



**AgroSense_Temperature and Humidity
Sensor_SHT31 LoRaWAN® Manual
V1.0**

Author: Yuki

Time: 2024.10.12

Contents

1 Product Description	1
1.1 Introduction	1
1.2 Feature	1
1.3 Parameter	2
2 Technical route	3
2.1 System Framework	3
2.2 Regional frequency band	4
3 Usage	5
3.1 TTN and ThingSpeak	5
3.1.1 Network Server configuration	5
3.1.2 Decoder	8
3.1.3 Application Server configuration	10
3.1.4 Connect the Network Server and Application Server	11
3.1.5 Change Time Interval (5-1440min)	12
3.2 Datacake	13
3.2.1 Change Time Interval (5-1440min)	20

1 Product Description

1.1 Introduction

This AgroSense LoRaWAN® Temperature & Humidity sensor uses the SHT31 sensor, to measures air temperature & humidity at the range of -40°C to 125°C and 0 to 100 %RH, with accuracy $\pm 0.2^{\circ}\text{C}$ and $\pm 2\% \text{RH}$ 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.



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	AGLWTH02
Temperature Measurement Range	-40°C ~125°C
Temperature Measurement Accuracy	±0.2°C
Temperature Resolution	0.01°C
Humidity Measurement Range	0%-100% RH
Humidity Measurement Accuracy	±2%
Humidity Resolution	0.01% RH

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

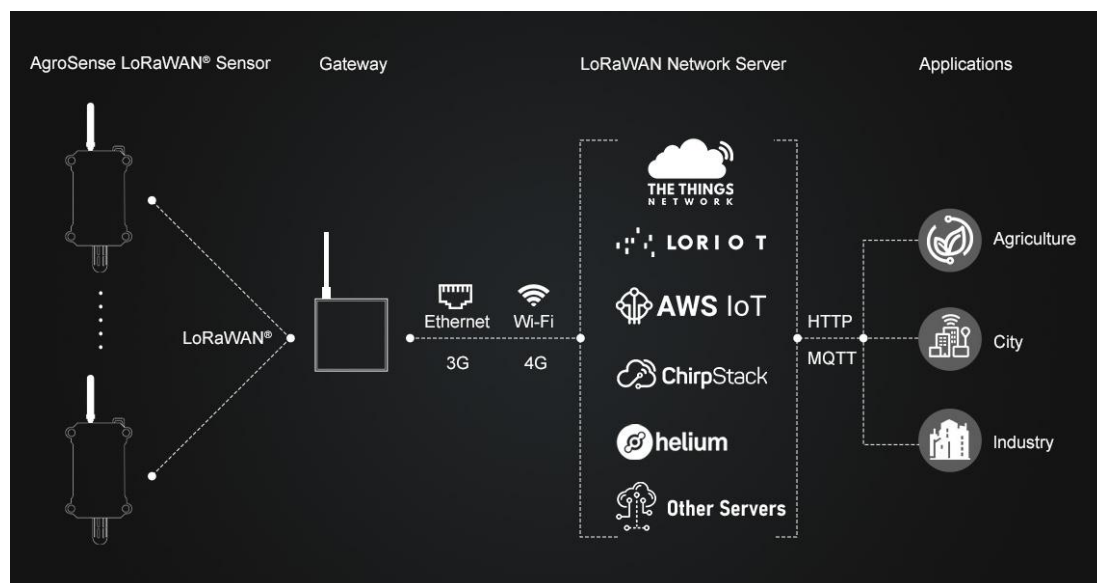
Power Supply	2 x AAA 1.5V batteries
Operating Temperature	-40°C ~85°C
Protection Class	IP68
Dimensions	131 × 62.7 × 27.5 mm
Mounting	Wall Mounting

2 Technical route

2.1 System Framework

AgroSense_Temperature and Humidity Sensor_SHT31 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 air(object surface) temperature and humidity is:

1. Collect the air(object surface) temperature and humidity 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 temperature and humidity 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.

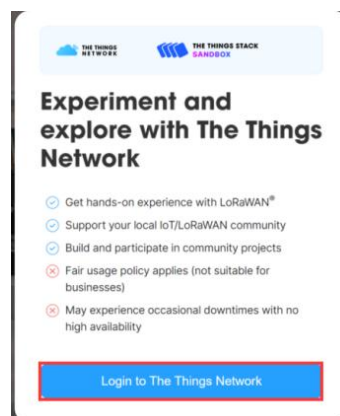
DEV EUI	Unique identification of device, authorized by IEEE
APP EUI	Unique identification of application
APP Key	One of the join network parameters on OTAA mode, calculated by DE EUI

- End Nodes and Gateway: AgroSense LoRaWAN® Temperature & Humility Sensor_SHT31. (The AgroSense series is applicable)
- Network Server: The Things Network. (Loriot, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

