



# **AgroSense\_GPS Tracker NEO\_6M Sensor LoRaWAN® Manual V1.0**

Author: Yuki

Time: 2024.11.01

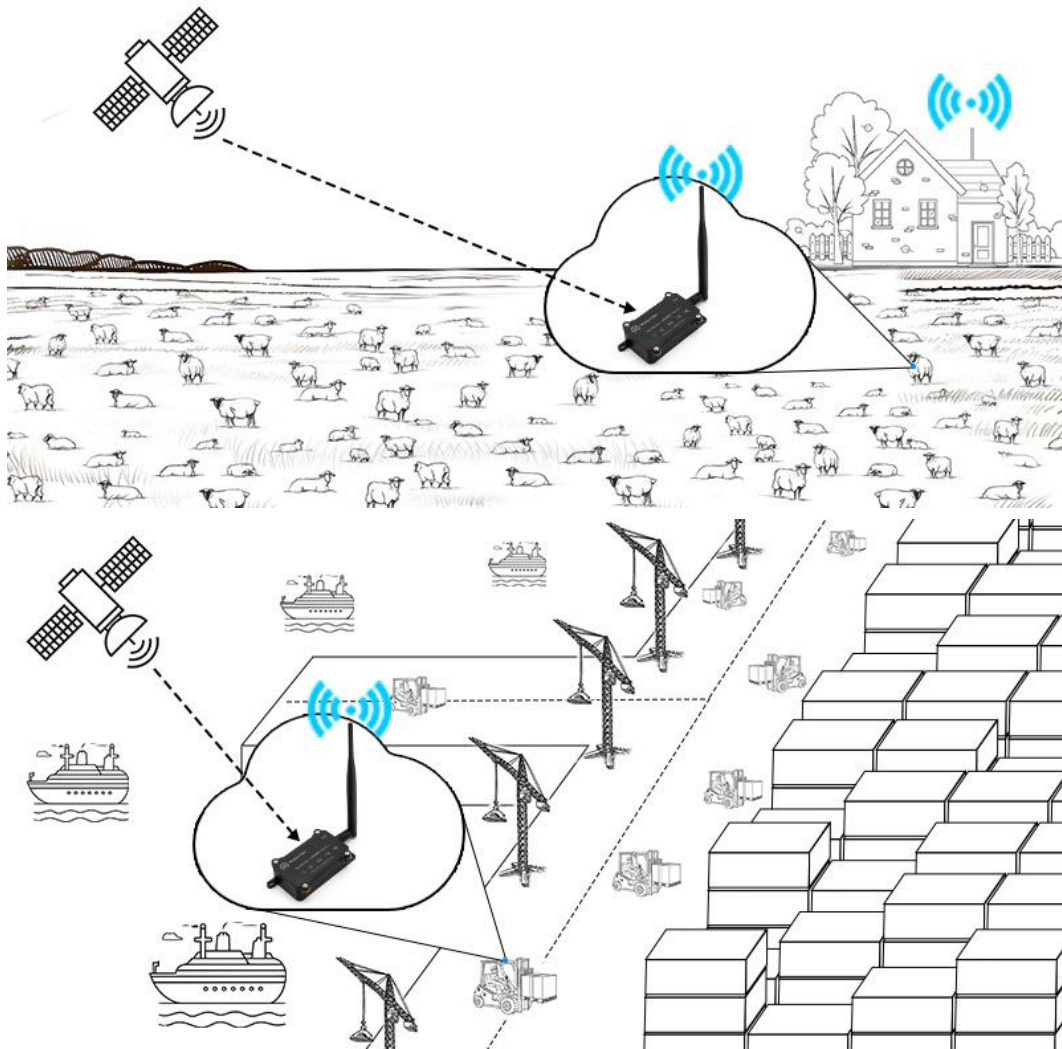
# Contents

<b>1 Product Description .....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Feature .....	2
1.3 Parameter .....	3
<b>2 Technical route .....</b>	<b>4</b>
2.1 System Framework .....	4
2.2 Regional frequency band .....	5
<b>3 Usage .....</b>	<b>6</b>
3.1 TTN and ThingSpeak .....	6
3.1.1 Network Server configuration .....	6
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 Change Time Interval (5-1440min) .....	14
3.2 Datacake .....	16
3.2.1 Change Time Interval and Sensor On/Off/Sensitivity .....	24

# 1 Product Description

## 1.1 Introduction

The AgroSense LoRaWAN® GPS Tracker NEO\_6M is an ideal GPS beacon for geolocation of equipment, it reports GPS location every 1 hour( by default, can be set from min 5 minute to max 24 hours). U-blox NEO-6M GPS module is used in this product, it is a widely used GPS module which features high precision. With onboard accelerometer LIS3DHTR, it detects vibration and reports via LoRaWAN, suitable for kinds of applications such as asset tracking, farm management. This module uses re-chargeable 18650 lipo battery, with our detailed hardware/software design, the working time can be 16 months (per the default setting) for each battery charge circle.



It can cover a transmission range of 2km in urban scenes and 5km in line-of-sight scenes while keeping lower power consumption during the transmission process. Together with a replaceable battery that supports up to 1 years of usage and an industrial IP68 enclosure. It supports at least  $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$  operating temperature and can be deployed in harsh environments. AgroSense is compatible with LoRaWAN® V1.0.3 protocol and can work with LoRaWAN® gateway.



## 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. GNSS

Product Model	AGLWGT02
positioning system	GPS, SBAS, PPP
positioning parameters	longitude: resolution of 0.0001
	latitude: resolution of 0.0001
	altitude: resolution of 1 meter
positioning accuracy	GPS: 2.5 meters
	SBAS: 2.0 meters
	2D horizontal position accuracy < 1 meter
	3D horizontal position accuracy < 2 meters

### 2.Device attitude

Detection content	Normal / shock
-------------------	----------------

### 3.Wireless Parameters

Communication Protocol	Standard LoRaWAN® protocol 1.0.3
Network Access/Operating Mode	OTAA Class A
MAX Transmit Power	21dBm
Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

