



AgroSense_Soil Monitor

LoRaWAN® Manual

V1.0

Author: Yuki

Time: 2025.02.05

Contents

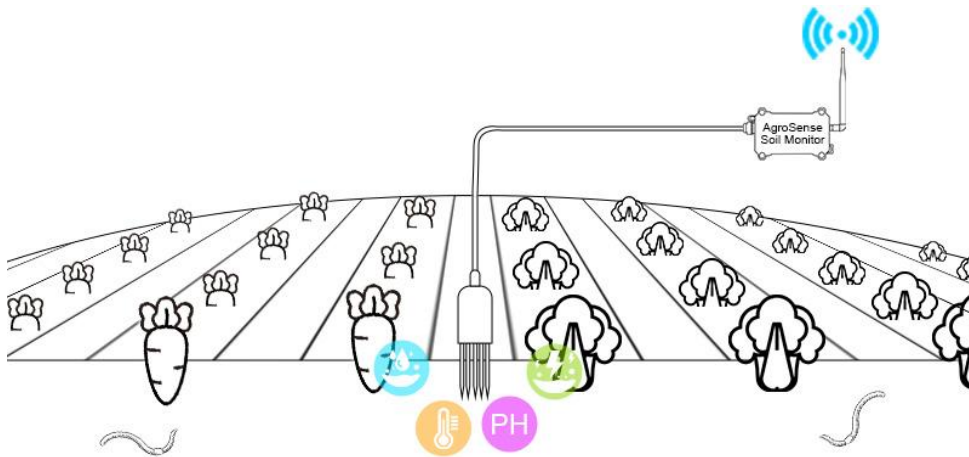
1 Product Description	1
1.1 Introduction	1
1.2 Feature	2
1.3 Parameter	2
2 Technical route	4
2.1 System Framework	4
2.2 Regional frequency band	5
3 Usage	6
3.1 TTN and ThingSpeak	6
3.1.1 Network Server configuration	6
3.1.2 Decoder	9
3.1.3 Application Server configuration	12
3.1.4 Connect the Network Server and Application Server	13
3.1.5 Downlink	14
3.2 Datacake	16
3.2.1 Downlink	25

1 Product Description

1.1 Introduction

This AgroSense LoRaWAN® soil monitor the key spec of soil, and transmit the result periodically via LoRaWAN to cloud server, for cloud remote monitoring, include : Soil Temperature, Soil Humidity, EC, PH.

This sensor reports the data to TTN/DataCake via LoRaWAN. It also stores max 3K results in internal flash so if LoRaWAN connection temporary not available for some reason It can store the data and resend them as LoRaWAN recover.



This Soil monitor is powered by 18650 lipo battery, with default setting (Soil status reporting every one hour), it can be used more than 9 month for each recharging around. Users can freely set the reports interval (1 hour by default) via Cloud server data downlink, from 5 minutes to 24 hours.

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	AGLWSM02
Temperature Measurement Range	-40°C ~80°C
Temperature Measurement Accuracy	±0.5°C
Temperature Resolution	0.1°C
Humidity Measurement Range	0%-100% RH
Humidity Measurement Accuracy	±2%
Humidity Resolution	0.1% RH
EC Measurement Range	0-20000μS/cm
EC Resolution	1μS/cm
PH Measurement Range	3 ~ 9PH

PH Resolution	0.1
---------------	-----

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

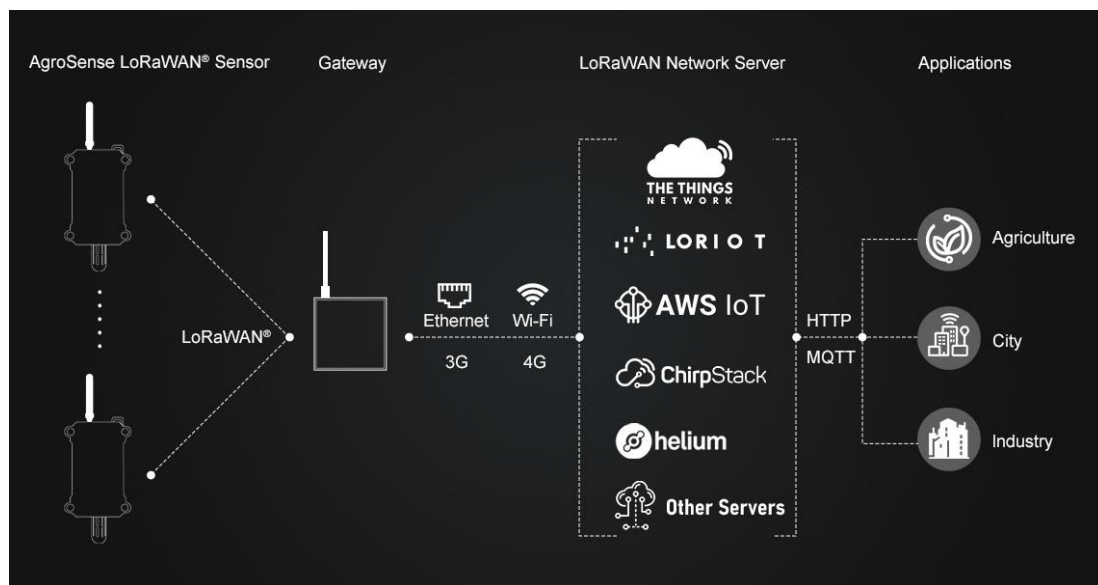
Power Supply	1 x 18650 3.7V Lion batteries
Operating Temperature	-40°C ~85°C
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

2 Technical route

2.1 System Framework

AgroSense Soil Monitor 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 soil is:

1. Collect the soil 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 soil 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 or Datacake 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.

