



**AgroSense-carbon dioxide(CO<sub>2</sub>) sensor  
LoRaWAN<sup>®</sup> Manual  
V1.0**

# Contents

<b>1 Product Description .....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Feature .....	1
1.3 Parameter .....	2
<b>2 Technical route .....</b>	<b>3</b>
2.1 System Framework .....	3
2.2 Regional frequency band .....	4
<b>3 Usage .....</b>	<b>5</b>
3.1 Network Server configuration .....	5
3.2 Decoder .....	8
3.3 Application Server configuration .....	10
3.4 Connect the Network Server and Application Server .....	11

# 1 Product Description

## 1.1 Introduction

AgroSense LoRaWAN® Carbon Dioxide (CO2) Sensor can remotely measures carbon dioxide levels in the atmosphere at the range of 400-60000ppm with accuracy  $\pm 20\%$  respectively and resolution 1ppm, also with highly waterproof performance tested to IP68 , making it widely applicable in **agricultural environmental sensing scenarios to support Field Cultivation, greenhouse planting, Warehouse, etc.**

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 18650 3.7V Li-ion cell 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 Carbon Dioxide and report them every 1 hour, duration of work 9 months.

The product requires a 20-second warm-up period to collect data, which results in higher battery consumption than other Agrosense products. So we have chosen lithium batteries as the power supply. Our lithium batteries support cyclic charging and multiple use, allowing users to remove the battery when it is low on power and recharge it.



## 1.2 Feature

- Includes a **high precision** sensor.
- 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 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**.

## 1.3 Parameter

### 1. General Parameters

Product Model	AGLWCD01
Measurement Range	400-60000ppm
Measurement Accuracy	±20%
Resolution	1ppm

### 2.Wireless Parameters

Communication Protocol	Standard LoRaWAN® protocol
Network Access/Operating Mode	OTAA Class A
MAX Transmit Power	21dBm
Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

### 3.Physical Parameters

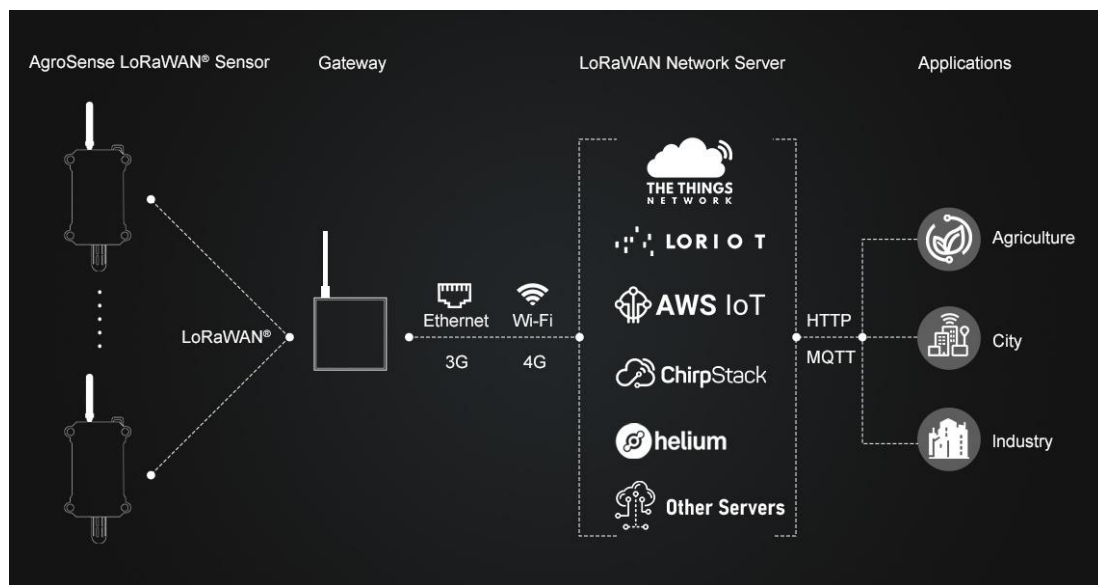
Power Supply	1 x 18650 3.7V Lion batteries
Operating Temperature	-40℃ ~85℃
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

