration | 12 |
| 3.1.4 Connect the Network Server and Application Server | 13 |
| 3.1.5 Downlink | 14 |
| 3.2 Datacake | 17 |
| 3.2.1 Downlink | 24 |

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

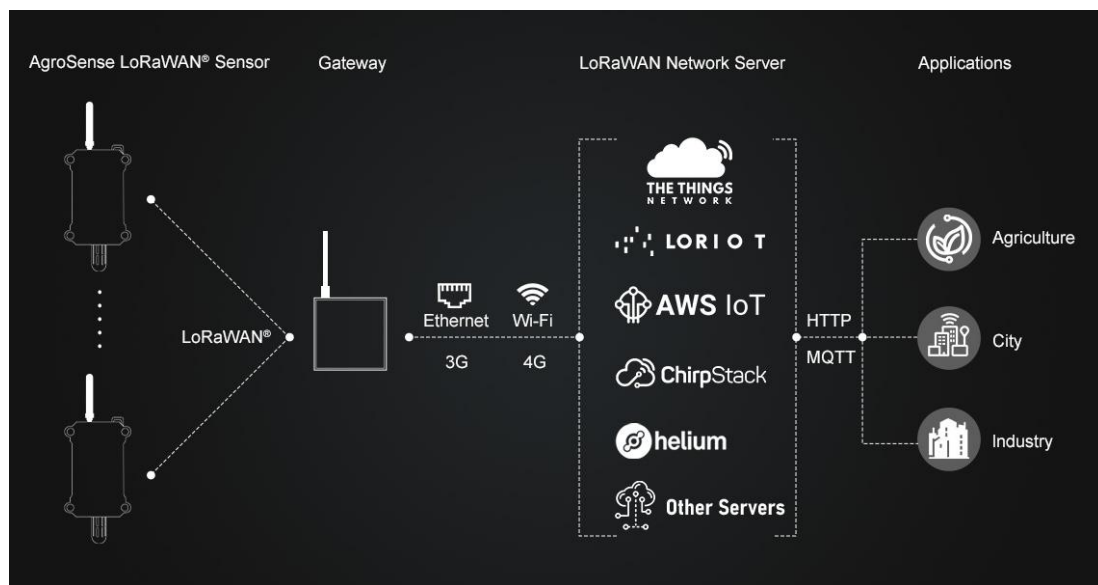
| | |
|-----------------------|-------------------------------|
| Power Supply | 1 x 18650 3.7V Lion batteries |
| Operating Temperature | -40°C ~85°C |
| Protection Class | IP68 |
| Dimensions | 131 × 62.7 × 27.5 mm |
| Mounting | Wall Mounting |

2 Technical route

2.1 System Framework

AgroSense 4-Channel ADC 12 bits Sensor uses LoRAWAN technology, and 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. |



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

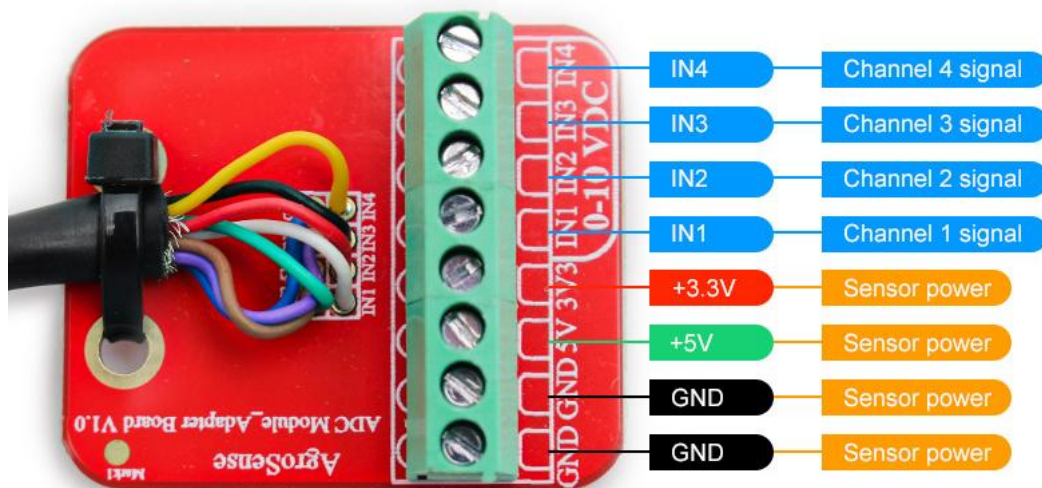
We use The Things Network as our Network Server, we need to configuration the country/ area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak or Datacake.