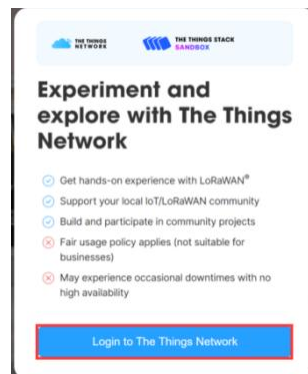
DEV EUI	Unique identification of device, authorized by IEEE
APP EUI	Unique identification of application
APP Key	One of the join network parameters on OTAA mode, calculated by DE EUI

- End Nodes and Gateway: AgroSense_Soil Monitor.(The AgroSense series is applicable)
- Network Server: The Things Network. (Datacake, Lorient, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbox, akenza, ect)

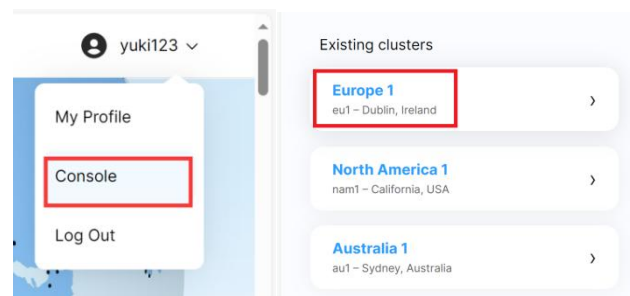
3.1 TTN and ThingSpeak

3.1.1 Network Server configuration

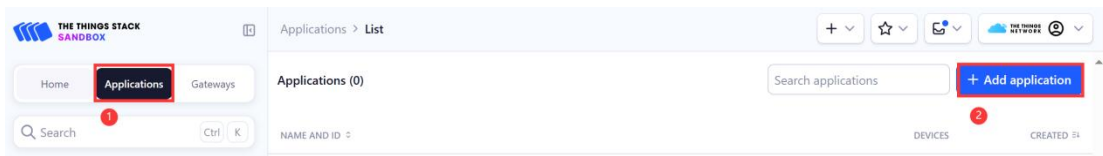
- Open The Things Network in your browser and login it. (Or register an account)



