



**AgroSense\_Barometric Pressure Sensor  
LoRaWAN® 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® Barometric Pressure Sensor measures the barometric pressure in the atmosphere at the range of 300 to 1100 hPa,  $-40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$  with accuracy  $\pm 0.12$  hPa and resolution 0.01 hPa respectively, also with highly waterproof performance tested to IP68, making it widely applicable in agricultural environmental sensing scenarios to support the smart agricultural production.

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 barometric pressure and report every 1 hour.



## 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°C ~ 85°C, 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        | AGLWBP01    |
| Measurement Range    | 300-1100hPa |
| Measurement Accuracy | 1hPa        |
| Resolution           | 0.01hPa     |

### 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

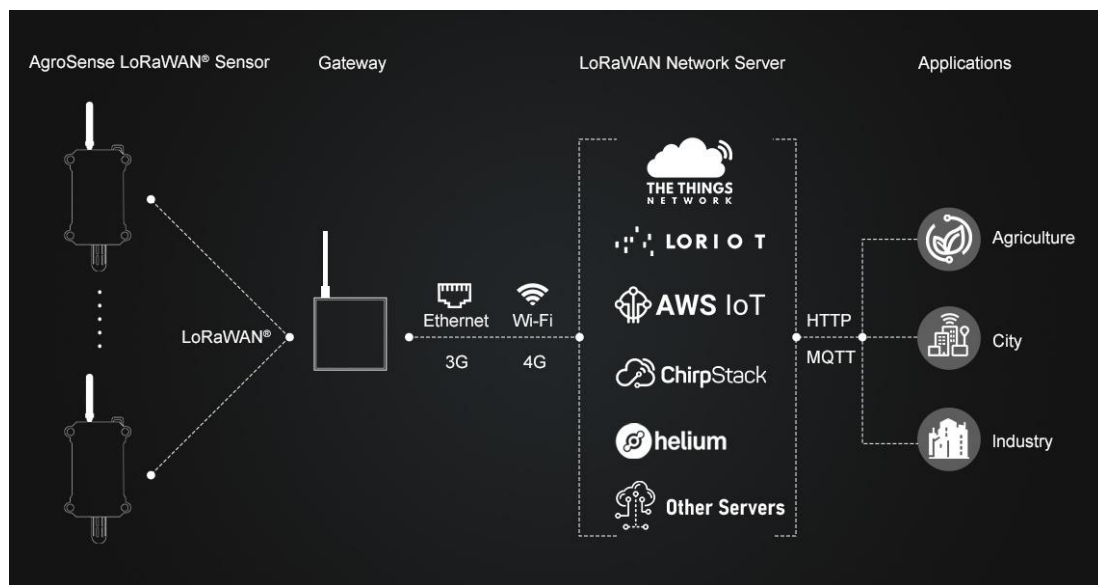
|                       |                        |
|-----------------------|------------------------|
| Power Supply          | 2 x AAA 1.5V batteries |
| Operating Temperature | -40°C ~85°C            |
| Protection Class      | IP68                   |
| Dimensions            | 131 × 62.7 × 27.5 mm   |
| Mounting              | Wall Mounting          |

## 2 Technical route

### 2.1 System Framework

AgroSense\_Barometric Pressure 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 Barometric Pressure is:**

1. Collect the Barometric Pressure 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 Barometric Pressure data 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

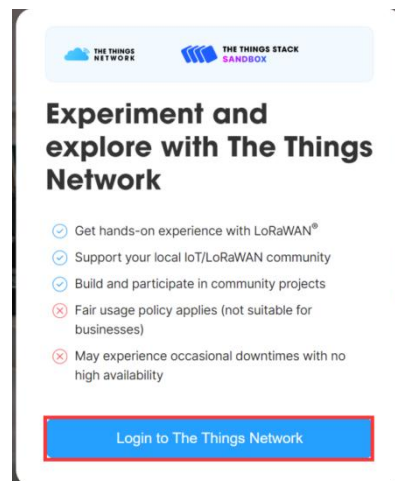
We use The Things Network as our Network Server, we need to configuration the country/ area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

