



**AgroSense\_Positioning Water Leak  
Sensor LoRaWAN® Manual  
V1.0**

Author: Yuki

Time: 2024.10.14

# 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
3.5 Change Time Interval (5-1440min) .....	12

# 1 Product Description

## 1.1 Introduction

This AgroSense LoRaWAN® Positioning Water Leak Sensor detects water leakage at **fixed points**. Suitable for applications such as garage/kitchen water leakage detection. It reports the water leakage status via LoRaWAN protocol every 1 hours (by default, can be set via LoRaWAN **downlink**).

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.



## 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°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**.
- Support **downlink** to modify the time interval (5min-1440min).

## 1.3 Parameter

### 1. General Parameters

Product Model	AGLWPW01
Detection types	Various liquids such as water, oil, etc.
Detection modes	Fixed position detection

### 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

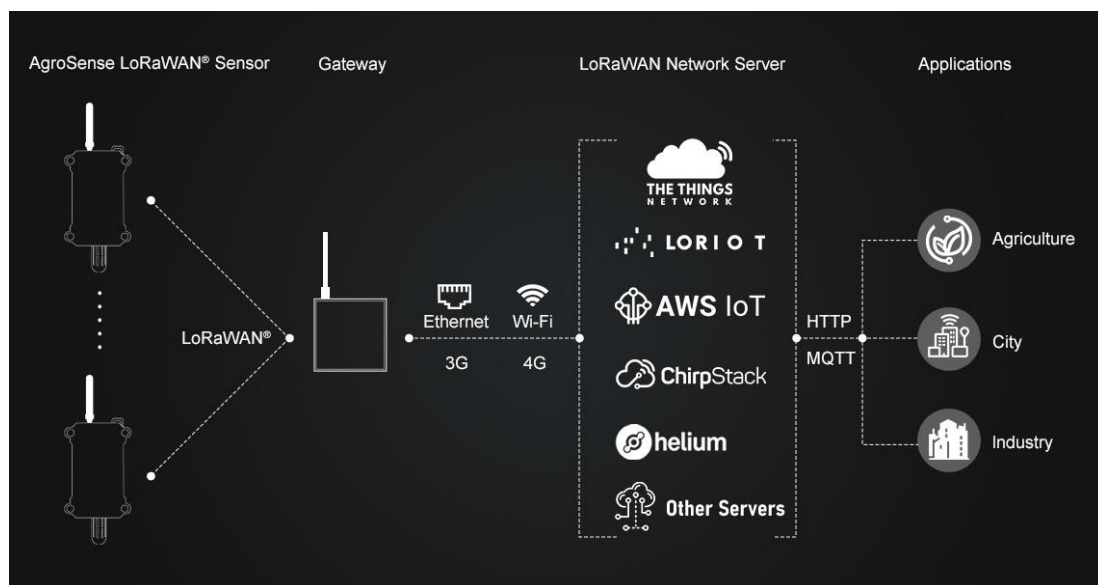
Lead Length	1 .0 meter
Power Supply	2 x AAA 1.5V batteries
Operating Temperature	-40°C ~85°C
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

## 2 Technical route

### 2.1 System Framework

AgroSense\_Positioning Water Leak 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 leak is:**

1. Collect the leak 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 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

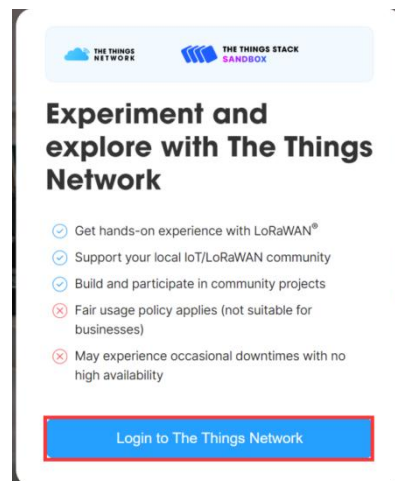
We use The Things Network as our Network Server, we need to configuration the country/ area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

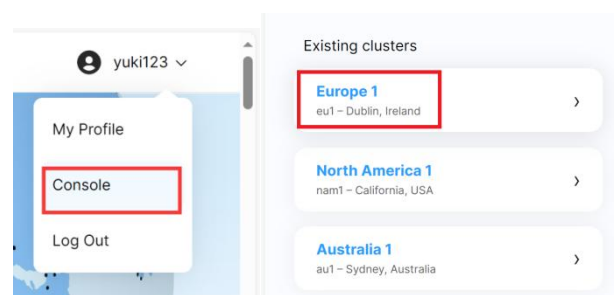
- End Nodes and Gateway: AgroSense\_Positioning Water Leak Sensor.(The AgroSense series is applicable)
- Network Server: The Things Network. ( Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbox, akenza, ect)