- 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 *

agrosense-sensor

Application name

My new application

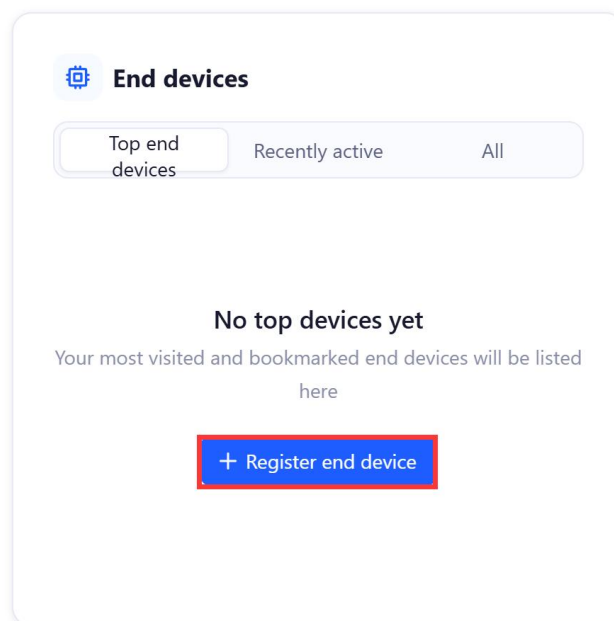
Description

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

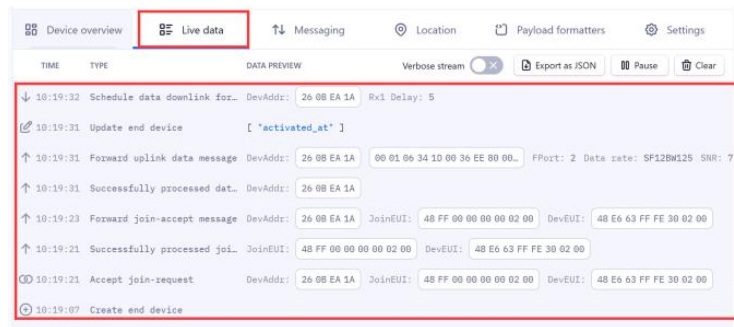
☒ View registered end device

☐ Register another end device of this type

Register end device

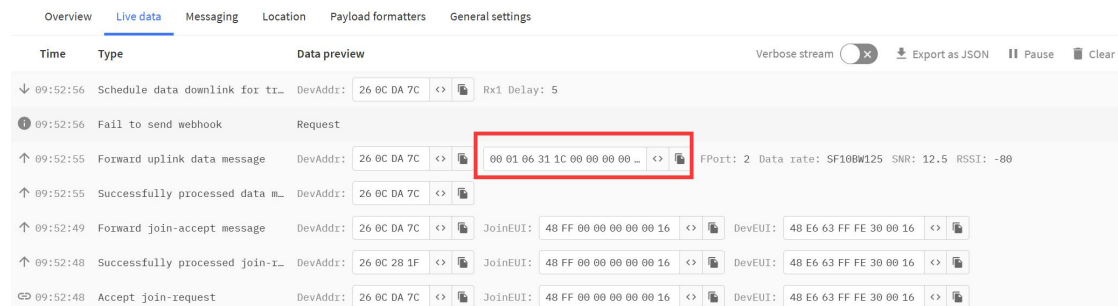


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



3.1.2 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	The data validity flag		0 is invalid, 1 is valid.

AgroSense_Soil Monitor LoRaWAN®

byte 4	Humidity sensor bits 8 to 15		This value is obtained after magnifying the data by 10 times. To obtain the actual relative humidity value, the real value needs to be calculated by dividing it by 10. For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the relative humidity value obtained is $0x00000285 = 645$. After converting and dividing by 10, the actual relative humidity is 64.5%RH.
byte 5	Humidity sensor bits 0 to 7		
byte 6	Temperature sensor bits 8 to 15		This value is obtained after magnifying the data by 10 times. To obtain the actual Temperature value, the real value needs to be calculated by dividing it by 10. For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the relative humidity value obtained is $0x00000285 = 645$. After converting and dividing by 10, the actual Temperature is 64.5°C.
byte 7	Temperature sensor bits 0 to 7		
byte 8	EC bits 8 to 15		For example, if the value from the 8th to the 15th bit is 0x02, and the lower 8 bits value is 0x85, then the EC value obtained is $0x00000285 = 645$, the EC is 645
byte 9	EC bits 0 to 7		
byte 10	PH bits 8 to 15		This value is obtained after magnifying the data by 10 times. To obtain the actual PH value, the real value needs to be calculated by dividing it by 10. For example, if the value from the 8th to the 15th bit is 0x00, and the lower 8 bits value is 0x3C, then the PH value obtained is $0x0000003C = 60$. After converting and dividing by 10, the PH is 6.
byte 11	PH bits 0 to 7		
byte 12	data transmission interval bits 24 to 31		The time interval for data transmission has been increased by a factor of 100. The unit is seconds.
byte 13	data transmission interval bits 16 to 23		
byte 14	data transmission interval bits 8 to 15		
byte 15	data transmission interval bits 0 to 7		
Fport 1	Change the data sending interval		
Fport 2	Upload the quantity of the latest local logged data		

Example: 0x00, 0x01, 0x28, 0x01, 0x00, 0x33, 0x01, 0x3F, 0x02, 0x85, 0x00, 0x3C, 0x00, 0x36, 0xEE, 0x80

Data parsing:

Battery voltage is 4V.

Humidity value is 5.1%.

Temperature value is 31.9°C.

EC value is 645.

PH value is 6.

- 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 Significant = input.bytes[3]
    var humi = (input.bytes[4] * 256 + input.bytes[5]) / 10.0
    var temp = input.bytes[6] * 256 + input.bytes[7]
    if (temp >= 0x8000) {
        temp -= 0x10000;
    }
    temp = temp / 10.0

    var ec = (input.bytes[8] * 256 + input.bytes[9])
    var ph = (input.bytes[10] * 256 + input.bytes[11]) / 10.0
    var interval = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 + input.bytes[15])
    / 1000

    if (Significant) {
        return {
            data: {
                field1: bat,
                field2: humi,
                field3: temp,
                field4: ec,
                field5: ph,
                field6: interval,
            },
        };
    }
    else {
        return {
            data: {
                Significant: "data invalid",
            },
        };
    }
}
```

- Select “Payload formatters” and follow the steps.

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

```
function decodeUplink(input) {
  // var num = input.bytes[0] * 256 + input.bytes[1]
  var bat = input.bytes[2] / 10.0
  var Significant = input.bytes[3]
  var humi = (input.bytes[4] * 256 + input.bytes[5]) / 10.0

  var temp = input.bytes[6] * 256 + input.bytes[7]
  if (temp >= 0x8000) {
    temp -= 0x10000;
  }
  temp = temp / 10.0

  var ec = (input.bytes[8] * 256 + input.bytes[9])
  var ph = (input.bytes[10] * 256 + input.bytes[11]) / 10.0
  var interval = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 + input.bytes[15]) / 1000