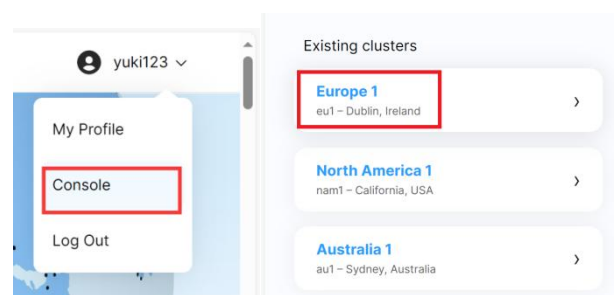
- End Nodes and Gateway: AgroSense\_Barometric Pressure Sensor LoRaWAN®. (The AgroSense series is applicable)
- Network Server: The Things Network. ( Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbox, akenza, ect)

#### 3.1 Network Server configuration

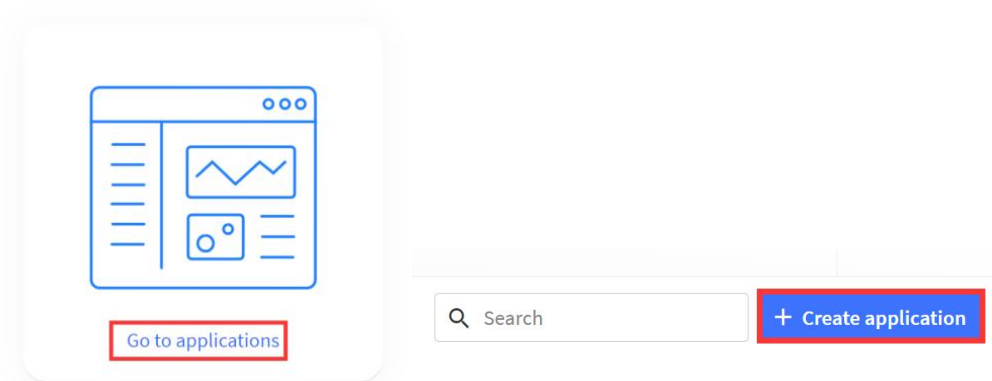
- Open The Things Network in your browser and login it. (Or register an account)



- 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-barometric-pressure-sensor

Application name

My new application

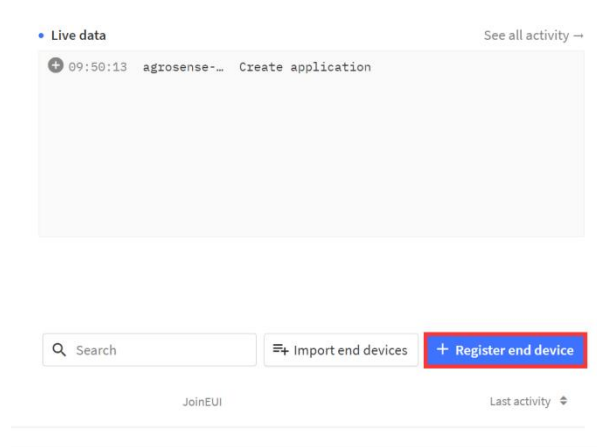
Description

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.