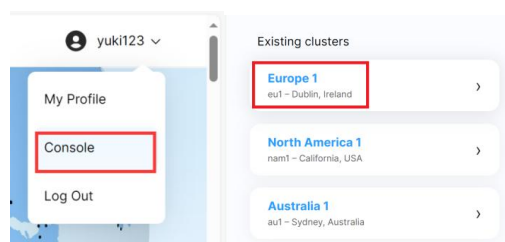
3.1 TTN and ThingSpeak

3.1.1Network Server configuration

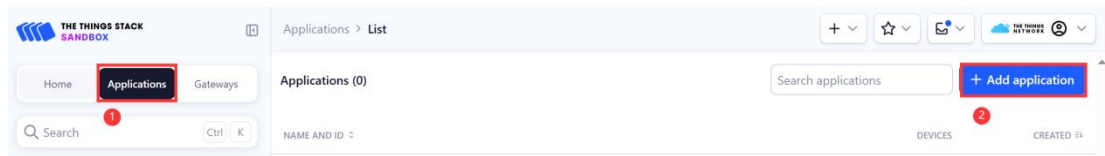
- Open The Things Network in your browser and login it. (Or register an account)



- 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-t-h-sensor

Application name

My new application

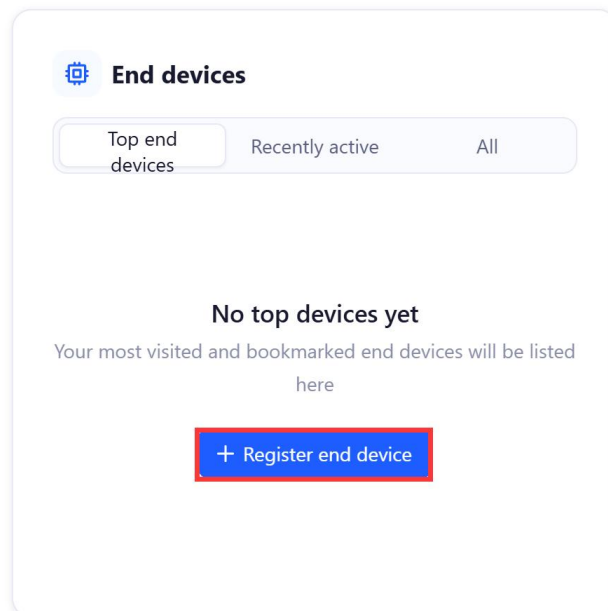
Description

Description for my new application

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

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

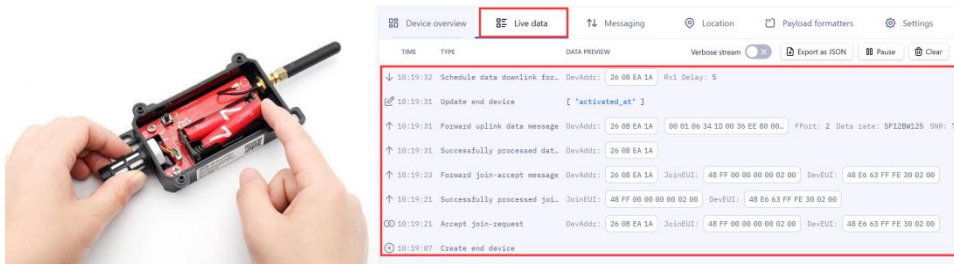
☒ View registered end device

☐ Register another end device of this type

Register end device

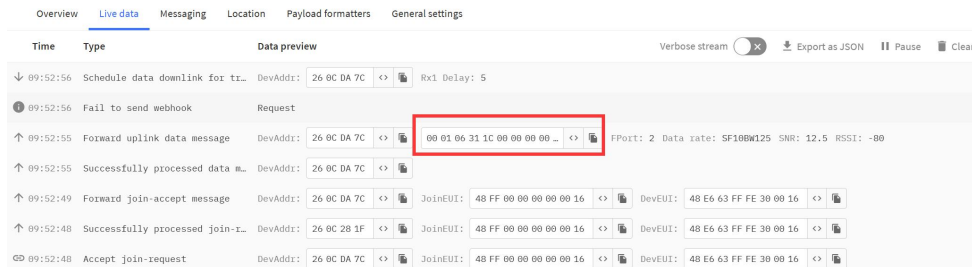


- Plug the battery and press RES button, you can see the device is connected successfully in the TTN.



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	Humidity sensor bits 8 to 15		This value is obtained by amplifying the data by 10 times. To obtain the actual relative Humidity value, the relative value needs to be converted by dividing by 10. For example, if the value of bits 8 to 15 is 0x02, and the low 8 bits value is 0x85, then the obtained temperature value is 0x0285 = 645. it is divided by 10 to get 64.5%RH.
byte 4	Humidity sensor bits 0 to 7		
byte 5	Temperature sensor bits 8 to 15		This value is obtained by amplifying the data by 10 times. To obtain the actual relative Temperature value, the relative value needs to be converted by dividing by 10. For example, if the value of bits 8 to 15 is 0x02, and the low 8 bits value is 0x85, then the obtained temperature value is 0x0285 = 645. it is divided by 10 to get 64.5℃.
byte 6	Temperature sensor bits 0 to 7		
byte 7	NC		
byte 8	NC		

Example: 0x00, 0x04, 0x1D, 0x01, 0x89, 0x00, 0xF8, 0x00, 0x00

Data parsing:

Battery voltage is 2.9V.