  if (Significant) {
    return {
      data: {
        field1: bat,
        field2: humi,
        field3: temp,
        field4: ec,
      }
    }
  }
}
```

A 'Save changes' button is located at the bottom of the configuration area.

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)

The screenshot shows the MathWorks login page. It includes an 'Email' input field, a 'Next' button, and links for 'No account? Create one!' and 'By signing in, you agree to our privacy policy.'

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

The screenshot shows the ThingSpeak 'My Channels' page. It includes a form to create a new channel with the following fields:

- Name: AgroSense Soil Monitor
- Description: (empty)
- Field 1: Bat
- Field 2: Humidity
- Field 3: Temperture
- Field 4: EC
- Field 5: PH
- Field 6: Time_interval

A 'New Channel' button is highlighted in green, and a 'Save Channel' button is also highlighted in green.

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

Channel ID: 2599652
 Author: mwa0000034232775
 Access: Private

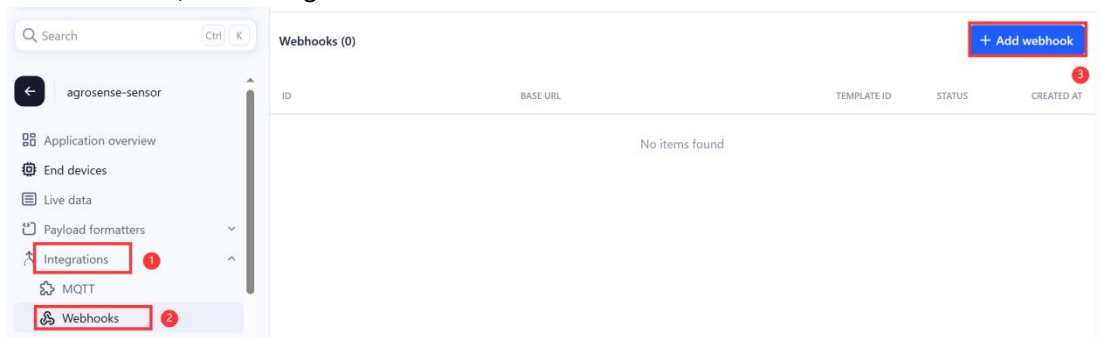
Private View Public View Channel Settings Sharing API Keys

Write API Key

Key N9IBFTBI3J36T779

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 *

2599652

ThingSpeak Channel ID

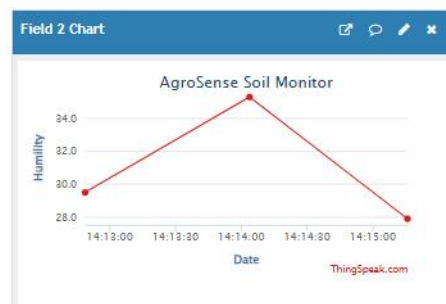
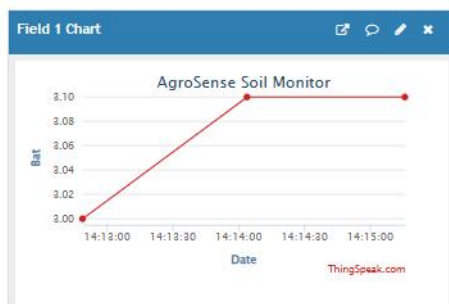
API Key *

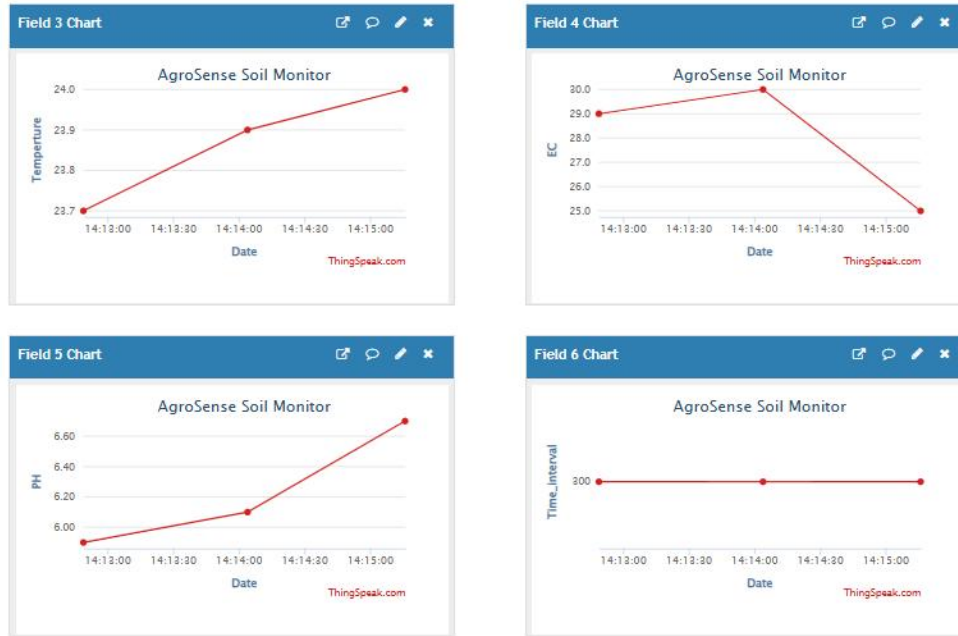
N9IBFTBI3J36T779

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.)





3.1.5 Downlink

The downlink has two functions:

- Modification time interval (Fport1)

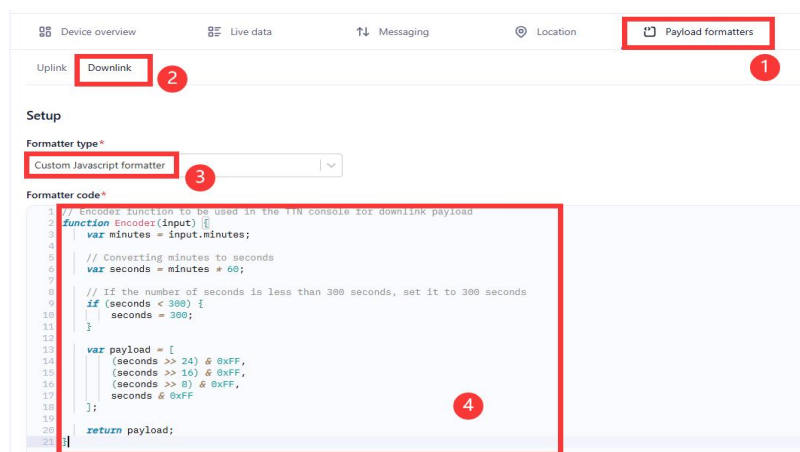
Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport2)

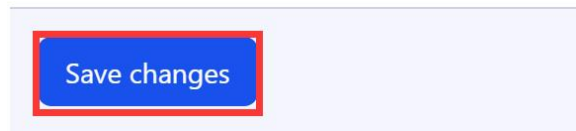
Users can view previous data based on this feature.

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

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

The decoded payload of the downlink message

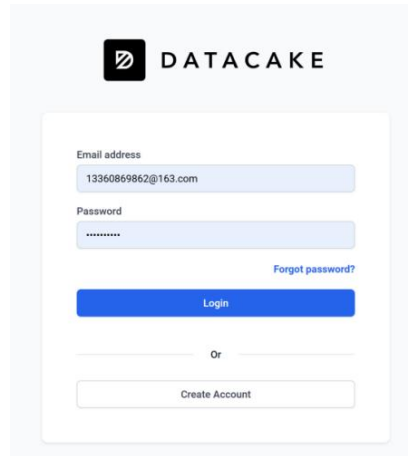
☐ Confirmed downlink

Schedule downlink

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

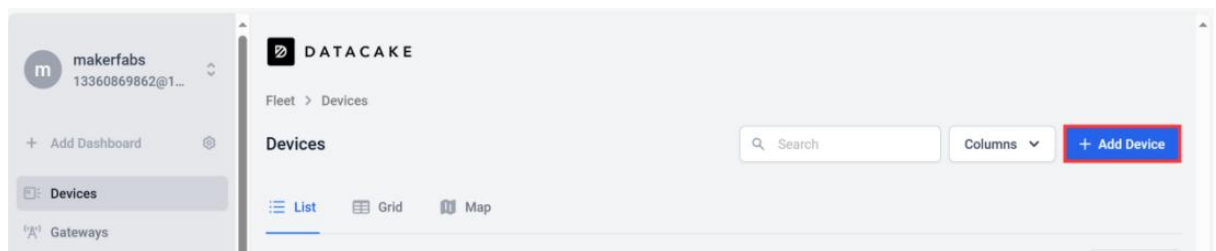
3.2 Datacake

1、Login datacake or Create Account

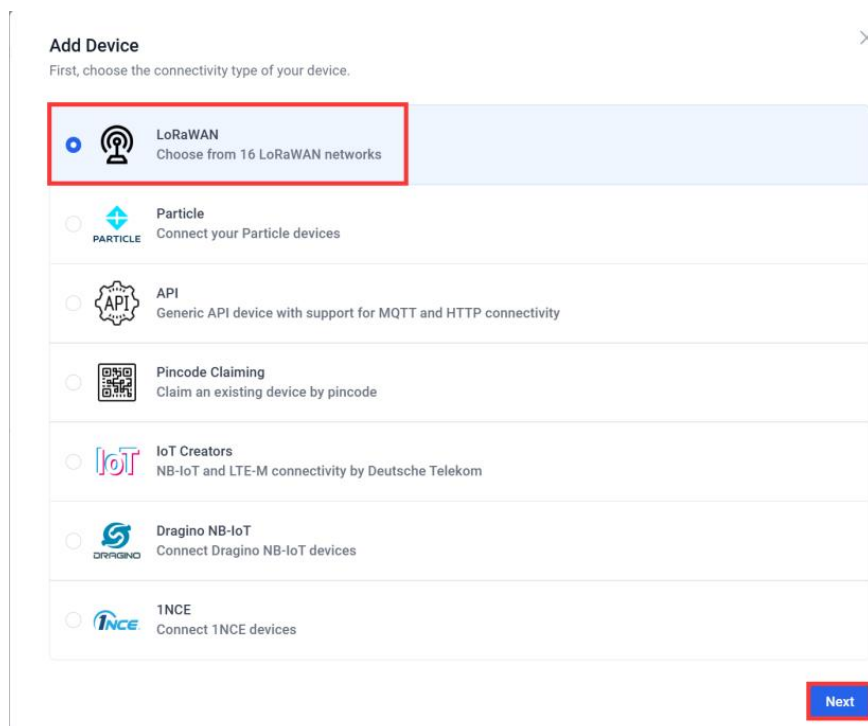


The image shows the Datacake login and registration interface. At the top is the Datacake logo. Below it is a form with two input fields: 'Email address' (containing '13360869862@163.com') and 'Password' (with masked characters). A 'Forgot password?' link is next to the password field. Below the fields 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 options, each with a radio button and an icon. The 'LoRaWAN' option is selected and highlighted with a red box. The other options are: Particle, API, Pincode Claiming, IoT Creators, Dragino NB-IoT, and 1NCE. At the bottom right is a blue '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