#### 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

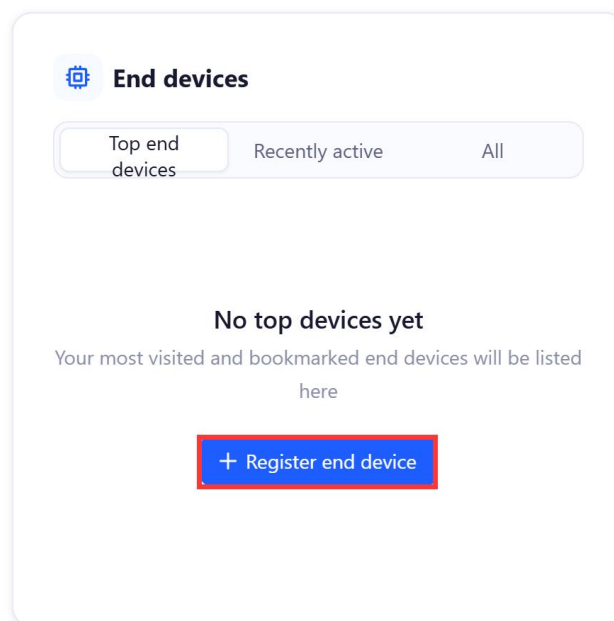
Description

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.

**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) | v

**LoRaWAN version** ⓘ \*

LoRaWAN Specification 1.0.3 | v

**Regional Parameters version** ⓘ \*

RP001 Regional Parameters 1.0.3 revision A | v 2

[Show advanced activation, LoRaWAN class and cluster settings](#)

**Provisioning information**

JoinEUI ⓘ \*

48 FF 00 00 00 00 01 65

Confirm

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

**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-48e663ffe300165

This value is automatically prefilled using the DevEUI

**After registration**

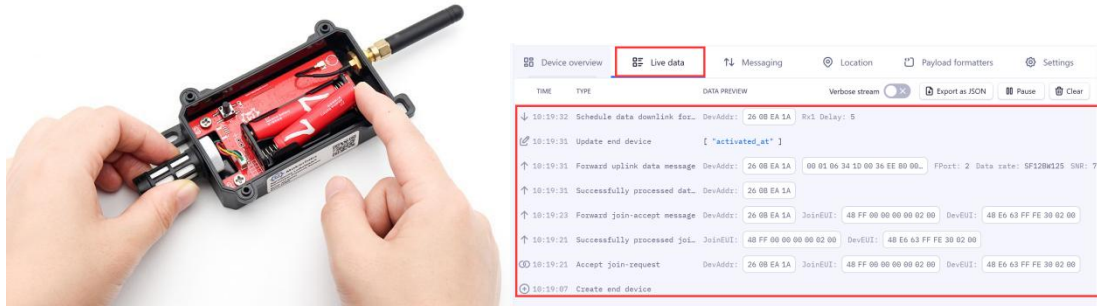
☒ View registered end device

☐ Register another end device of this type

Register end device

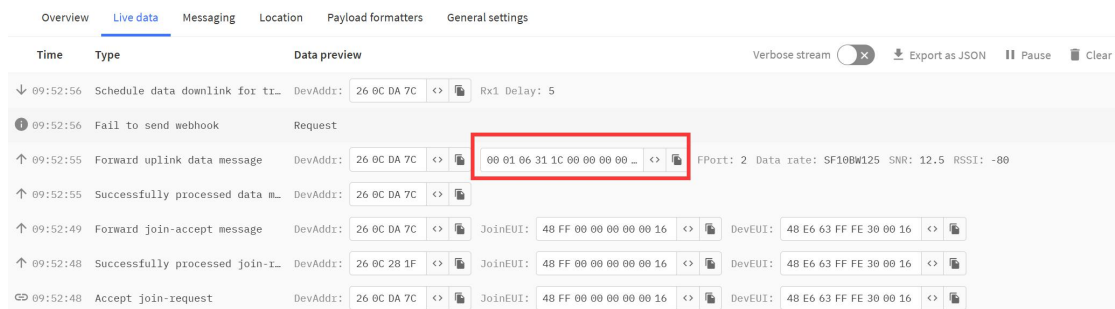


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



## 3.2 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	water leakage symbol		Normal is 0,leakage is 1.
byte 4	Number of leaks in 12 hours 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 number of leaks is 0x0285 = 645. it is get number of leaks value is 645.
byte 5	Number of leaks bits in 12 hours 0		