Humidity is 39.3%

Temperature is 24.8 °C.

- 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 humi = (input.bytes[3] * 256 + input.bytes[4]) / 10.0
  var temp = input.bytes[5] * 256 + input.bytes[6]
  if (temp >= 0x8000) {
    temp -= 0x10000;
  }
  temp = temp / 10.0
  return {
    data: {
      field1: bat,
      field2: humi,
      field3: temp,
    },
  };
}
```

- Select “Payload formatters” and follow the steps.

The screenshot shows the 'Payload formatters' tab in the LoRaWAN interface. The 'Uplink' section is active. Under 'Setup', the 'Formatter type' is set to 'Custom Javascript formatter'. The 'Formatter code' field contains the JavaScript function defined in the previous block. A 'Save changes' button is visible at the bottom.

Device overview Live data Messaging Location Payload formatters

Uplink Downlink

Setup

Formatter type*
Custom Javascript formatter

Formatter code*

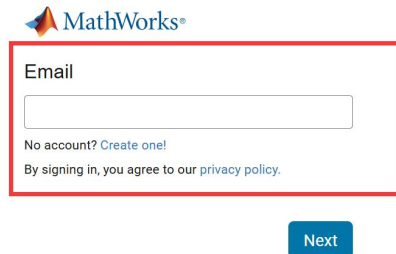
```
function decodeUplink(input) {
  // var num = input.bytes[0] * 256 + input.bytes[1]
  var bat = input.bytes[2] / 10.0
  var humi = (input.bytes[3] * 256 + input.bytes[4]) / 10.0
  var temp = input.bytes[5] * 256 + input.bytes[6]
  if (temp >= 0x8000) {
    temp -= 0x10000;
  }
  temp = temp / 10.0
  return {
    data: {
      field1: bat,
      field2: humi,
      field3: temp,
    },
  };
}
```

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®

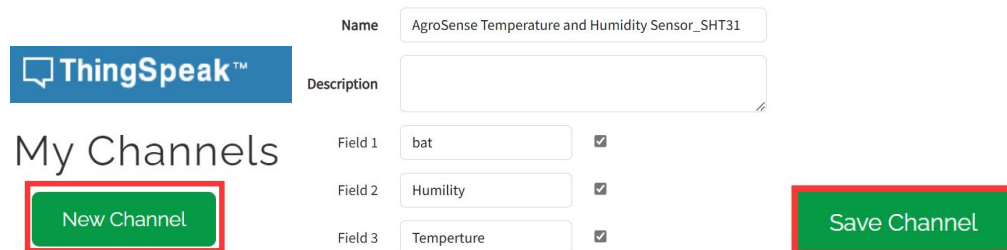
Email