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

Back
Next

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

Uplinks
Downlinks

☐
The Things Stack V3
TTN V3 / Things Industries

Uplinks
Downlinks

☐
Helium
Use your own console

Uplinks
Downlinks

☐
LORIoT

Uplinks
Downlinks

☐
ChirpStack

Uplinks
Downlinks

☐
Activity

Uplinks
Downlinks

☐
KPN

Uplinks
Downlinks

Showing 1 to 6 of 15 results

Previous
Next

Back
Next

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

The screenshots show the 'Add LoRaWAN Device' form with the following fields filled in:

- DEVEUI:** 48 55 63 FF FE 30 00 1E
- NAME:** agrosense sensor
- APPEUI:** 48 FF 00 00 00 00 1E
- APPKEY 1.0 / NWKEY 1.1:** F3 FD 45 8F 9F 21 C7 C8 54 89 7A 48 C5 34 CC A9
- FREQUENCY:** United States 902.928 MHz, FSB 2
- DEVICE CLASS:** Class A

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

The pricing options are:

- Free:** €0.00 / month, 7 days data retention, 500 datapoints / day, max 5 per workspace, Cancel any time.
- Light:** €1.00 / month, 1 month data retention, 1,000 datapoints / day, Cancel any time.
- Standard:** €3.00 / month, 3 months data retention, 2,500 datapoints / day, Cancel any time.
- Plus:** €5.00 / month, 12 months data retention, 7,500 datapoints / day, Cancel any time.

The 'Add 1 device' button is highlighted in red.

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

DATA CAKE

Fleet > Devices

Devices

Search Columns + Add Device

List Grid Map

DEVICE	PRIMARY	SECONDARY	DEVICE SIGNAL	DEVICE BATTERY	Actions
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	👁️ ⋮ ⏏️

Showing 1 to 3 of 3 results

50 per page Previous Next

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

The screenshot shows the Datacake interface for the 'agrosense sensor'. The 'Configuration' tab is selected and highlighted with a red box. Below the configuration tabs, the 'Payload Decoder' section is visible. The decoder function code is shown in a text editor, with a red box highlighting the code block. The code defines a 'Decoder' function that takes 'payload' and 'port' as arguments and returns a decoded object. The decoded object contains 'bat', 'Significant', 'humi', 'temp', 'ec', 'ph', and 'interval' fields. The 'Save' button at the bottom right is also highlighted with a red box.