### 4.Physical Parameters

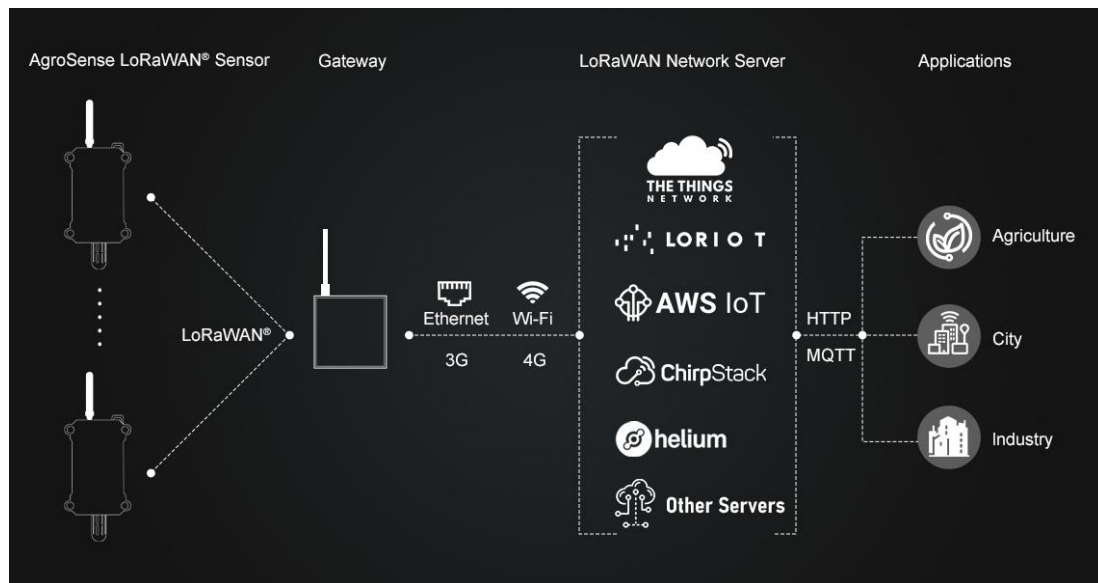
Power Supply	1 x 18650 3.7V Lion batteries
Operating Temperature	-40°C ~85°C
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

## 2 Technical route

### 2.1 System Framework

AgroSense\_GPS Tracker NEO\_6M 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 location information and shock status is:**

1. Collect the location information and motion status 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 location information and shock status 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.

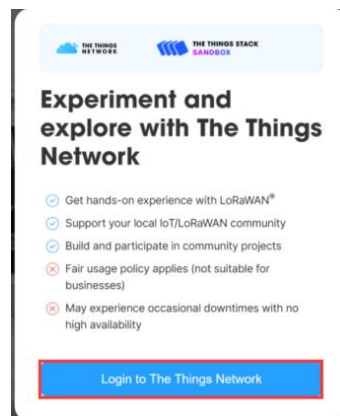
DEV EUI	Unique identification of device, authorized by IEEE
APP EUI	Unique identification of application
APP Key	One of the join network parameters on OTAA mode, calculated by DE EUI

- End Nodes and Gateway: AgroSense\_GPS Tracker NEO\_6M Sensor LoRaWAN®. (The AgroSense series is applicable)
- Network Server: The Things Network. ( Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

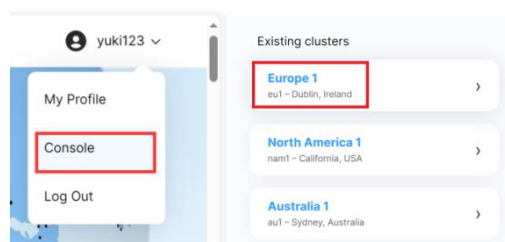
### 3.1 TTN and ThingSpeak

#### 3.1.1 Network Server configuration

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

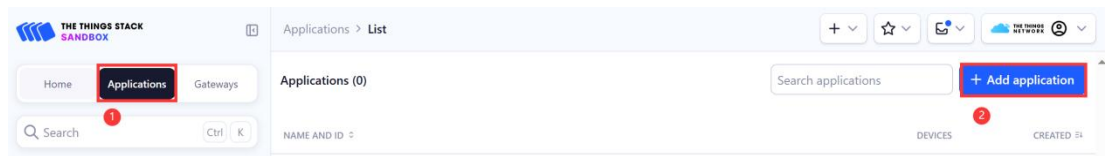


- Click “Console” and select clusters. (we take the European region for example.)