## AgroSense\_Positioning Water Leak Sensor LoRaWAN®

	to 7		
<b>byte 6</b>	Number of leaks bits in 24 hours to 31		<p>For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the lumen value obtained is 0x00000285, which equals 645. the actual time is 645s</p>
<b>byte 7</b>	Number of leaks bits in 12 hours 16 to 23		
<b>byte 8</b>	Number of leaks bits in 12 hours 8 to 15		
<b>byte 9</b>	Number of leaks bits in 12 hours 0 to 7		

Example: 0x00, 0x03, 0x1D, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x03

Data parsing:

Battery voltage is 2.9 V.

water leakage symbol is 0.(Normal state)

Number of leaks is 2.

Length of last leak is 3s.

- 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 water_leak_flag = input.bytes[3]
  var water_leak_cnt = input.bytes[4] * 256 + input.bytes[5]
  var water_leak_time = input.bytes[6] * 16777216 + input.bytes[7] * 65536 + input.bytes[8] * 256 +
input.bytes[9]
  return {
    data: {
      field1: bat,
      field2: water_leak_flag,
      field3: water_leak_cnt,
      field4: water_leak_time,
    },
  };
}
```

- Select “Payload formatters” and follow the steps.

Device overview Live data Messaging Location **Payload formatters** Settings

Uplink Downlink

Setup

Formatter type\*  
Custom Javascript formatter

Formatter code\*