```
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 Significant = input.bytes[3]
  var humi = (input.bytes[4] * 256 + input.bytes[5]) / 10.0
  var temp = input.bytes[6] * 256 + input.bytes[7]
  if (temp >= 0x8000) {
    temp -= 0x10000;
  }
  temp = temp / 10.0
  var ec = (input.bytes[8] * 256 + input.bytes[9])
  var ph = (input.bytes[10] * 256 + input.bytes[11]) / 10.0
  var interval = (input.bytes[12] * 16777216 + input.bytes[13] * 65536 + input.bytes[14] * 256 +
input.bytes[15]) / 1000

  var decoded = {
    bat: bat,
    Significant: Significant,
```

```

        humi: humi,
        temp: temp,
        ec: ec,
        ph: ph,
        interval: interval,
    };

    // 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);
    }

    if (Significant) {
        return [
            { field: "bat", value: decoded.bat },
            { field: "humi", value: decoded.humi },
            { field: "temp", value: decoded.temp },
            { field: "ec", value: decoded.ec },
            { field: "ph", value: decoded.ph },
            { field: "interval", value: decoded.interval },
            { field: "lora_rssi", value: decoded.lora_rssi },
            { field: "lora_snr", value: decoded.lora_snr },
            { field: "lora_datarate", value: decoded.lora_datarate }
        ];
    }