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

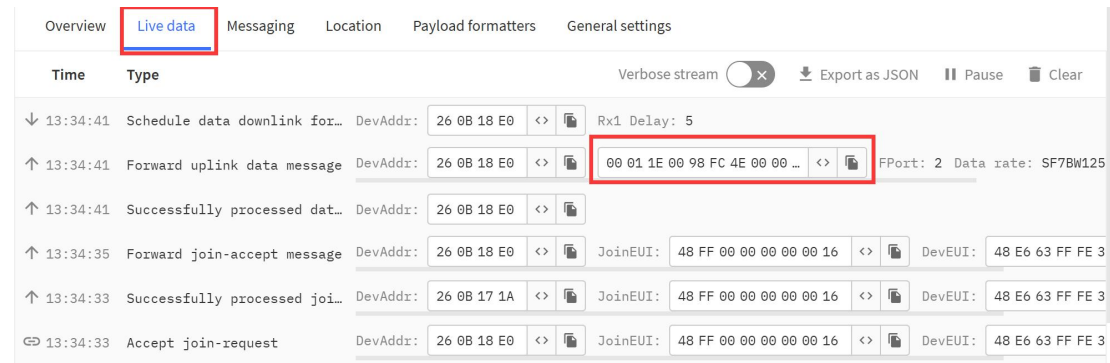


Live data See all activity →

|            |  |                      |
|------------|--|----------------------|
| ↓ 13:34:41 | Schedule data downlink for transmission on Gateway |                      |
| ↑ 13:34:41 | Forward uplink data message                        | DevAddr: 26 0B 18 E0 |
| ↑ 13:34:41 | Successfully processed data message                | DevAddr: 26 0        |
| ↑ 13:34:35 | Forward join-accept message                        | DevAddr: 26 0B 18 E0 |
| ↑ 13:34:33 | Successfully processed join-request                | DevAddr: 26 0        |
| ↻ 13:34:33 | Accept join-request                                | DevAddr: 26 0B 18 E0 |

## 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      | Pressure sensor bits 24 to 31           |             | This data is enlarged 100 times, the real value needs to be divided by 100 to get the actual atmospheric pressure value, the unit Pa. The purpose of multiplying by 100 is to retain the value of atmospheric pressure after 2 decimal places. For example, the Bit23 to Bit0 of the value is 0x0098bb53 = 10009427, divided by 100, it is 100094.27hPa. |
| byte 4      | Pressure sensor bits 16 to 23           |             |  |
| byte 5      | Pressure sensor bits 8 to 15            |             |  |
| byte 6      | Pressure sensor bits 0 to 7             |             |  |
| byte 7      | NC                                      |             |  |
| byte 8      | NC                                      |             |  |

Example: 0x00, 0x02, 0x1C, 0x00, 0x9A, 0x7E, 0xAA, 0x00, 0x00

Data parsing:

Battery voltage is 2.8 V.

Atmospheric pressure is 101249.70 Pa.

- 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 press = (input.bytes[3] * 16777216 + input.bytes[4] * 65536 + input.bytes[5] * 256 + input.bytes[6]) /
100.0

    return {

        data: {

            field1: num,

            field2: bat,

            field3: press,

        },

    };

}
```

- Select “Payload formatters” and follow the steps.

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

1

Uplink Downlink

Setup

Formatter type\*

Custom Javascript formatter 2

Formatter code\*