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: AgroSense Temperature and Humidity Sensor_SHT31

Description:

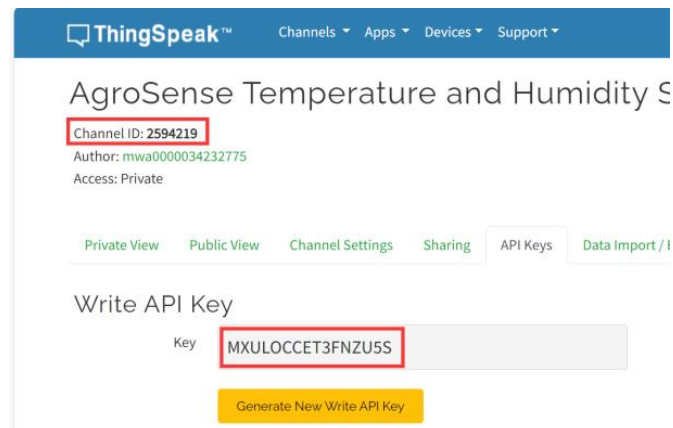
Field 1: bat ☒

Field 2: Humidity ☒

Field 3: Temperature ☒

Save Channel

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



ThingSpeak™ Channels Apps Devices Support

AgroSense Temperature and Humidity S

Channel ID: 2594219

Author: mwa0000034232775

Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / t

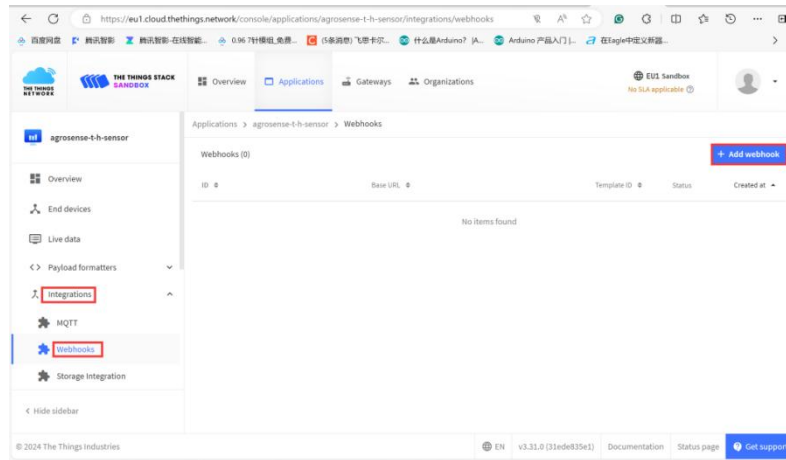
Write API Key

Key: MXULOCCE3FNZU5S

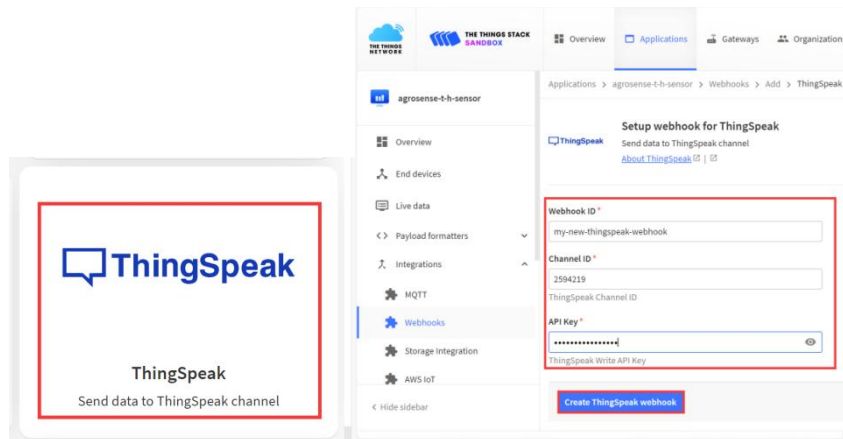
Generate New Write API Key

3.1.4 Connect the Network Server and Application Server

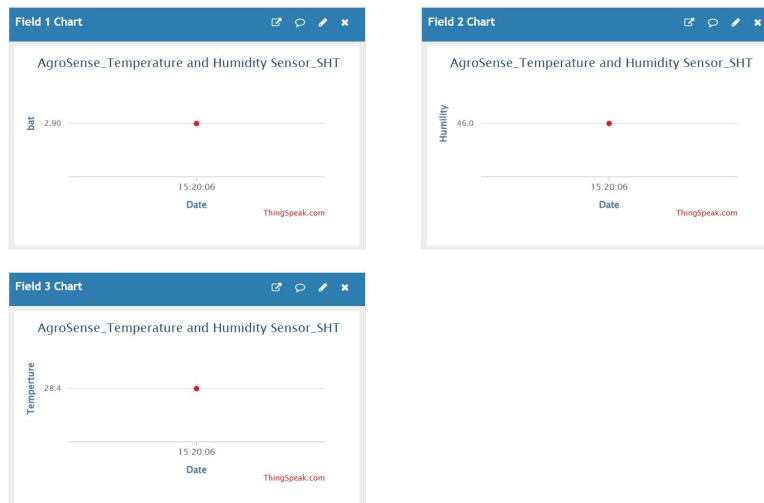
- In the TTN, click “integrations” --> “Webhooks” --> “+ Add webhook”.



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



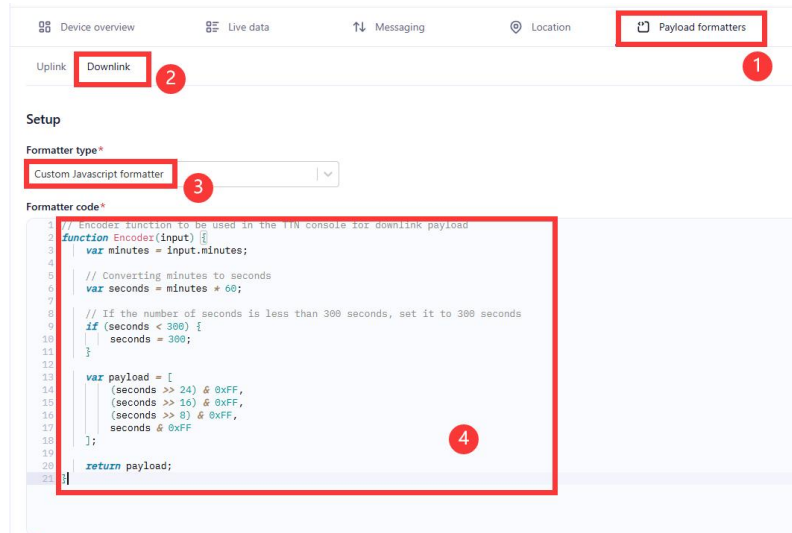
- Press RES button, wait about a minute, you will successfully see the data in ThingSpeak.(You will receive the data every hour.)



3.1.5 Change Time Interval (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](#).

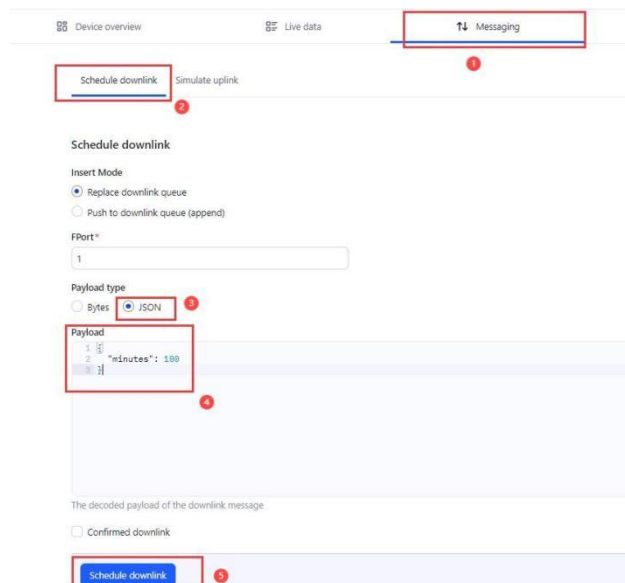


2、Click “Save changes”.

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

Note: you must use this format:

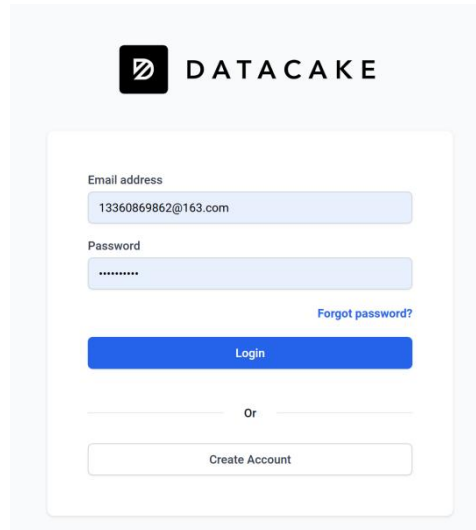
```
{
  "minutes": 5
}
```



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

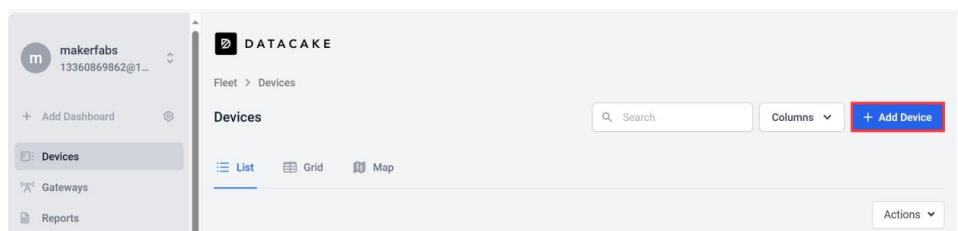
3.2 Datacake

1、Login datacake or Create Account