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



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

**Application ID \***

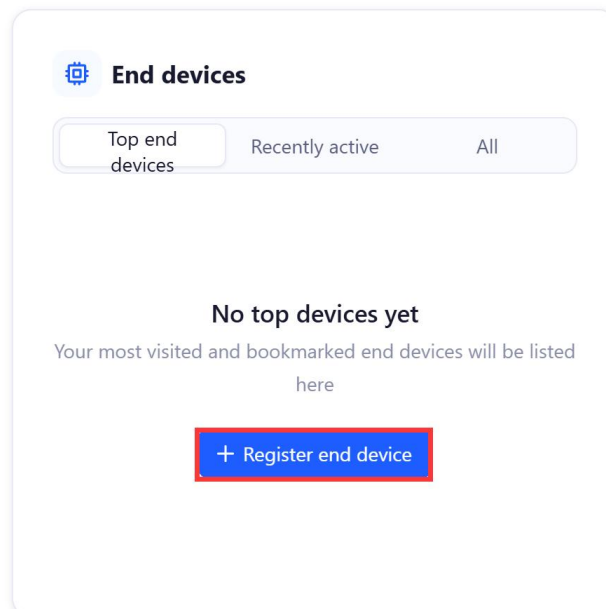
**Application name**

**Description**

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**

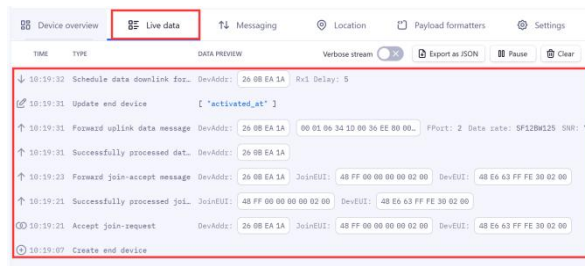
☒ View registered end device

☐ Register another end device of this type

**Register end device**

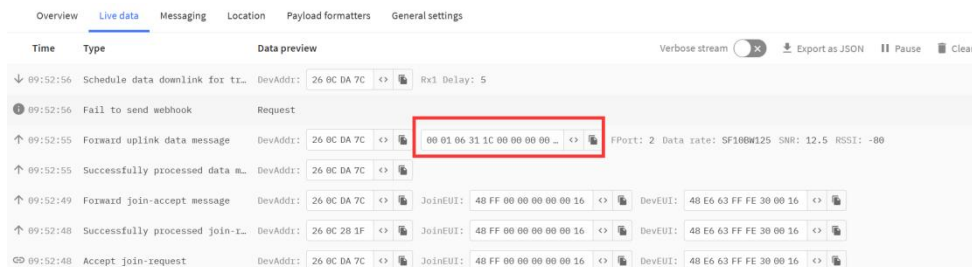


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



### 3.1.2 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	Shock status		Normal is 0, Shock is 1, Continuous Shock is 2
byte 4	GPS status		Invalid is 0, other is valid
byte 5	Year high 8 bits		For example, if the high 8 bits value is 0x07, and the low 8 bits value is 0xE8, then the obtained number of leaks is 0x07E8 = 2024. it is get number of year
byte 6	Year low 8 bits		

### AgroSense\_GPS Tracker NEO\_6M Sensor LoRaWAN®

			value is 2024.
<b>byte 7</b>	Month		Corresponding month
<b>byte 8</b>	Day		Corresponding day
<b>byte 9</b>	Hour		Corresponding hour
<b>byte 10</b>	Minute		Corresponding minute
<b>byte 11</b>	Second		Corresponding second
<b>byte 12</b>	Latitude bits 24 to 31		The value is obtained by amplifying the data by 100000 times, and the actual value needs to be divided by 100000 to convert to the actual latitude. The purpose of multiplying by 100000 is to retain one decimal place of the latitude value. For example, if the value is 0x05316A82 = 87124610, then the latitude value is 871.24610.
<b>byte 13</b>	Latitude bits 16 to 23		
<b>byte 14</b>	Latitude bits 8 to 15		
<b>byte 15</b>	Latitude bits 0 to 7		
<b>byte 16</b>	N/S sign		Units of latitude
<b>byte 17</b>	Longitude bits 24 to 31		The value is obtained by amplifying the data by 100000 times, and the actual value needs to be divided by 100000 to convert to the actual longitude. The purpose of multiplying by 100000 is to retain one decimal place of the longitude value. For example, if the value is 0x02326A18 = 36858392, then the longitude value is 368.58392.
<b>byte 18</b>	Longitude bits 16 to 23		
<b>byte 19</b>	Longitude bits 8 to 15		
<b>byte 20</b>	Longitude bits 0 to 7		
<b>byte 21</b>	E/W sign		Units of longitude
<b>byte 22</b>	Shock switch status		Off is 0, on is 1
<b>byte 23</b>	Shock sensitivity status		Sensitive is 16, Not so sensitive is 32

Example: 0x00, 0x06, 0x24, 0x00, 0x01, 0x07, 0xE8, 0x0A, 0x19, 0x08, 0x39, 0x10, 0x00, 0x22, 0x7F, 0x7E, 0x4E, 0x00, 0xAD, 0xB1, 0x89, 0x45, 0x00

Data parsing:

Battery voltage is 3.6 V.

Latitude value is 22.60862

Longitude value is 113.83177

Shock status is 0.(No shock)

- 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 batteryLevel = input.bytes[2] / 10.0;
  var gSensorState = input.bytes[3];
```

```

var gpsStatus = input.bytes[4];
var gsensor_onoff = input.bytes[22];
var gsensor_sensitivity = input.bytes[23];

if (gpsStatus != 0) {
    var year = input.bytes[5] * 256 + input.bytes[6];
    var month = input.bytes[7];
    var date = input.bytes[8];
    var hour = input.bytes[9];
    var minute = input.bytes[10];
    var second = input.bytes[11];

    var latitude = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 +
input.bytes[15]) / 100000;
    var nsHemi = input.bytes[16] === 0 ? 'N' : 'S';

    var longitude = (input.bytes[17] * 16777216 + input.bytes[18] * 65536 + input.bytes[19] * 256 +
input.bytes[20]) / 100000;
    var ewHemi = input.bytes[21] === 0 ? 'E' : 'W';
} else {
    var year = 0;
    var month = 0;
    var date = 0;
    var hour = 0;
    var minute = 0;
    var second = 0;
    var latitude = 0;
    var nsHemi = 'N';
    var longitude = 0;
    var ewHemi = 'E';
}

// Correct hemisphere values based on actual GPS data
nsHemi = latitude < 0 ? 'S' : 'N';
ewHemi = longitude < 0 ? 'W' : 'E';

// Convert latitude and longitude to positive values if necessary
latitude = Math.abs(latitude);
longitude = Math.abs(longitude);

return {
    data: {
        field1: batteryLevel,
        field2: latitude,

```

```

        field3: longitude,
        field4: gSensorState,
        field5: gsensor_onoff,
        field6: gsensor_sensitivity,
    },
};
}

```

- Select “Payload formatters” and follow the steps.

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

Uplink Downlink

Setup

Formatter type\* Custom Javascript formatter

Formatter code\*

```

function decodeUplink(input) {
    var num = input.bytes[0] * 256 + input.bytes[1];
    var batteryLevel = input.bytes[2] / 10.0;
    var gSensorState = input.bytes[3];
    var gpsStatus = input.bytes[4];
    var gsensor_onoff = input.bytes[22];
    var gsensor_sensitivity = input.bytes[23];

    if (gpsStatus !== 0) {
        var year = input.bytes[5] * 256 + input.bytes[6];
        var month = input.bytes[7];
        var date = input.bytes[8];
        var hour = input.bytes[9];
        var minute = input.bytes[10];
        var second = input.bytes[11];

        var latitude = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 + input.bytes[15]) / 100000;
        var nsHem1 = input.bytes[16] === 0 ? 'N' : 'S';