```
function decodeUplink(input) {
    var num = input.bytes[0] * 256 + input.bytes[1]
    var bat = input.bytes[2] / 10.0
    var press = (input.bytes[3] * 16777216 + input.bytes[4] * 65536 + input.bytes[5] * 256 + input.bytes[6]) / 100.0
    return {
        data: {
            field1: num,
            field2: bat,
            field3: press,
        },
    };
}
```

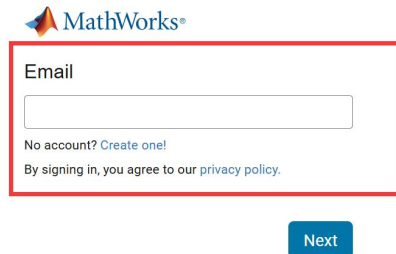
3

Save changes 4

### 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®

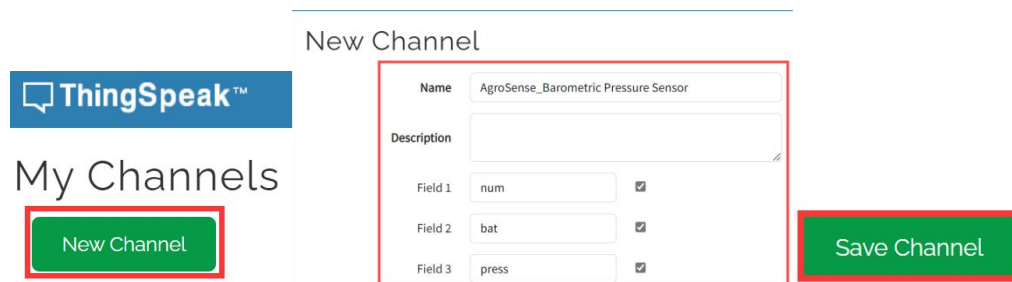
Email

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\_Barometric Pressure Sensor

Description:

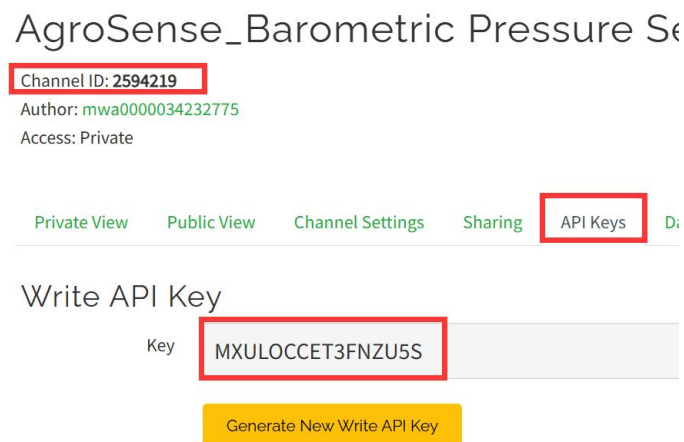
Field 1: num ☒

Field 2: bat ☒

Field 3: press ☒

Save Channel

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



AgroSense\_Barometric Pressure Sensor

Channel ID: 2594219

Author: mwa0000034232775

Access: Private

Private View Public View Channel Settings Sharing API Keys

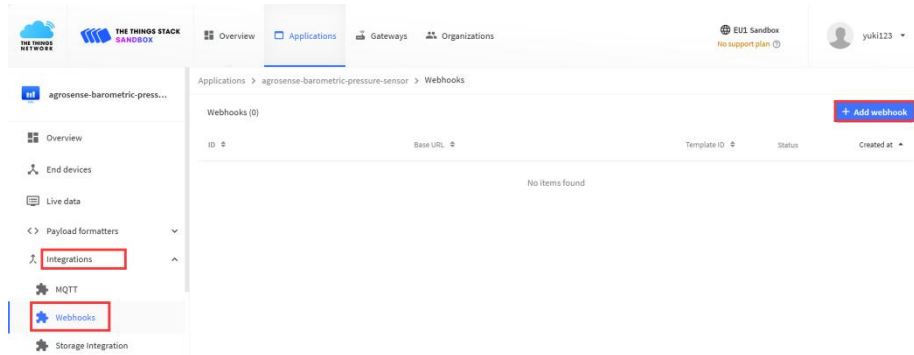
Write API Key

Key: MXULOC CET3FNZU5S

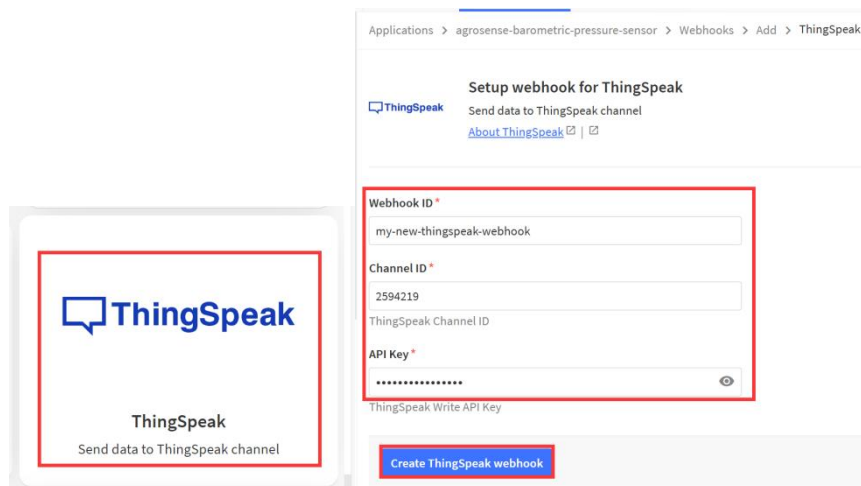
Generate New Write API Key

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