## 2 Technical route

### 2.1 System Framework

AgroSense-carbon dioxide(CO2) 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 CO2 is:**

1. Collect the CO2 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 CO2 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

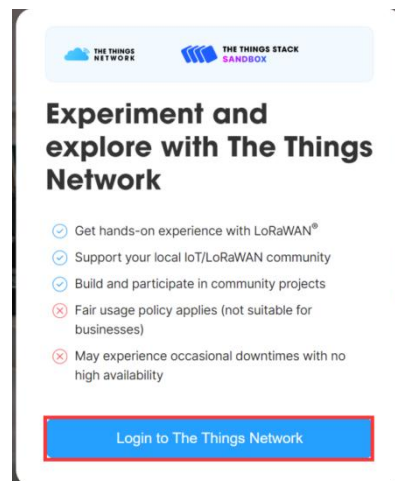
We use The Things Network as our Network Server, we need to configuration the country/ area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

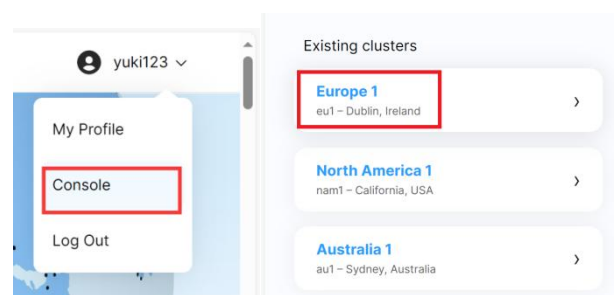
- End Nodes and Gateway: AgroSense-carbon dioxide(CO2) sensor .(The AgroSense series is applicable)
- Network Server: The Things Network. ( Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbox, akenza, ect)

#### 3.1 Network Server configuration

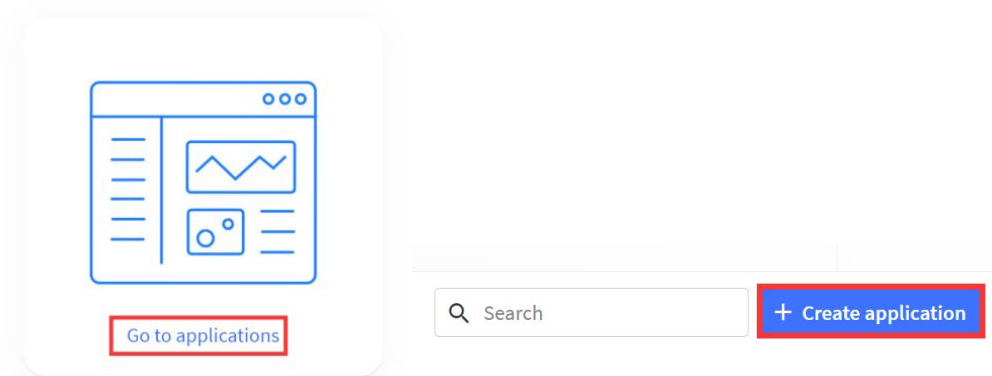
- Open The Things Network in your browser and login it. (Or register an account)