| | |
|---------|---|
| 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)

Wiring diagram

IN1–IN4 are connected to the sensor signal lines, while 3.3V and 5V are connected to the sensor power supply.



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

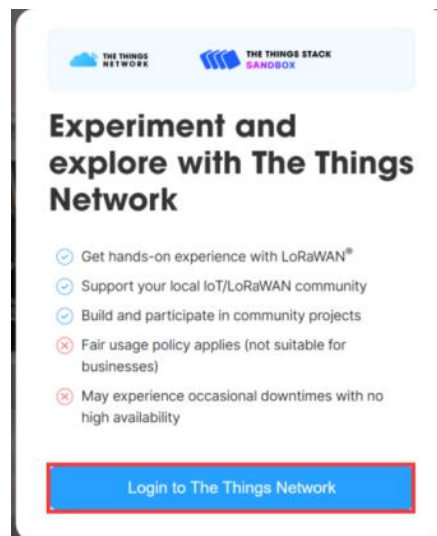


| Agrosense | ADC model |
|-----------|-----------|
| IN1 | OUT |
| 3V3 | VCC |
| GND | GND |

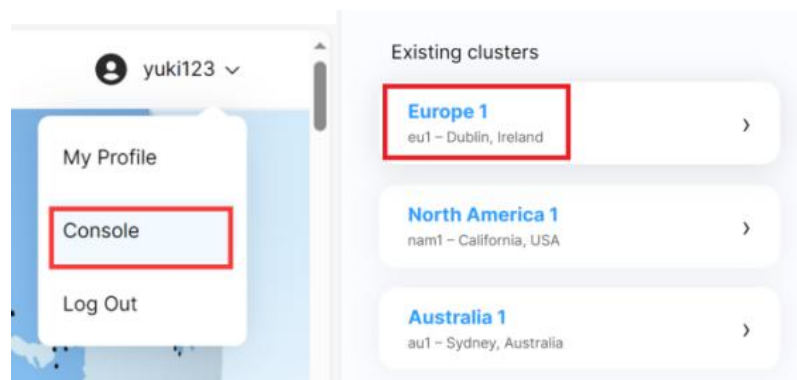
3.1 TTN and ThingSpeak

3.1.1 Network Server configuration

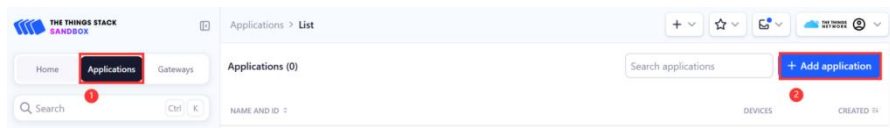
- Open The Things Network in your browser and login it. (Or register an account)



- 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

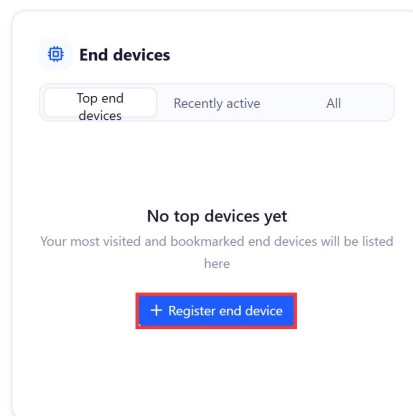
Description

Description for my new application

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

Create application

- Click “+ Register and device”.