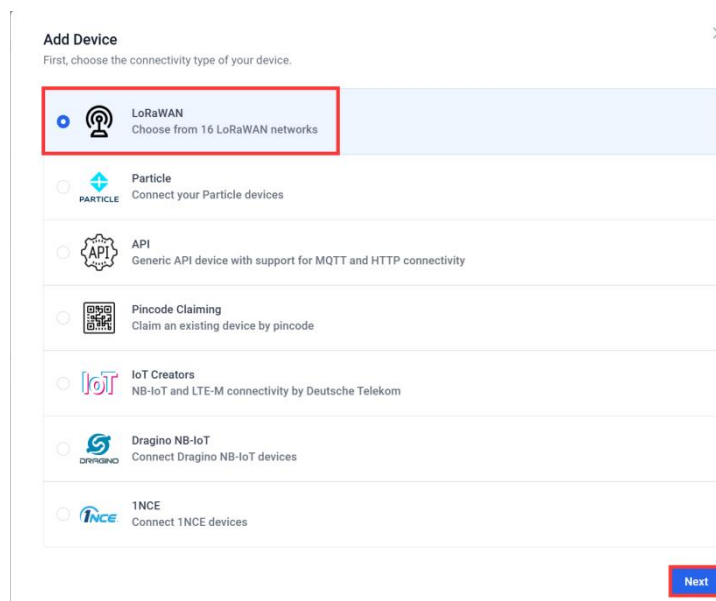


The image shows the Datacake login and registration interface. At the top is the Datacake logo. Below it is a form with two main sections. The first section is for login, with fields for 'Email address' (containing '13360869862@163.com') and 'Password' (masked with dots). There is a 'Forgot password?' link next to the password field. A blue 'Login' button is below these fields. The second section is for registration, separated by a horizontal line and the word 'Or'. It contains a 'Create Account' button.

2、Click “Add Device”



3、Select LoRaWAN and click “Next”



The image shows the 'Add Device' dialog box. The title is 'Add Device' with a close button. Below the title is the instruction 'First, choose the connectivity type of your device.' There is a list of connectivity options, each with a radio button and an icon. The first option, 'LoRaWAN', is selected and highlighted with a red box. It says 'Choose from 16 LoRaWAN networks'. Other options include 'Particle' (Connect your Particle devices), 'API' (Generic API device with support for MQTT and HTTP connectivity), 'Pincode Claiming' (Claim an existing device by pincode), 'IoT Creators' (NB-IoT and LTE-M connectivity by Deutsche Telekom), 'Dragino NB-IoT' (Connect Dragino NB-IoT devices), and '1NCE' (Connect 1NCE devices). A blue 'Next' button is at the bottom right, also 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"

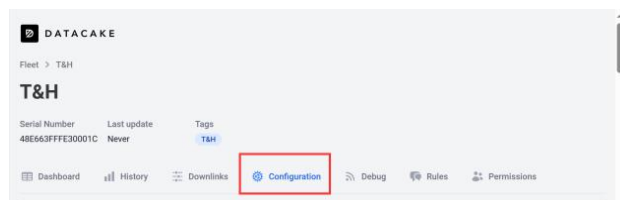