        var longitude = (input.bytes[17] * 16777216 + input.bytes[18] * 65536 + input.bytes[19] * 256 + input.bytes[20]) / 100000;
        var ewHem1 = input.bytes[21] === 0 ? 'E' : 'W';
    } else {
        var year = 0;
        var month = 0;
    }
}

```

Save changes

### 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)

MathWorks®

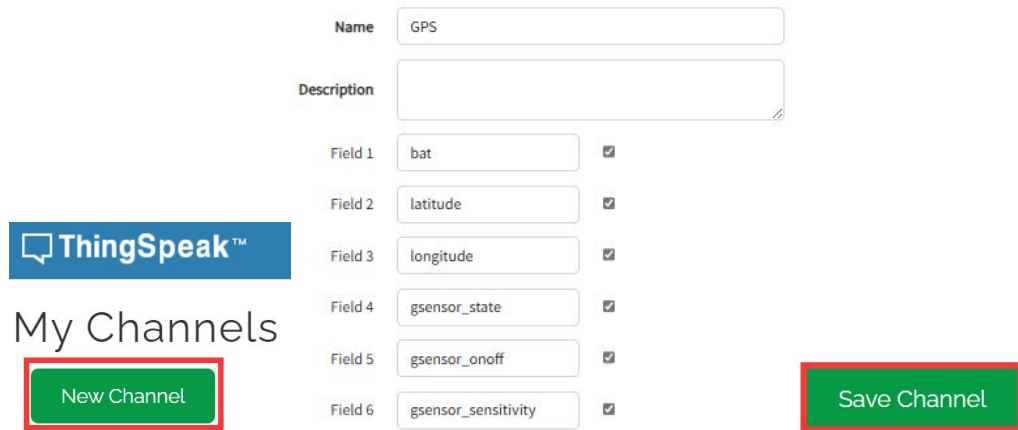
Email

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: GPS

Description:

Field 1: bat

Field 2: latitude

Field 3: longitude

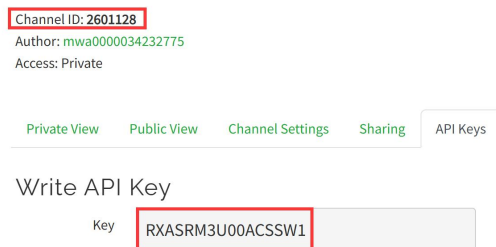
Field 4: gsensor\_state

Field 5: gsensor\_onoff

Field 6: gsensor\_sensitivity

Save Channel

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



Channel ID: 2601128

Author: mwa0000034232775

Access: Private

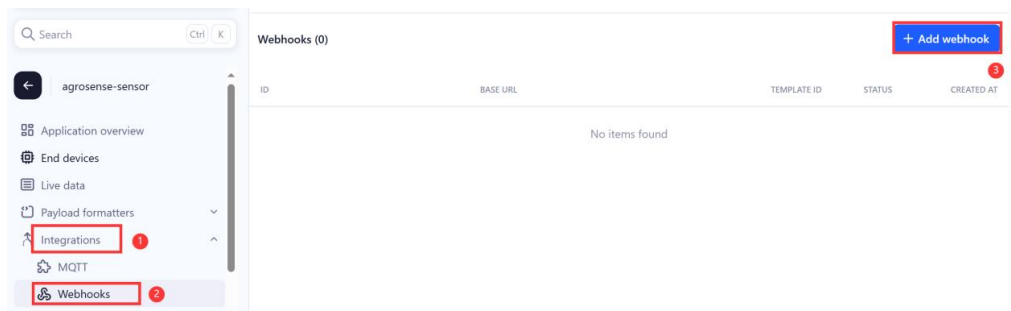
Private View Public View Channel Settings Sharing API Keys

Write API Key

Key: RXASRM3U00ACSSW1

### 3.1.4 Connect the Network Server and Application Server

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



Search

agrosense-sensor

Application overview

End devices

Live data

Payload formatters

Integrations

MQTT

Webhooks

Webhooks (0)

+ Add webhook

ID

BASE URL

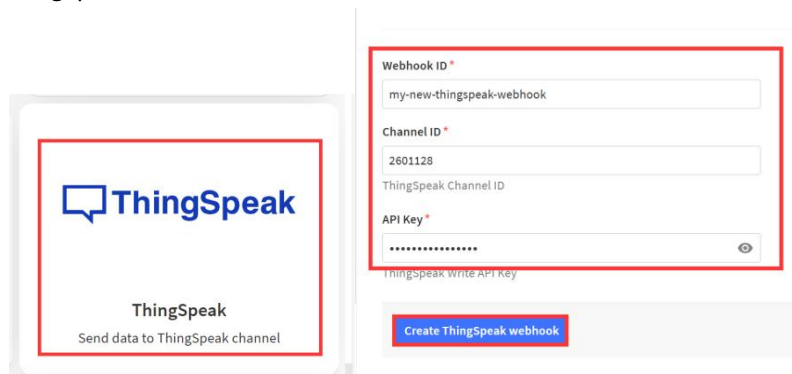
TEMPLATE ID

STATUS

CREATED AT

No items found

- 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 \*

my-new-thingspeak-webhook

Channel ID \*

2601128

ThingSpeak Channel ID

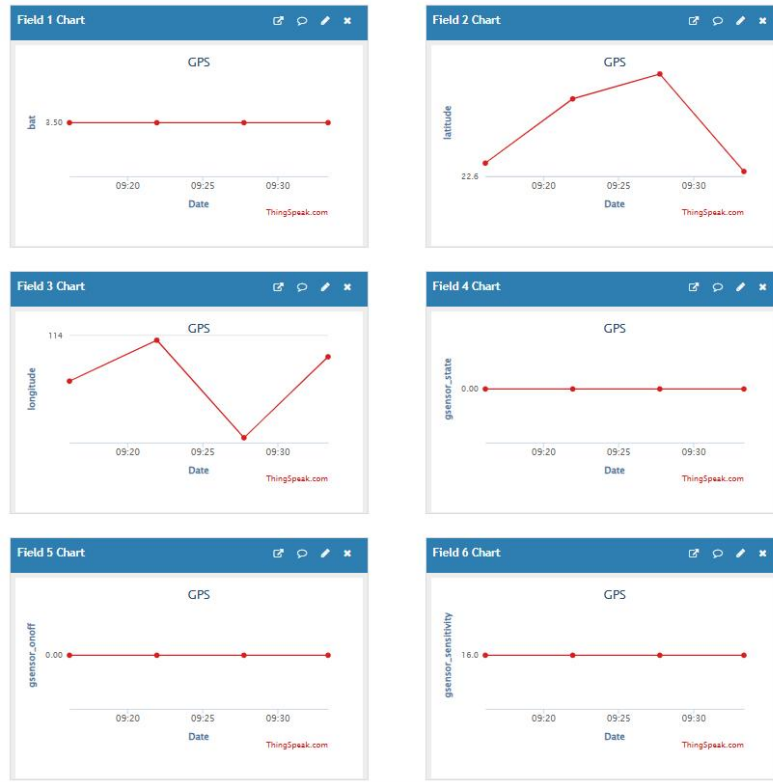
API Key \*

.....

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 Change Time Interval and Sensor Sensitivity/on/off

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' interface in ThingSpeak. The 'Downlink' tab is selected, and the 'Custom Javascript formatter' is chosen. The code is as follows:

```
// TTN Console downlink payload encoder function
function encodeDownlink(input) {
  var fPort = input.fPort;
  var payload = [];

  if (fPort === 1) {
    var minutes = input.data.minutes;
    // Convert minutes to seconds
    var seconds = minutes * 60;
    // If the number of seconds is less than 300, set it to 300 seconds
    if (seconds < 300) {
      seconds = 300;
    }
    // Create payload byte array
    payload = [
      (seconds >> 24) & 0xFF,
      (seconds >> 16) & 0xFF,
      (seconds >> 8) & 0xFF,
      seconds & 0xFF
    ];
  } else if (fPort === 3) {
    var gsensor_sensitivity = input.data.gsensor_sensitivity;
  }
}
```

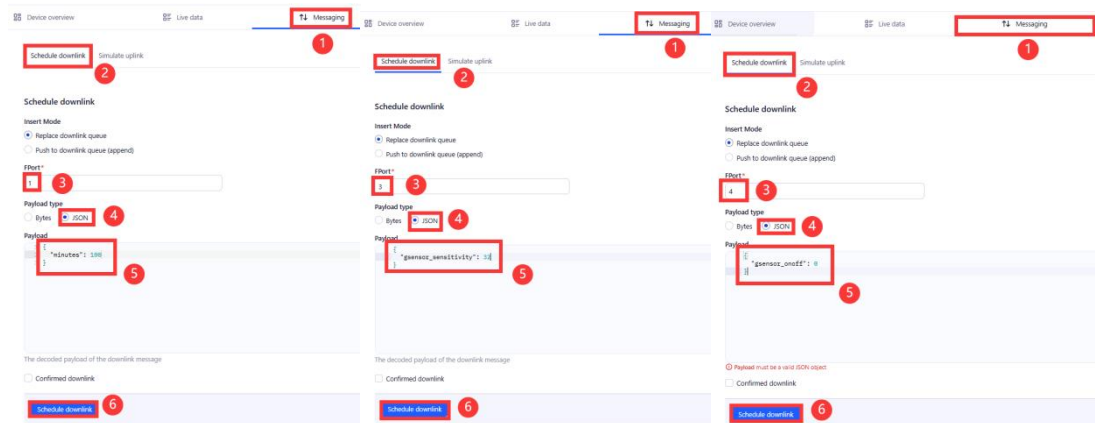


2、Click “Save changes”.

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

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

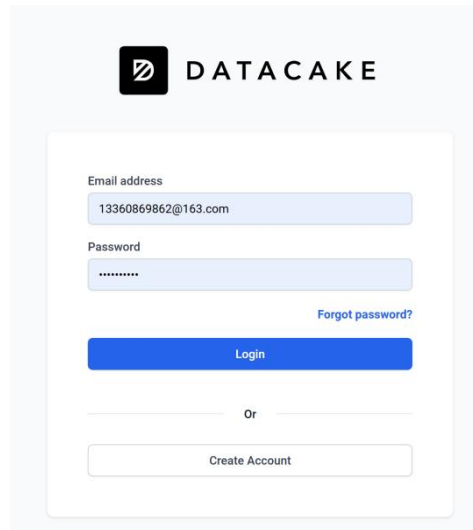
```
{
  "minutes": 5
}
{
  "gsensor_sensitivity": 32
}
{
  "gsensor_onoff": 0
}
```



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

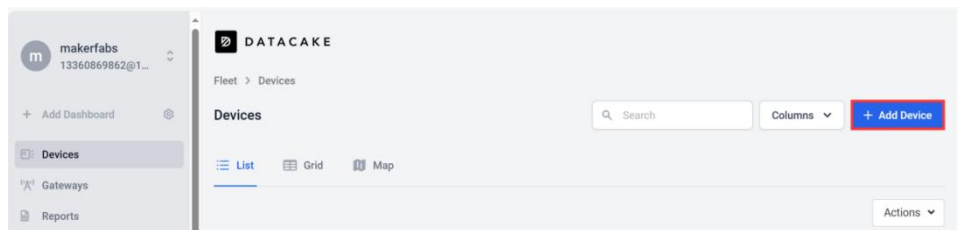
## 3.2 Datacake

### 1、Login datacake or Create Account

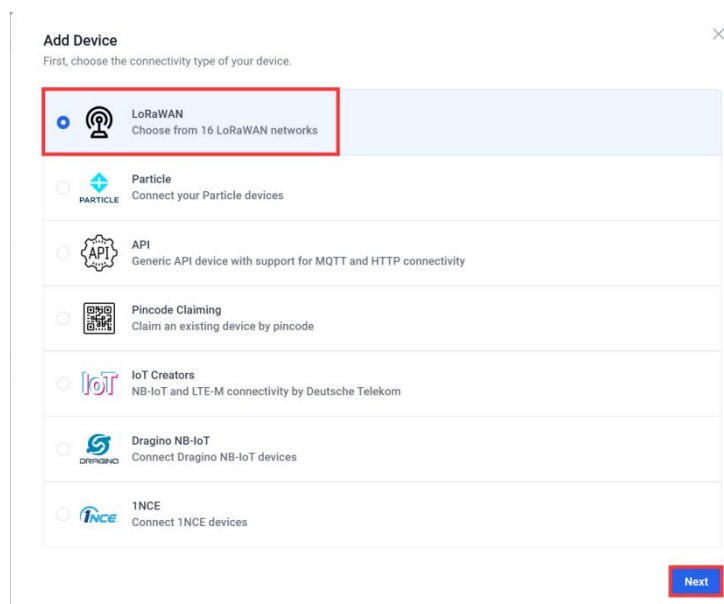


The image shows the Datacake login and registration interface. At the top is the Datacake logo. Below it is a form with two main sections: 'Email address' and 'Password'. The 'Email address' field contains the text '13360869862@163.com'. The 'Password' field is masked with dots. To the right of the password field is a link that says 'Forgot password?'. Below the password field is a blue 'Login' button. Below the 'Login' button is a horizontal line with the word 'Or' in the center. Below that is a 'Create Account' button.

### 2、Click “Add Device”



### 3、Select LoRaWAN and click “Next”



The image shows the 'Add Device' dialog box. At the top is the title 'Add Device' and a subtitle 'First, choose the connectivity type of your device.' Below this is a list of connectivity options. The first option, 'LoRaWAN', is selected and highlighted with a red box. It includes a radio button, an antenna icon, and the text 'Choose from 16 LoRaWAN networks'. The other options are 'Particle' (with a blue diamond icon), 'API' (with a gear icon), 'Pincode Claiming' (with a QR code icon), 'IoT Creators' (with an 'IoT' icon), 'Dragino NB-IoT' (with a 'S' icon), and '1NCE' (with an '1NCE' icon). At the bottom right is a blue 'Next' button highlighted with a red box.

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.

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](#))

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

    var num = input.bytes[0] * 256 + input.bytes[1];