- Following 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.

End device type

Input method ⓘ

☐ Select the end device in the LoRaWAN Device Repository

☒ Enter end device specifics manually 1

Frequency plan ⓘ

Europe 863-870 MHz (SF9 for RX2 - recommended)

LoRaWAN version ⓘ

LoRaWAN Specification 1.0.3

Regional Parameters version ⓘ

RP001 Regional Parameters 1.0.3 revision A 2

Show advanced activation, LoRaWAN class and cluster settings

Provisioning information

JoinEUI ⓘ *

48 FF 00 00 00 01 65 Confirm

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

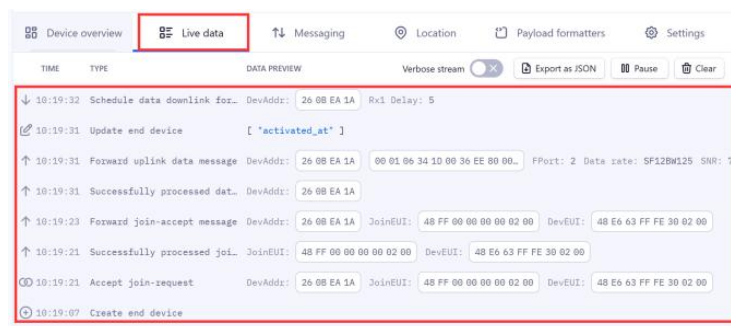
☐ Register another end device of this type

Register end device

3 continue, please enter the JoinEUI 4 and device so we can determine onboarding

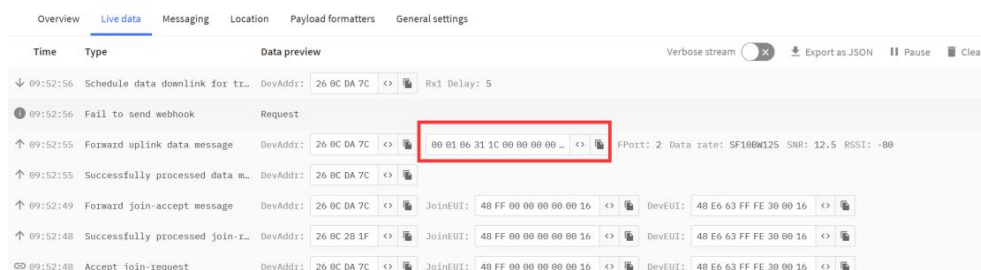


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



3.1.2 Decoder

- Now, we need to decoder the data.



| Data length | Data description | Value range | Explanation |
|-------------|-----------------------------|-------------|---|
| byte 0 | Data packet sequence number | 0-0xFFFF | Counting starts from 0 and increments, resetting back to 0 after reaching 65535 |

AgroSense LoRaWAN® 4-Channel ADC 12 bits

| | | | |
|----------------|---|--------------|--|
| | high 8 bits | | |
| 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 For example, if the value is 0x0CE4 = 3300, then the value is 3300mV. |
| byte 6 | ADC_1 bits 0 to 7 | | |
| 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.