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



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



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

Application ID \*

agrosense-sensor

Application name

My new application

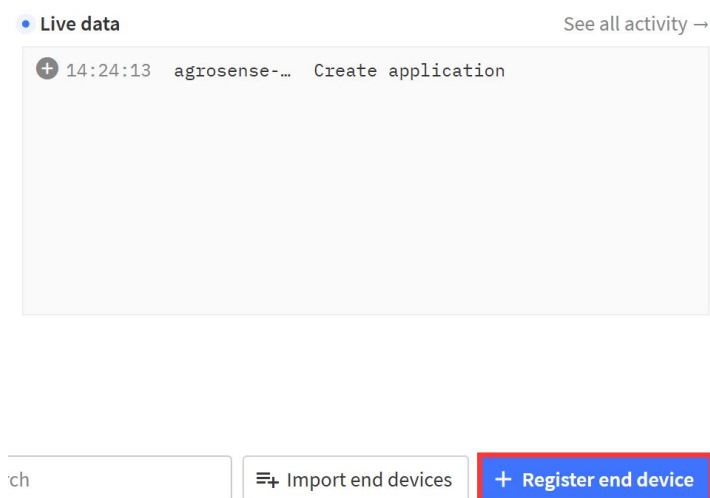
Description

Description for my new application

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

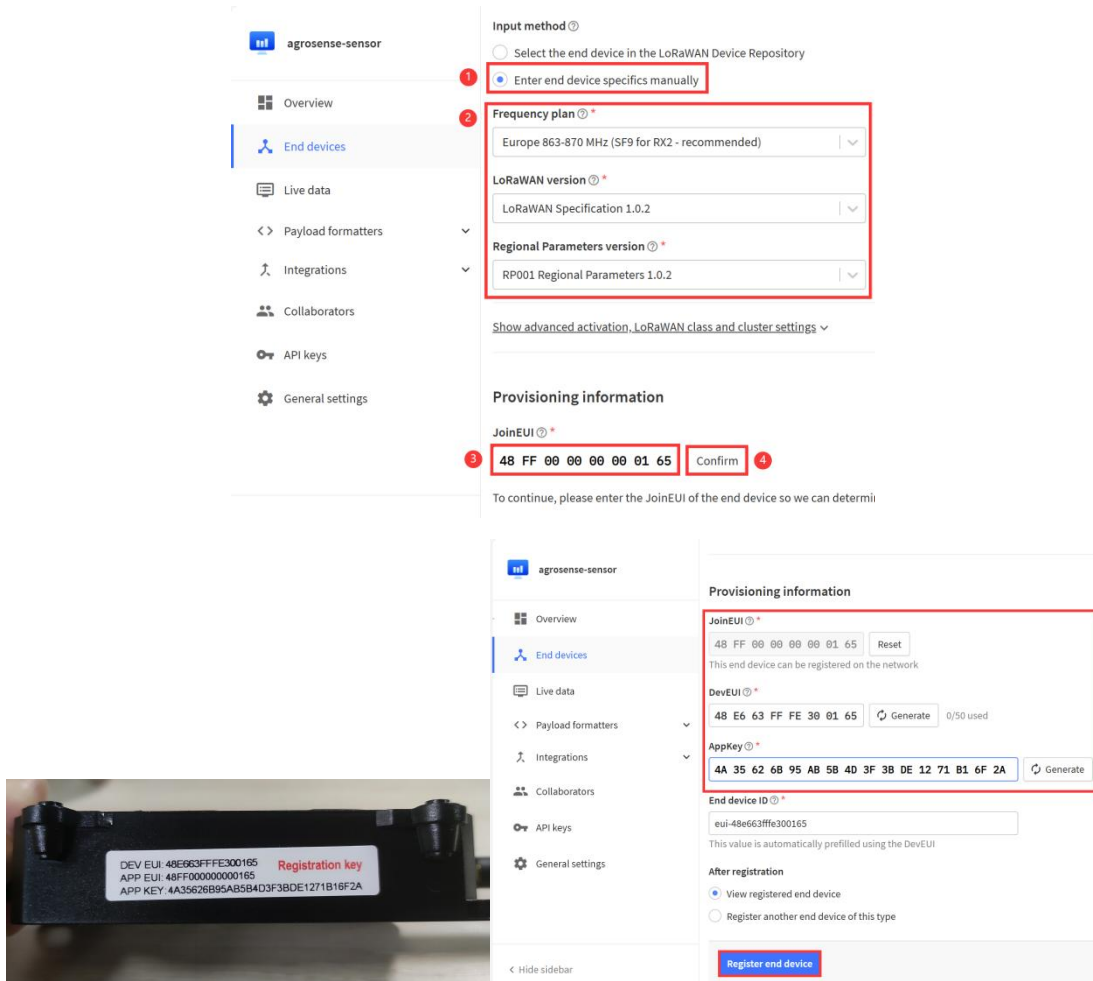
Create application

- Click “+ Register and device”.