6、Enter DEVEUI、APPEUI、APPKEY、FREQUENCY(take 915 for example) and DEVICE CLASS.

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
light	97.49 lx	41	N/A	N/A
soil	1,584	2.9	N/A	N/A
T&H	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, serial) {
  var input = 0;
  var bytes = payload;
  var num = (input.bytes[0] > 20 ? input.bytes[1] : 0);
  var hum = (input.bytes[2] > 20 ? input.bytes[3] : 0);
  var temp = (input.bytes[4] > 20 ? input.bytes[5] : 0);
  // 16-bit float
  if (input.num) num = 0;
  if (input.hum) hum = 0;
  if (input.temp) temp = 0;
  return {
    "num": num,
    "hum": hum,
    "temp": temp
  };
}
    
```

```
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 humidity = ( input.bytes[3] * 256 + input.bytes[4] ) / 10.0;
  var temperature = input.bytes[5] * 256 + input.bytes[6];
  if (temperature >= 0x8000) {
    temperature -= 0x10000;
  }
  temperature = temperature / 10.0
  var decoded = {
    bat: bat,
    humidity: humidity,
    temperature: temperature,
  };
  // 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: "bat", value: decoded.bat },
    { field: "humidity", value: decoded.humidity },
    { field: "temperature", value: decoded.temperature },
    { 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.

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

num

Identifier

NUM

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 context to global visualisations and reports.

Formula

Optional