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

Channel ID: 2601128
 Author: mwa0000034232775
 Access: Private

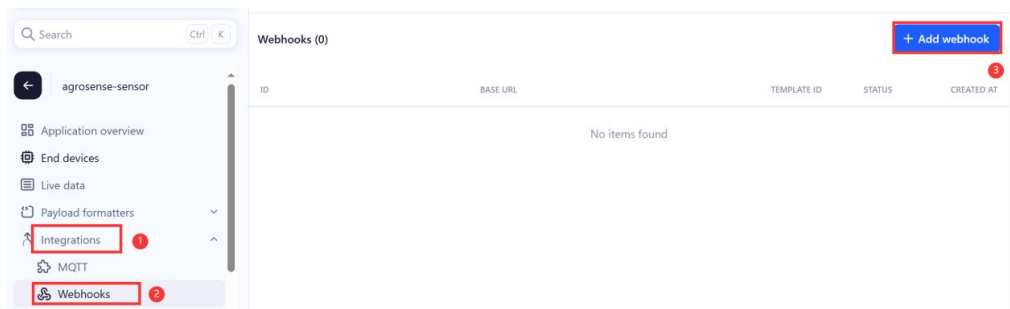
Private View Public View Channel Settings Sharing API Keys

Write API Key

Key RXASRM3U00ACSSW1

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”.

Webhook ID *

my-new-thingspeak-webhook

Channel ID *

2601128

ThingSpeak Channel ID

API Key *

.....

Thingspeak Write API Key

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 interval (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);

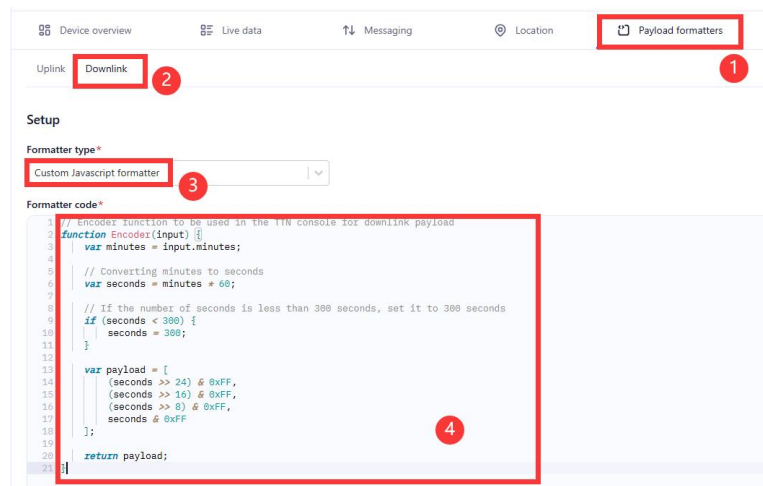
| Differentialbits | ADC4 | ADC3 | ADC2 | ADC1 | Note |
|------------------|------|------|------|------|-------------------------------|
| 0 | 0 | 0 | 0 | 1 | 0x01 =enable the ADC1 channel |

| | | | | | |
|---|---|---|---|---|--|
| 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 |

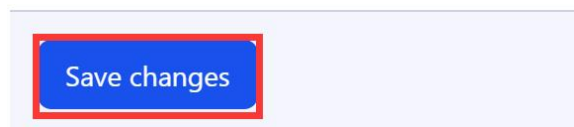
- Set the third and fourth channels as differential inputs (Fport7)

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
}
```

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