    else {
        return [
            { field: "Significant", value: "data invalid" },
        ];
    }
}

```

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: Optional

Role: None

Formula: Optional

☐ Use Formula

Cancel Add Field

NAME	IDENTIFIER	TYPE
Lora Snr	LORA_SNR	Float
Lora Rssi	LORA_RSSI	Integer
Lora Datarate	LORA_DATARATE	String
Bat	BAT	Float
Humi	HUMI	Float
Temp	TEMP	Float
Ec	EC	Integer
Ph	PH	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](#)

[Live data](#)

NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
Lora Snr	LORA_SNR	Float	N/A	1.4	0 seconds ago
Lora Rssi	LORA_RSSI	Integer	N/A	-24	0 seconds ago
Lora Datarate	LORA_DATARATE	String	N/A	SF7BW125.0	0 seconds ago
Bat	BAT	Float	N/A	3	0 seconds ago
Humi	HUMI	Float	N/A	11.4	0 seconds ago
Temp	TEMP	Float	N/A	22.9	0 seconds ago
Ec	EC	Integer	N/A	22	0 seconds ago
Ph	PH	Float	N/A	7.5	0 seconds ago

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

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

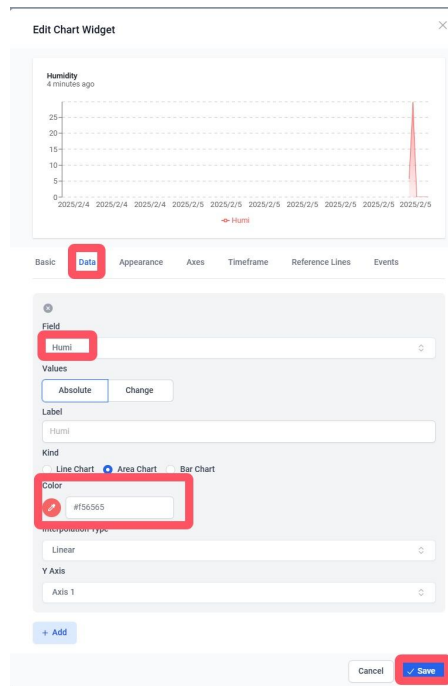


13、Select “Value” and set Title, Field and presentation form as well as the interval color.

The first screenshot shows the 'Edit Value Widget' dialog with the 'Value' widget selected in the left sidebar. The title is 'Humidity' and the field is 'Humidity'. The second screenshot shows the 'Gauge' tab selected, with 'Circular' gauge type and a color scale for values from 0 to 120.

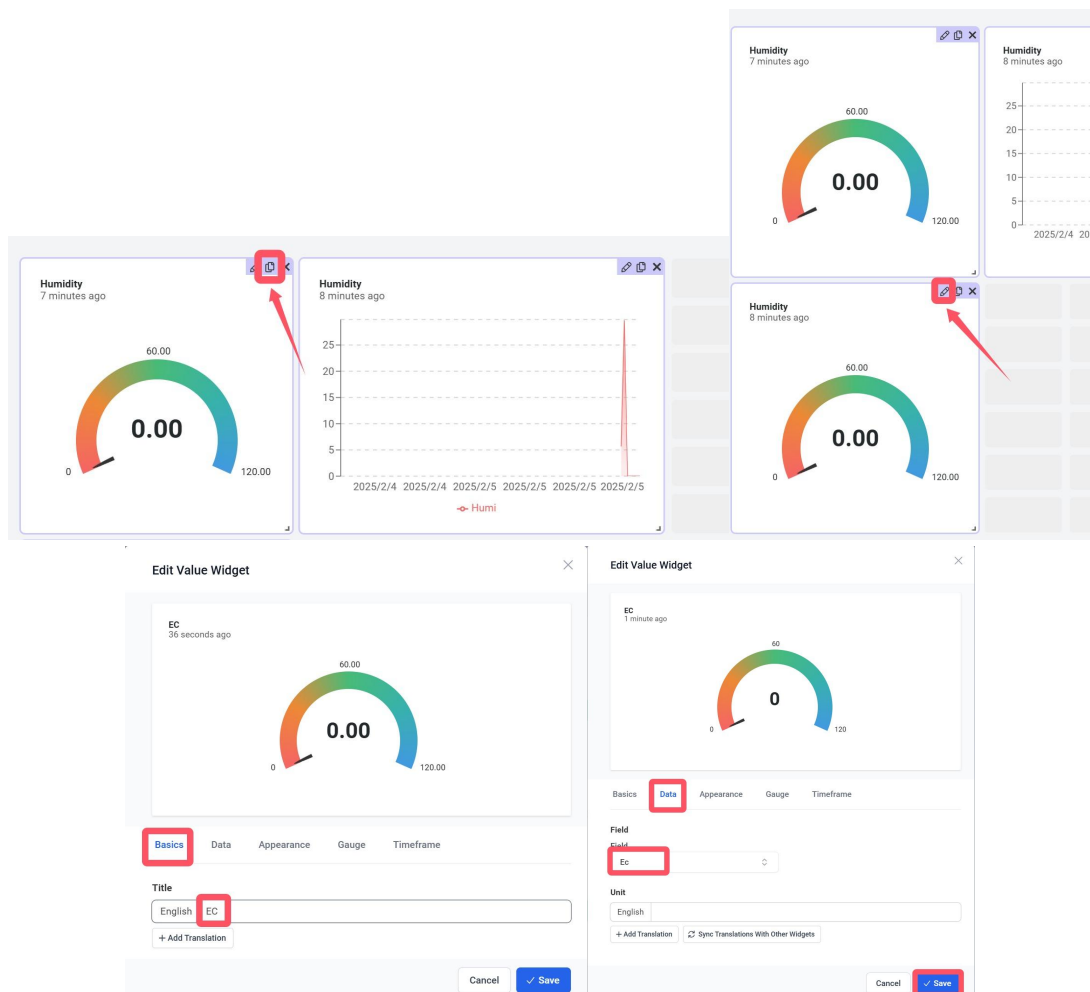
14、Select Chart and set Title, Field, Kind, Line Thickness and click “save”.

The first screenshot shows the 'Edit Chart Widget' dialog with the 'Chart' widget selected in the left sidebar. The title is 'Humidity' and the field is 'Humidity'. The second screenshot shows the 'Basic' tab selected, with 'Humidity' as the title and 'Humidity' as the field.



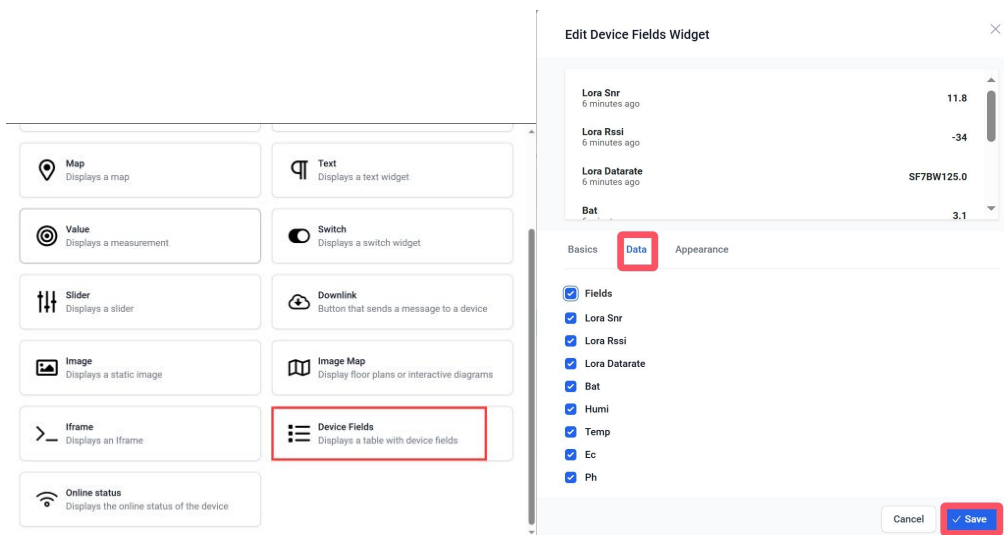
15、Click the Copy button and click Edit to change the name and channel to EC.

(Chart as the same way; You can add PH and temperature in the same way)





16、Select Device Fields, check “Fields” and click “Save”.