    var batteryLevel = input.bytes[2] / 10.0;
    var gSensorState = input.bytes[3];
    var gpsStatus = input.bytes[4];
    var gsensor_onoff = input.bytes[22];
    var gsensor_sensitivity = input.bytes[23];

    var year, month, date, hour, minute, second, latitude, nsHemi, longitude, ewHemi, device_location;

    if (gpsStatus != 0) {
        year = input.bytes[5] * 256 + input.bytes[6];
        month = input.bytes[7];
        date = input.bytes[8];
        hour = input.bytes[9];
        minute = input.bytes[10];
        second = input.bytes[11];

        latitude = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 +
input.bytes[15]) / 100000;
        nsHemi = input.bytes[16] === 0 ? 'N' : 'S';

        longitude = (input.bytes[17] * 16777216 + input.bytes[18] * 65536 + input.bytes[19] * 256 +
input.bytes[20]) / 100000;
        ewHemi = input.bytes[21] === 0 ? 'E' : 'W';
    } else {
        year = 0;
        month = 0;
        date = 0;
        hour = 0;
        minute = 0;
        second = 0;
        latitude = 0;
        nsHemi = 'N';
        longitude = 0;
        ewHemi = 'E';
    }

    // Correct hemisphere values based on actual GPS data
    nsHemi = latitude < 0 ? 'S' : 'N';
}
```

```

ewHemi = longitude < 0 ? 'W' : 'E';

// Convert latitude and longitude to positive values if necessary
latitude = Math.abs(latitude);
longitude = Math.abs(longitude);

var decoded = {
    batteryLevel: batteryLevel,
    gSensorState: gSensorState,
    latitude: latitude,
    longitude: longitude,
    device_location: "(" + latitude + "," + longitude + ")"
    gsensor_onoff: gsensor_onoff,
    gsensor_sensitivity: gsensor_sensitivity
};

// 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: "batteryLevel", value: decoded.batteryLevel },
    { field: "gSensorState", value: decoded.gSensorState },
    { field: "latitude", value: decoded.latitude },
    { field: "longitude", value: decoded.longitude },
    { field: "device_location", value: decoded.device_location },
    { field: "gsensor_onoff", value: decoded.gsensor_onoff },
    { field: "gsensor_sensitivity", value: decoded.gsensor_sensitivity },
    { 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)

+ Add Field

**Fields**

Fields describe the data the device will store.

**Add Field**

Fields define the schema of the data the device stores.

Type Float 1

Name Latitude 2

Identifier LATITUDE 3

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 None

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

Formula  Optional

Formula 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

Cancel
Add Field 4

**Fields**

Fields describe the data the device will store.