```

1 {
2   "minutes": 5
3 }
    
```

The decoded payload of the downlink message

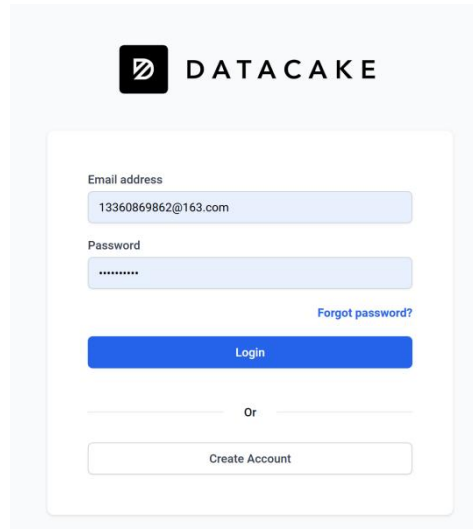
☐ Confirmed downlink

Schedule downlink

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

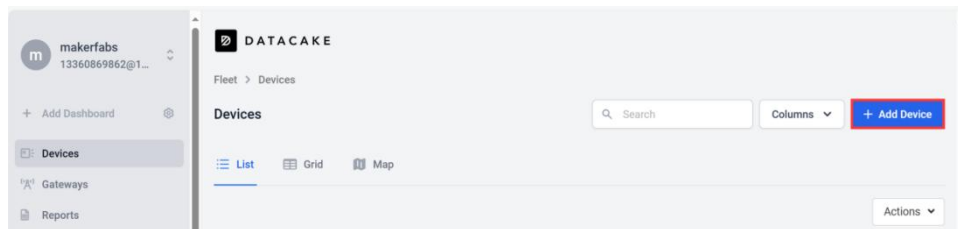
3.2 Datacake

1、Login datacake or Create Account

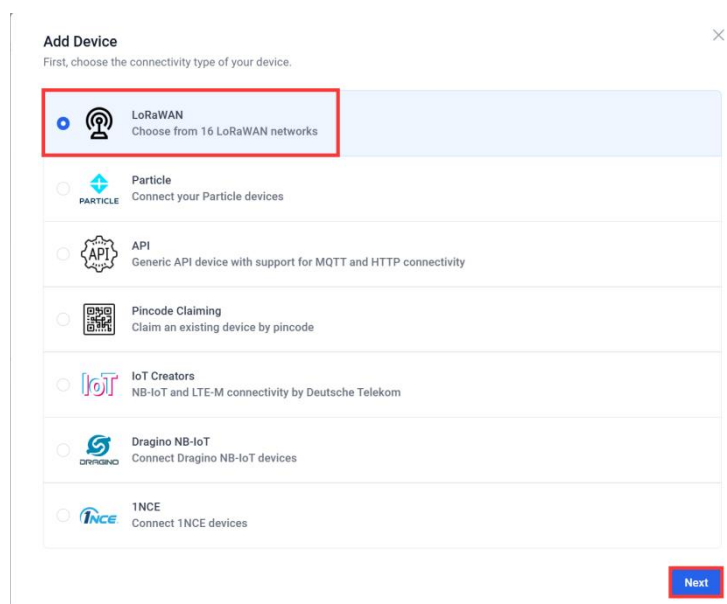


The image shows the Datacake login and registration interface. At the top is the Datacake logo. Below it is a form with two input fields: 'Email address' containing '13360869862@163.com' and 'Password' with masked characters. A 'Forgot password?' link is next to the password field. Below the inputs is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

2、Click “Add Device”



3、Select LoRaWAN and click “Next”



The image shows the 'Add Device' dialog box. It has a 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 and an icon:

- LoRaWAN** (selected, highlighted with a red box): 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

 At the bottom right is a red-bordered 'Next' button.

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 1

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 2

Back Next 3

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

The Things Stack V3
TTN V3 / Things Industries

Helium
Use your own console

LORIoT

ChirpStack

Activity

KPN

Showing 1 to 6 of 15 results Previous Next

Back Next

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

Add LoRaWAN Device You can add individually billed devices. ✕

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 | APPKEY 1.0 / NWKEY 1.1 | FREQUENCY | DEVICE CLASS |
|---------------------------------|------------------|---------------------------------|--|----------------------------------|--------------|
| 48 E5 63 FF FE 30 00 1E 8 bytes | agrosense sensor | 48 FF 00 00 00 00 00 1E 8 bytes | F3 FD 45 8F 9F 21 C7 CB 54 89 7A AB CS 34 CC AB 16 bytes | United States 902-928 MHz, FSB 2 | Class A |

+ Add another device

Back Next

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

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

| 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 |

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

When your devices sends data, the payload will be passed to the payload decoder, alongside the event's name. The payload decoder then transforms it to measurements.