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



3.2.1 Downlink

The downlink has two functions:

- Modification time interval (Fport1)

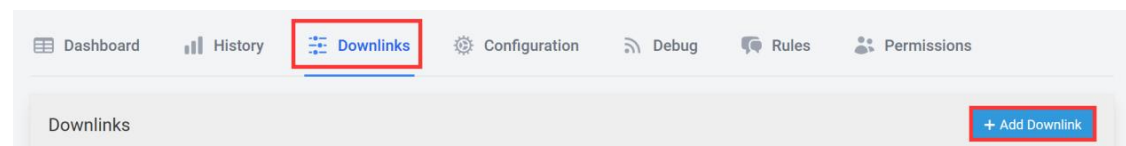
Modify the time interval for uploading data, the default is one hour.

- Upload the quantity of the latest local logged data (Fport2)

Users can view previous data based on this feature.

1 、 If you need to change time Interval (Default 60 minutes), you can click “Configuration-->Fields-->+Add Field”

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
Set User-Defined Sending Time Interval

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

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.
SENDING_TIME_INTERVAL

☐ 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
1

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 // String.prototype.padStart() polyfill
12 // https://github.com/uxitten/polyfill/blob/master/string.polyfill.js
13 // https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/padStart
14 //
15 if (!String.prototype.padStart) {
16   String.prototype.padStart = function padStart(targetLength,padString) {
17     targetLength = targetLength>0; //truncate if number or convert non-number to 0;
18     padString = String((typeof padString !== 'undefined' ? padString : ' '));
19     if (this.length > targetLength) {
20       return String(this);
21     }
22     else {
23       targetLength = targetLength-this.length;
24       if (targetLength > padString.length) {
25         padString = padString.repeat(targetLength/padString.length); //append to original to en
26       }
27     }
28   }
29 }

```

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

Select “Downlink” and setting as follow image.

Edit Downlink Widget

Basics | Data | Appearance

Title
English: User-Defined Time Interval(5Min-1440Min)
German: [dropdown]
+ Add Translation

Cancel **Save**

Edit Downlink Widget

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.

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

Sending Time Interval

1

Cancel **Save measurements and send downlink**