Formulas 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

Add Field

Fields define the schema of the data the device stores.

Type

Float

Name

bat

Identifier

BAT

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 context to global visualisations and reports.

Formula

Optional

Formulas 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

Add Field

Fields define the schema of the data the device stores.

Type

Float

Name

hum

Identifier

HUMI

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 context to global visualisations and reports.

Formula

Optional

Formulas 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

Add Field

Fields define the schema of the data the device stores.

Type

Float

Name

temp

Identifier

TEMP

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 context to global visualisations and reports.

Formula

Optional

Formulas 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

Fields

Fields describe the data the device will store.

+ Add Field

Live data

NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
num	NUM	Float	N/A	0	3 minutes ago
bat	BAT	Float	N/A	0	2 minutes ago
humidity	HUMIDITY	Float	N/A	0	2 minutes ago
temperature	TEMPERATURE	Float	N/A	0	in 0 seconds

11、Press RST button, wait until the sensor connects to the gateway successfully, you will see the data the sensor is currently reading.

Fields + Add Field

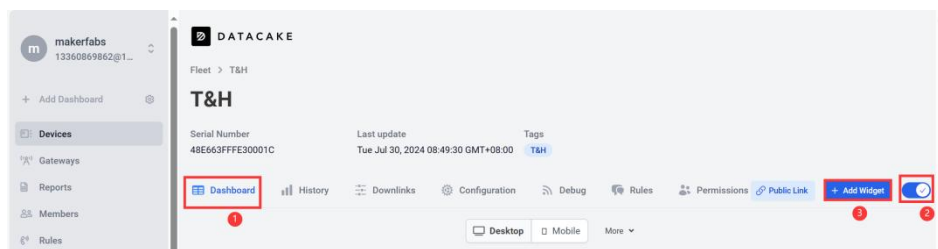
Fields describe the data the device will store.