- 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.



- Pull out the power switch and press the RES button, you can see the device is connected successfully in the TTN.

**Notice:** Because the sensor has a warm-up time, it will take a while to receive the data.

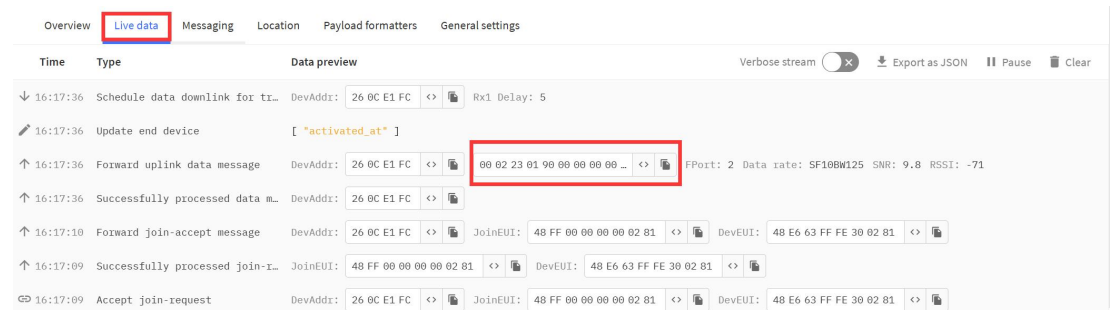


• Live data See all activity →

↑ 16:14:24	Forward join-accept message	DevAddr: 26 0C CD E4	JoinEUI: 48
↑ 16:14:22	Successfully processed join-request	JoinEUI: 48 FF 00 00 00 02 81	
⌂ 16:14:22	Accept join-request	DevAddr: 26 0C CD E4	JoinEUI: 48 FF 00 00
↑ 16:08:23	Forward join-accept message	DevAddr: 26 0C 37 83	JoinEUI: 48
↑ 16:08:21	Successfully processed join-request	JoinEUI: 48 FF 00 00 00 02 81	
⌂ 16:08:21	Accept join-request	DevAddr: 26 0C 37 83	JoinEUI: 48 FF 00 00

## 3.2 Decoder

- Now, we need to decode 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	Carbon Dioxide sensor bits 8 to 15		For example, if the value of bits 8 to 15 is 0x02, and the low 8 bits value is 0x85, then the obtained temperature value is 0x0285 = 645. it is get Carbon Dioxide value is 645ppm.
byte 4	Carbon Dioxide sensor bits 0 to 7		

Example: 0x00, 0x01, 0x28, 0x02, 0x06

Data parsing:

Battery voltage is 4.0V.

CO2 is 518 ppm

- 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 co2 = input.bytes[3] * 256 + input.bytes[4]

    return {

        data: {

            field1: num,

            field2: bat,

            field3: co2,

        },

    };