NAME	IDENTIFIER	TYPE	ROLE
Batterylevel	BATTERYLEVEL	Float	Device Battery
Latitude	LATITUDE	Float	N/A
Longitude	LONGITUDE	Float	N/A
Device Location	DEVICE_LOCATION	Location	N/A
Gsensorstate	GSENSORSTATE	Integer	N/A
Gsensor Onoff	GSENSOR_ONOFF	Integer	N/A
Gsensor Sensitivity	GSENSOR_SENSITIVITY	Integer	N/A
Lora Rssi	LORA_RSSI	Integer	N/A
Lora Snr	LORA_SNR	Float	N/A
Lora Datarate	LORA_DATARATE	String	N/A

## 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

Live data

NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
Batterylevel	BATTERYLEVEL	Float	Device Battery	3.5	1 second ago
Latitude	LATITUDE	Float	N/A	22.61	0 seconds ago
Longitude	LONGITUDE	Float	N/A	113.83	0 seconds ago
Device Location	DEVICE_LOCATION	Location	N/A	22.488204, 113.818188	0 seconds ago
Gsensorstate	GSENSORSTATE	Integer	N/A	0	1 second ago
Gsensor Onoff	GSENSOR_ONOFF	Integer	N/A	0	0 seconds ago
Gsensor Sensitivity	GSENSOR_SENSITIVITY	Integer	N/A	16	0 seconds ago
Lora Rssi	LORA_RSSI	Integer	N/A	-69	0 seconds ago
Lora Snr	LORA_SNR	Float	N/A	11.5	0 seconds ago
Lora Datarate	LORA_DATARATE	String	N/A	SF7BW125.0	0 seconds ago

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

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

**DATA CAKE**

Fleet > agrosense sensor

**agrosense sensor**

Serial Number 48E663FFE30001F Last update Never

Dashboard History Downlinks Configuration Debug Rules Permissions

Public Link
+ Add Widget
Refresh

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 with the 'Value' widget selected in the left sidebar. The 'Basics' tab is active, showing the title 'Battery' and a value of '0.00'.

The second screenshot shows the 'Data' tab selected. The 'Field' is set to 'Batterylevel' and the 'Unit' is 'English'. The 'Decimal Places' are set to '2'.

The third screenshot shows the 'Gauge' tab selected. The 'Gauge Type' is set to 'Fill Level'. The 'Values' table is configured with the following data:

Value	Color
3.5	#f56565
4	#ecc94b
4.5	#38b2ac
5	#48bb78

The 'Gauge' tab also shows a preview of the gauge with a value of 3.50 and a red circle around the value. The 'Gauge' tab is highlighted with a red box and a red circle around the 'Gauge' tab label.

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

The first screenshot shows the 'Edit Value Widget' dialog with the 'Value' widget selected in the left sidebar. The 'Basics' tab is active, showing the title 'Shock' and a value of '0.00'.

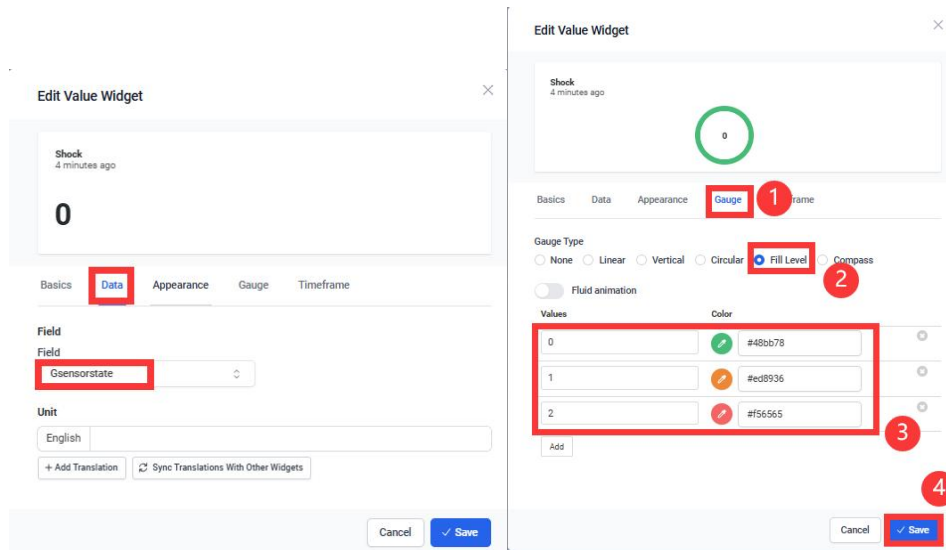
The second screenshot shows the 'Data' tab selected. The 'Field' is set to 'Shock' and the 'Unit' is 'English'. The 'Decimal Places' are set to '2'.