```

1 function decoder(payload, port) {
2   var input = {
3     bytes: payload
4   };
5
6   // var num = input.bytes[0] * 256 + input.bytes[1];
7   var bat = input.bytes[0] / 1000;
8   var ADC1 = (input.bytes[2] * 256 + input.bytes[3]) / 1000.0; //V
9   var ADC2 = (input.bytes[4] * 256 + input.bytes[5]) / 1000.0; //V
10  var ADC3 = (input.bytes[6] * 256 + input.bytes[7]) / 1000.0; //V
11  var ADC4 = (input.bytes[8] * 256 + input.bytes[9]) / 1000.0; //V
12  var Differentialbits = (input.bytes[10] * 256 + input.bytes[11]) / 1000.0; //V
13  var time_interval = (input.bytes[12] * 256 + input.bytes[13]) * 65536 + input.bytes[14] * 256 + input.bytes[15] / 1000.0; //S
14
15  var decoded = {
16    bat: bat,
17    ADC1: ADC1,
18    ADC2: ADC2,
19    ADC3: ADC3,
20    ADC4: ADC4,
21    Differentialbits: Differentialbits
22  };
23
24  // Test for LoRa properties in normalizedPayload
25  try {
26    console.log('normalizedPayload', normalizedPayload); // Log to check normalizedPayload structure
27
28    decoded.lora_rssi =
29      (normalizedPayload.gateways && Array.isArray(normalizedPayload.gateways) && normalizedPayload.gateways.length > 0 && normalizedPayload.gateways[0].rssi) || 0;
30    decoded.lora_sn =
31      (normalizedPayload.gateways && Array.isArray(normalizedPayload.gateways) && normalizedPayload.gateways.length > 0 && normalizedPayload.gateways[0].snr) || 0;
32    decoded.lora_data_rate = normalizedPayload.data_rate || 'not retrievable';
33  } catch (error) {
34    console.log('error occurred while decoding LoRa properties: ' + error);
35  }
36 }
    
```

Save

```
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)

Fields
 Fields describe the data the device will store.

+ Add Field

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

Type

Float

Name

bat

Identifier

BAT

Unit

Role

None

Formula

Use Formula

| NAME | IDENTIFIER | TYPE |
|---------------|---------------|---------|
| Lora Snr | LORA_SNR | Float |
| Adc1 | ADC1 | Float |
| Lora Datarate | LORA_DATARATE | String |
| Lora Rssi | LORA_RSSI | Integer |
| bat | BAT | Float |

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

Fields
 Fields describe the data the device will store.

+ Add Field

| NAME | IDENTIFIER | TYPE | ROLE | CURRENT VALUE | LAST UPDATE |
|---------------|---------------|---------|------|---------------|---------------|
| Lora Snr | LORA_SNR | Float | N/A | 10.2 | 0 seconds ago |
| Adc1 | ADC1 | Float | N/A | 1.92 | 1 seconds ago |
| Lora Datarate | LORA_DATARATE | String | N/A | SF7BW125.0 | 0 seconds ago |
| Lora Rssi | LORA_RSSI | Integer | N/A | -77 | 0 seconds ago |
| bat | BAT | Float | N/A | 3.8 | 1 seconds 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 and presentation form as well as the interval color.

The first screenshot shows the 'Edit Value Widget' dialog. The 'Value' widget is selected in the left panel. The 'Title' is set to 'ADC' and the language is 'English'. The 'Data' tab is selected.

The second screenshot shows the 'Gauge' tab selected. The 'Gauge Type' is set to 'Circular'. The 'Values' table is shown with the following data:

| Value | Color |
|-------|---------|
| 0 | #f56565 |
| 1 | #ed8936 |
| 2 | #ecc94b |
| 3.3 | #48bb78 |