```

1 function decodeUplink(input) {
2   //var num = input.bytes[0] * 256 + input.bytes[1]
3   var bat = input.bytes[2] / 10.0
4   var water_leak_flag = input.bytes[3]
5   var water_leak_cnt = input.bytes[4] * 256 + input.bytes[5]
6   var water_leak_time = input.bytes[6] * 16777216 + input.bytes[7] * 65536 + input.bytes[8] * 256 + input.bytes[9]
7   return {
8     data: {
9       Field1: bat,
10      Field2: water_leak_flag,
11      Field3: water_leak_cnt,
12      Field4: water_leak_time,
13    },
14  };
15 }

```

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

Name: AgroSense\_Positioning Water Leak Sensor

Description:

Field 1: bat [checked]

Field 2: water leakage state [checked]

Field 3: Number of leaks [checked]

Field 4: Length of last leak [checked]

Save Channel

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

## AgroSense Soil Moisture Sensor

Channel ID: 2599652  
 Author: mwa000034232775  
 Access: Private

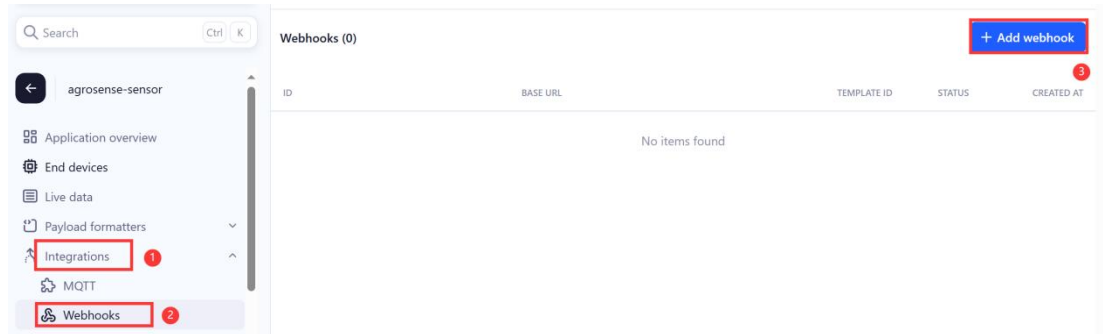
Private View Public View Channel Settings Sharing API Keys

### Write API Key


Key N9IBFTBI3J36T779

## 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”.



**ThingSpeak**

Send data to ThingSpeak channel

**Webhook ID \***

**Channel ID \***

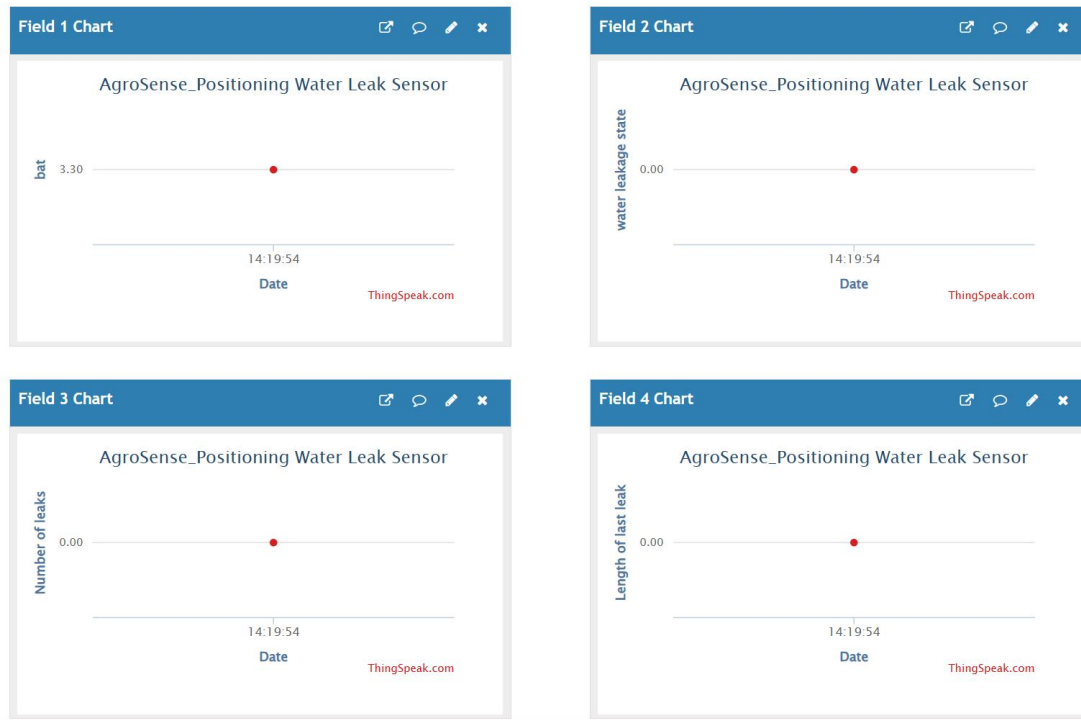
ThingSpeak Channel ID

**API Key \***

ThingSpeak Write API Key

**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.)



### 3.5 Change Time Interval (5-1440min)

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](#).

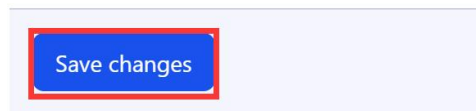
The screenshot shows the 'Payload formatters' configuration page in ThingSpeak. The 'Downlink' tab is active. The 'Formatter type' is set to 'Custom Javascript formatter'. The 'Formatter code' field contains the following JavaScript code:

```

1 // Encoder function to be used in the TTN console for downlink payload
2 function Encoder(input) {
3   var minutes = input.minutes;
4
5   // Converting minutes to seconds
6   var seconds = minutes * 60;
7
8   // If the number of seconds is less than 300 seconds, set it to 300 seconds
9   if (seconds < 300) {
10    seconds = 300;
11  }
12
13  var payload = [
14    (seconds >> 24) & 0xFF,
15    (seconds >> 16) & 0xFF,
16    (seconds >> 8) & 0xFF,
17    seconds & 0xFF
18  ];
19
20  return payload;
21 }

```

2、Click “Save changes”.



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

**Note:** you must use this format:

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

The screenshot shows the 'Messaging' tab selected in the top navigation bar. Below it, the 'Schedule downlink' button is highlighted with a red box. The 'Simulate uplink' button is also visible. Under the 'Schedule downlink' section, the 'Insert Mode' is set to 'Replace downlink queue'. The 'FPort\*' is set to '1'. The 'Payload type' is set to 'JSON', which is highlighted with a red box. The 'Payload' field contains the JSON object `{ "minutes": 5 }`, also highlighted with a red box. Below the payload field, there is a note: 'The decoded payload of the downlink message'. At the bottom, there is a checkbox for 'Confirmed downlink' and a 'Schedule downlink' button highlighted with a red box.

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