The third screenshot shows the 'Gauge' tab selected. The 'Gauge Type' is set to 'Fill Level'. The 'Values' table is configured with the following data:

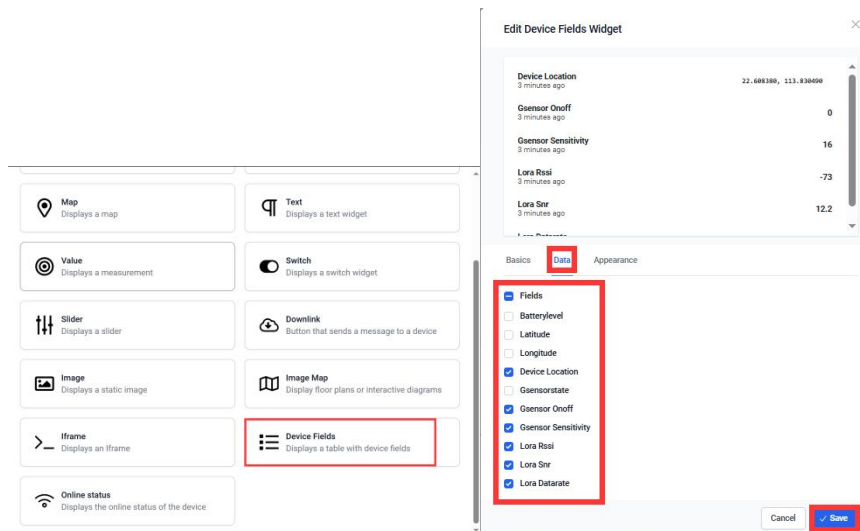
Value	Color
3.5	#f56565
4	#ecc94b
4.5	#38b2ac
5	#48bb78

The 'Gauge' tab also shows a preview of the gauge with a value of 3.50 and a red circle around the value. The 'Gauge' tab is highlighted with a red box and a red circle around the 'Gauge' tab label.

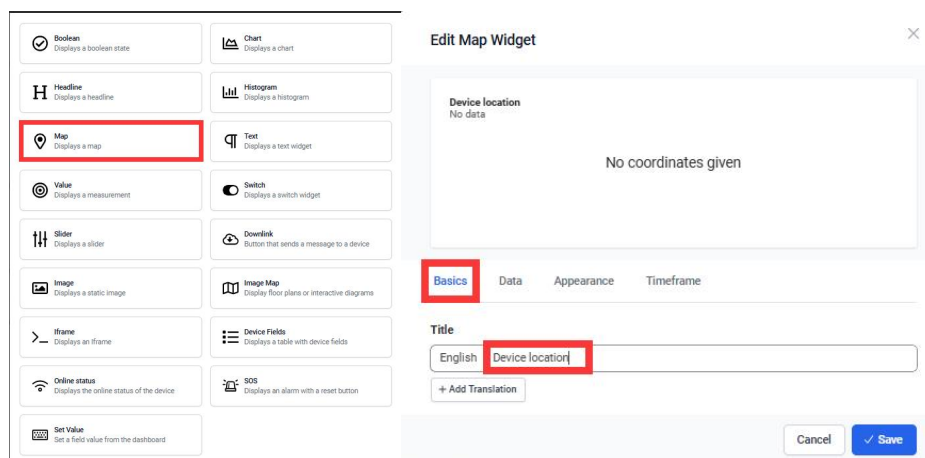


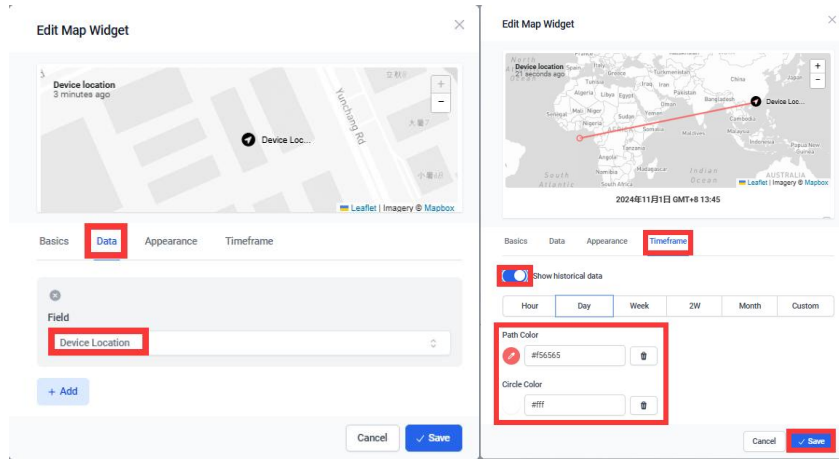


15、Select Device Fields, select the field you want to monitor and click “Save”.

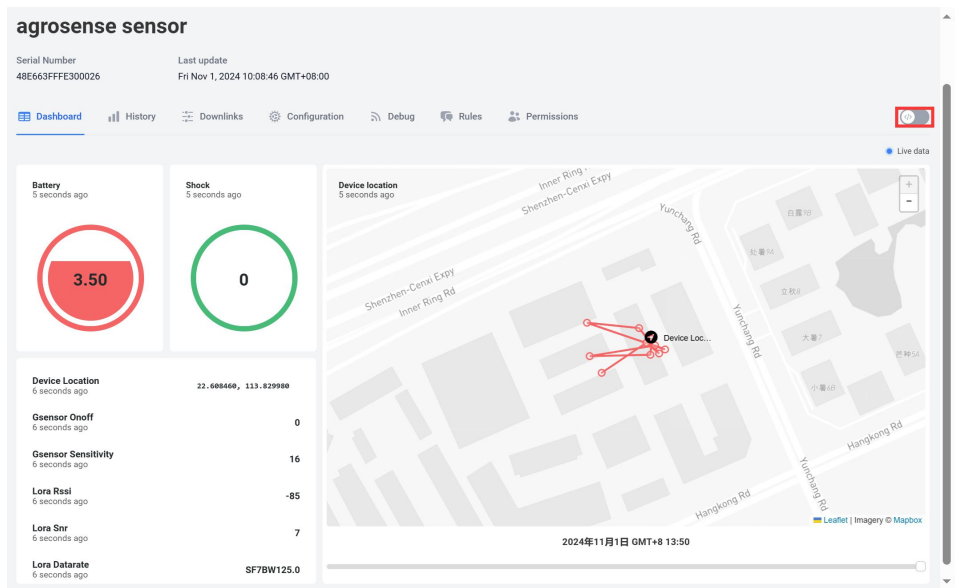


16、Select Map, and set Title, Field and color.





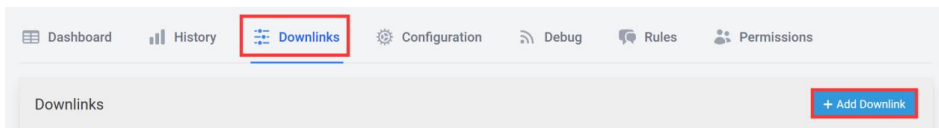
17、Click the switch to save, and you can see the data visually.



### 3.2.1 Change Time Interval and Sensor On/Off/Sensitivity

1、 If you need to change time Interval (Default 60 minutes), Shock Sensor on/off (Default on), Shock Sensor Sensitivity (Default 16), you can click “Configuration-->Fields-->+Add Field”.

## 2、Click "Downlink-->Add Downlink".



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

### Change time Interval:

Name: Set User-Defined Sending Time Interval

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

### Shock Sensor on/off:

Name: Change GSensor On/Off.

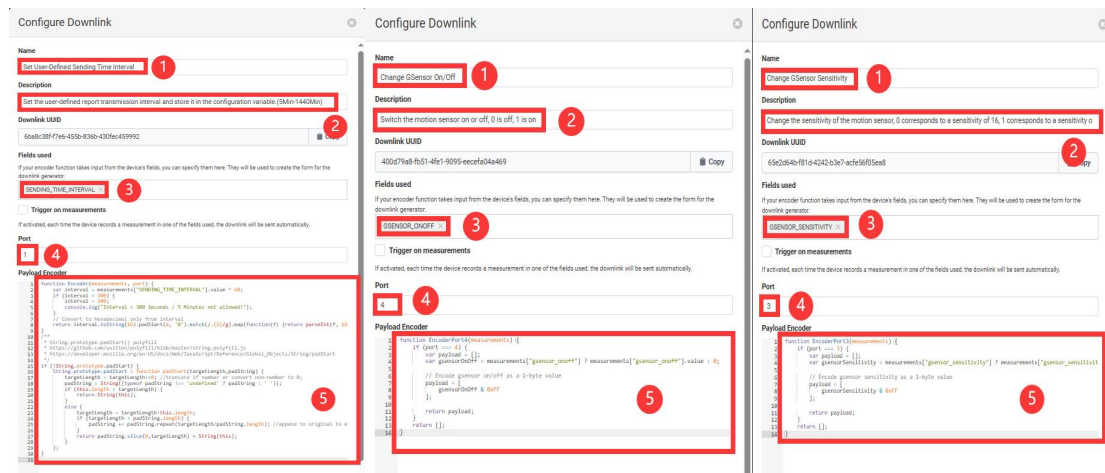
Description: Switch the motion sensor on or off, 0 is off, 1 is on.

### Shock Sensor Sensitivity:

Name: Change GSensor Sensitivity.

Description: Change the sensitivity of the motion sensor, 0 corresponds to a sensitivity of 16, 1 corresponds to a sensitivity of 32.

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



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

Select "Downlink" and setting as follow image. (Sensor On/Off/Sensitivity as the same way)

Edit Downlink Widget

User-Defined Time Interval(5Min-1440Min)

Basics

Data

Appearance

Title

English

User-Defined Time Interval(5Min-1440Min)

German

+ Add Translation

Cancel

Save

Edit Downlink Widget

User-Defined Time Interval(5Min-1440Min)

Basics

Data

Appearance

Downlink

Set User-Defined Sending Time Interval

Additional Downlinks

+ Add

Cancel

Save

4、Click the switch to save, and you can click to change your time Interval, shock sensor on/off/sensitivity.

GSensor On/Off (0 or 1)

User-Defined Time Interval (5-1440min)

Change GSensor Sensitivity (0 or 1)

Sending Time Interval

10

Cancel

Save measurements and send downlink

Gsensor Onoff

1

Cancel

Save measurements and send downlink

Gsensor Sensitivity

1

Cancel

Save measurements and send downlink

26