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

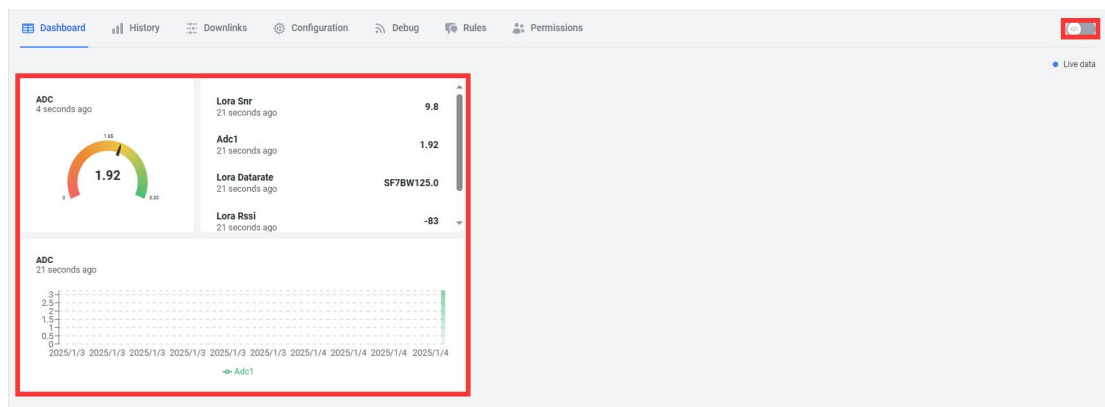
The first screenshot shows the 'Edit Chart Widget' dialog. The 'Chart' widget is selected in the left panel. The 'Title' is set to 'ADC' and the language is 'English'. The 'Basic' tab is selected.

The second screenshot shows the 'Basic' tab selected. The 'Title' is set to 'ADC' and the language is 'English'. The 'Basic' tab is selected.

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 interval (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”

Add Field

Fields define the schema of the data the device stores.

Type:

Name:

Identifier:

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:

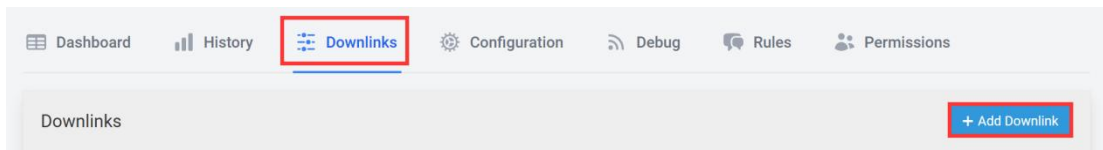
You can define the role of a field, which are unique per product and can be used to add context to global visualisations and reports.

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.(5Min-1440Min)

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

Configure Downlink

Name

Description

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.

☐ 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

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 /**
12  * String.prototype.padStart() polyfill
13  * https://github.com/uxitten/polyfill/blob/master/string.polyfill.js
14  * https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/padStart
15  */
16 if (!String.prototype.padStart) {
17   String.prototype.padStart = function padStart(targetLength,padString) {
18     targetLength = targetLength>>0; //truncate if number or convert non-number to 0;
19     padString = String((typeof padString !== 'undefined' ? padString : ' '));
20     if (this.length > targetLength) {
21       return String(this);
22     }
23     else {
24       targetLength = targetLength-this.length;
25       if (targetLength > padString.length) {
26         padString += padString.repeat(targetLength/padString.length); //append to original to en

```

3、Click "Dashboard-->switch-->+ Add Widget".

Select "Downlink" and setting as follow image.

The image shows two side-by-side screenshots of the 'Edit Downlink Widget' dialog box. The left screenshot shows the 'Basics' tab with the title 'User-Defined Time Interval(5Min-1440Min)' highlighted. The right screenshot shows the 'Data' tab with the 'Downlink' dropdown set to 'Set User-Defined Sending Time Interval' and the 'Save' button highlighted.

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

The image shows two screenshots. The first screenshot shows the 'User-Defined Time Interval(5Min-1440Min)' widget. The second screenshot shows the 'Sending Time Interval' dialog box with the value '1' in the input field and the 'Save measurements and send downlink' button highlighted.