}
```

- Select “Payload formatters” and follow the steps.

Overview Live data Messaging Location **Payload formatters** General settings

Uplink Downlink

**Setup**

Formatter type\*  
Custom Javascript formatter

Formatter code\*

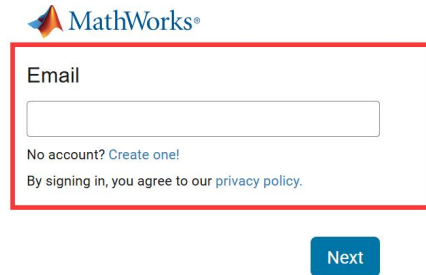
```
function decodeUplink(input) {
    var num = input.bytes[0] * 256 + input.bytes[1]
    var bat = input.bytes[2] / 10.0
    var co2 = input.bytes[3] * 256 + input.bytes[4]
    return {
        data: {
            field1: num,
            field2: bat,
            field3: co2,
        },
    };
}
```

Save changes

### 3.3 Application Server configuration

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

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



MathWorks®

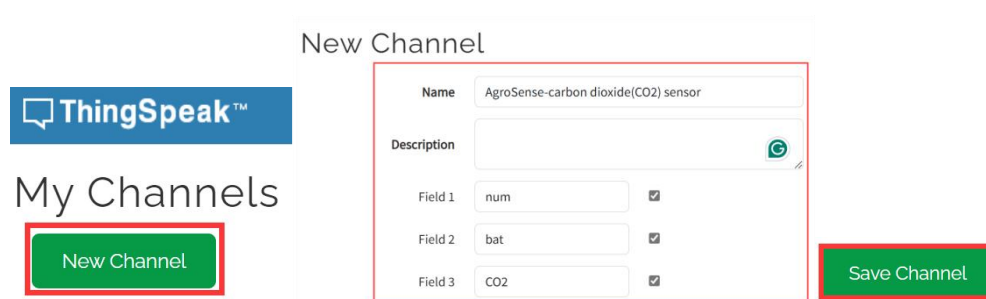
Email

No account? [Create one!](#)

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

Next

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



ThingSpeak™

My Channels

New Channel

New Channel

Name: AgroSense-carbon dioxide(CO2) sensor

Description:

Field 1: num ☒

Field 2: bat ☒

Field 3: CO2 ☒

Save Channel

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

## AgroSense-carbon dioxide(CO2) s

Channel ID: **2601128**

Author: [mwa0000034232775](#)

Access: Private

[Private View](#)

[Public View](#)

[Channel Settings](#)

[Sharing](#)

[API Keys](#)

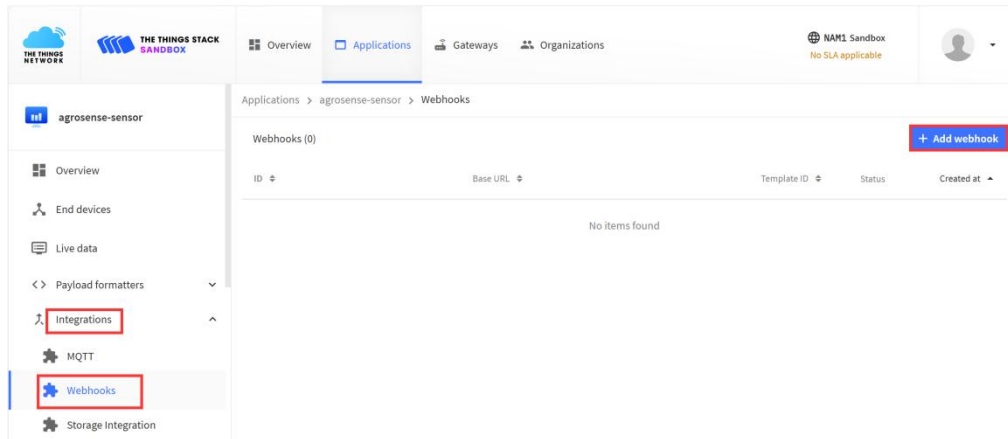
### Write API Key

Key

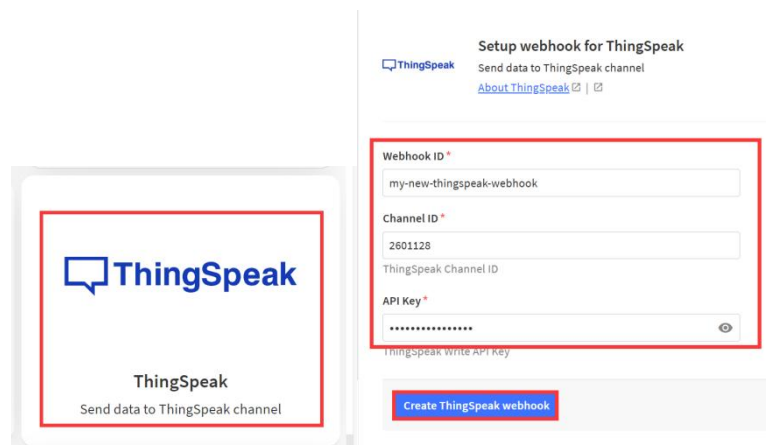
**RXASRM3U00ACSSW1**

### 3.4 Connect the Network Server and Application Server

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



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



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