Live data

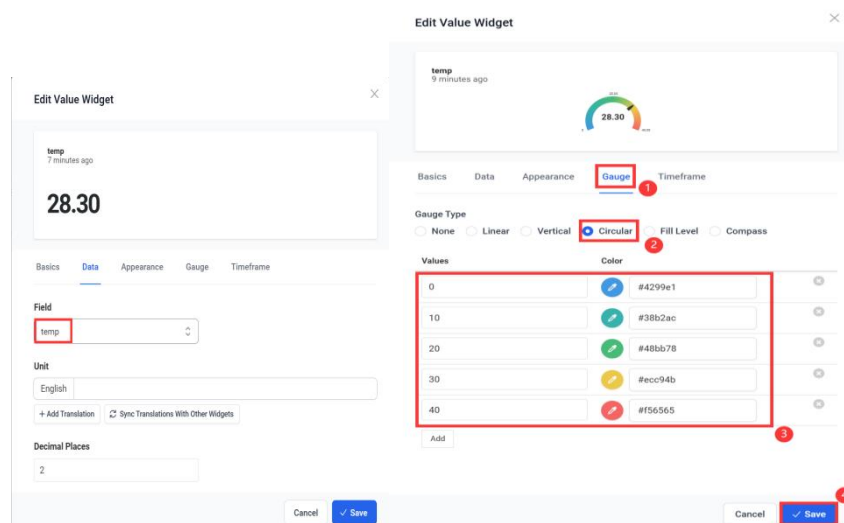
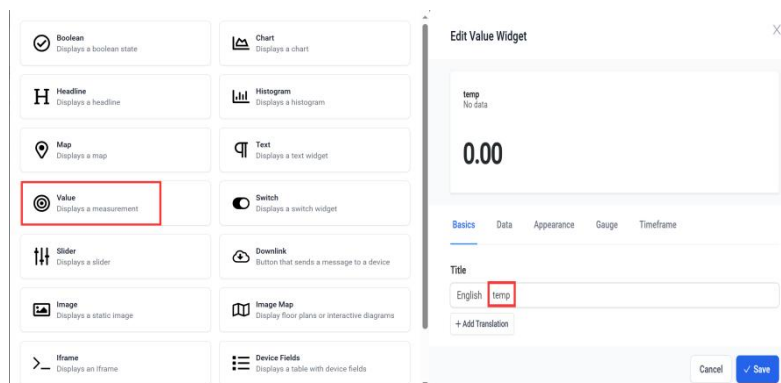
NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE
num	NUM	Float	N/A	20	7 seconds ago
bat	BAT	Float	N/A	2.9	7 seconds ago
humid	HUMI	Float	N/A	62.8	7 seconds ago
temp	TEMP	Float	N/A	28.3	7 seconds ago

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

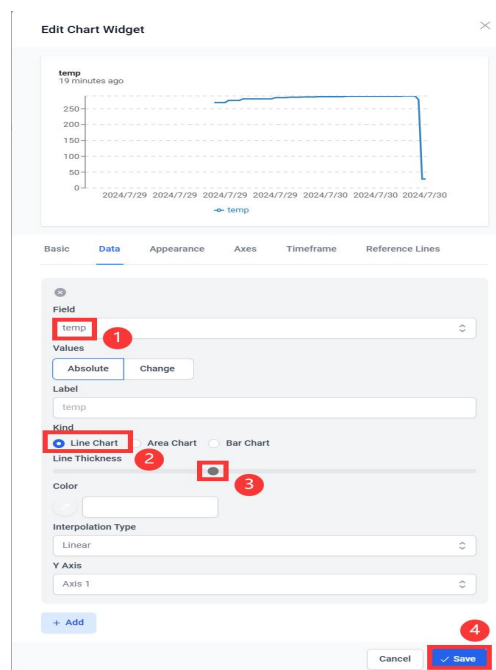
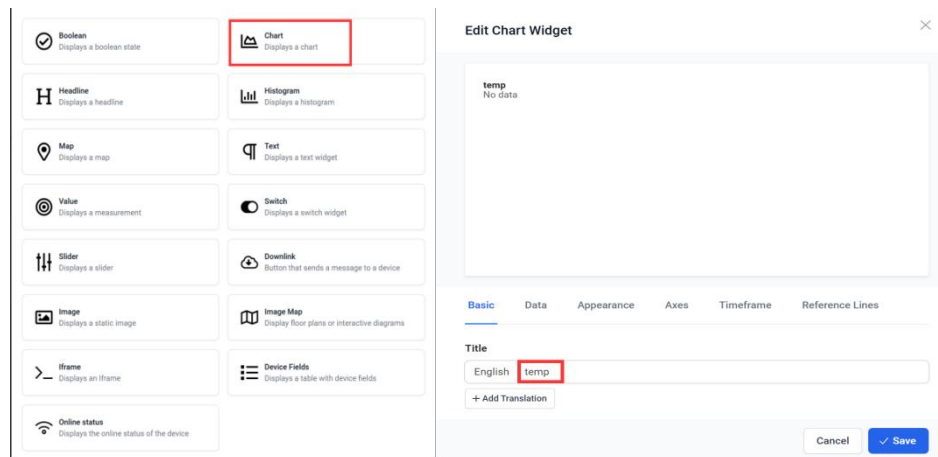
Click “Dashboard-->switch-->+ Add Widget”.



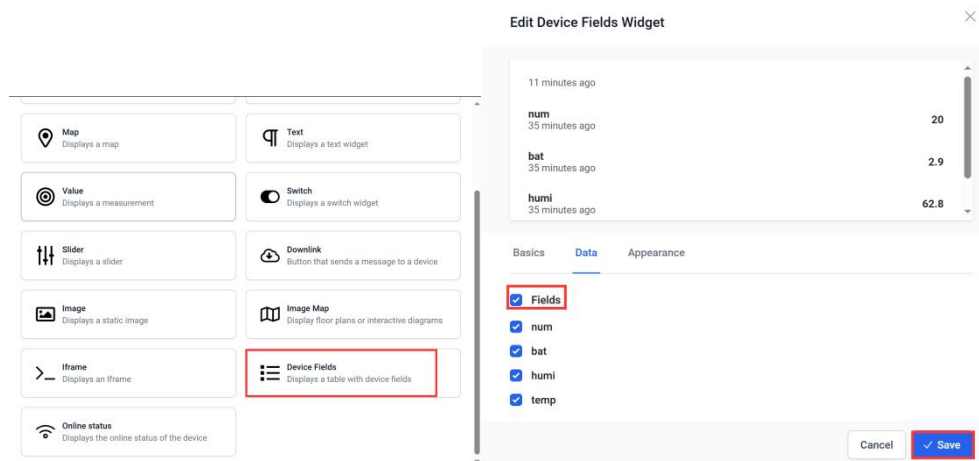
13、Select “Value” and set Title, Field and presentation form as well as the interval color.



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

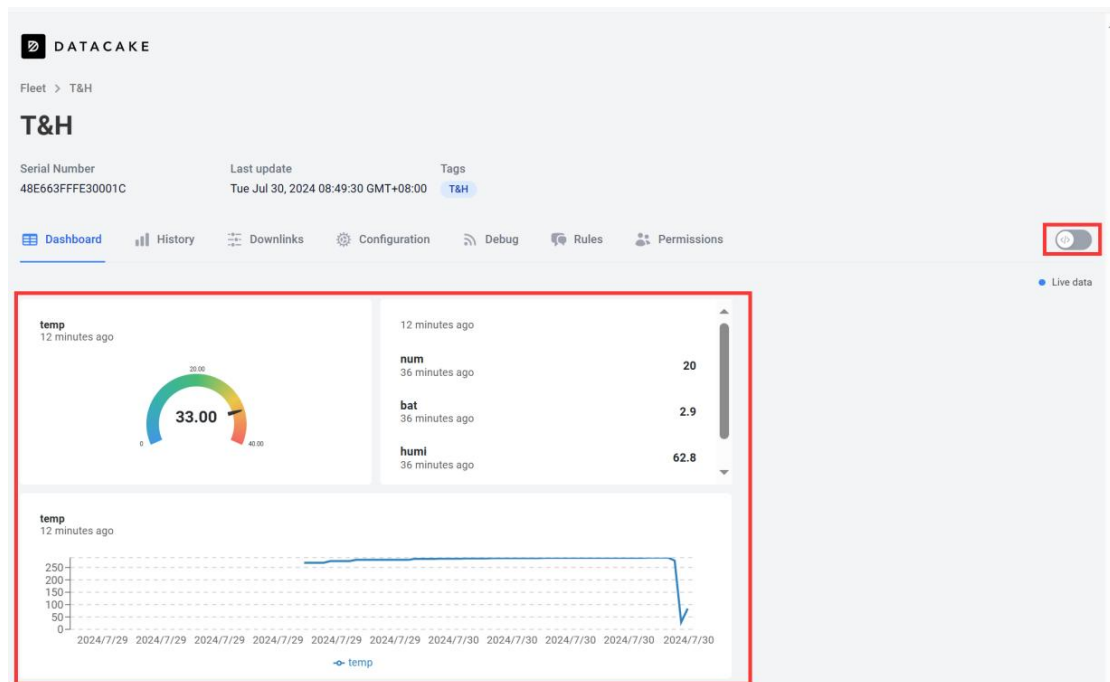


15、Select Device Fields, check “Fields” and click “Save”.



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

17、The steps for humidity are the same as above, and you can add your own.



3.2.1 Change Time Interval (5-1440min)

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

Add Field

Fields define the schema of the data the device stores.

Type

Integer

Name

Sending Time Interval

Identifier

SENDING_TIME_INTERVAL

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 context to global visualisations and reports.

Formula

Optional

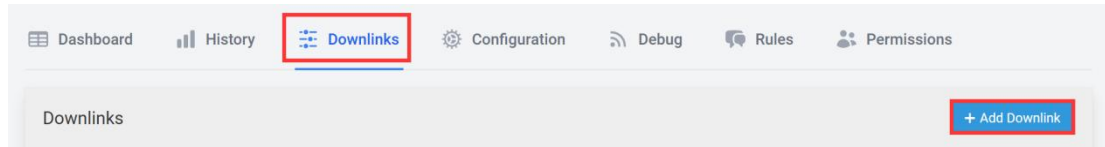
Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

☐ Use Formula

Cancel

Add Field

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

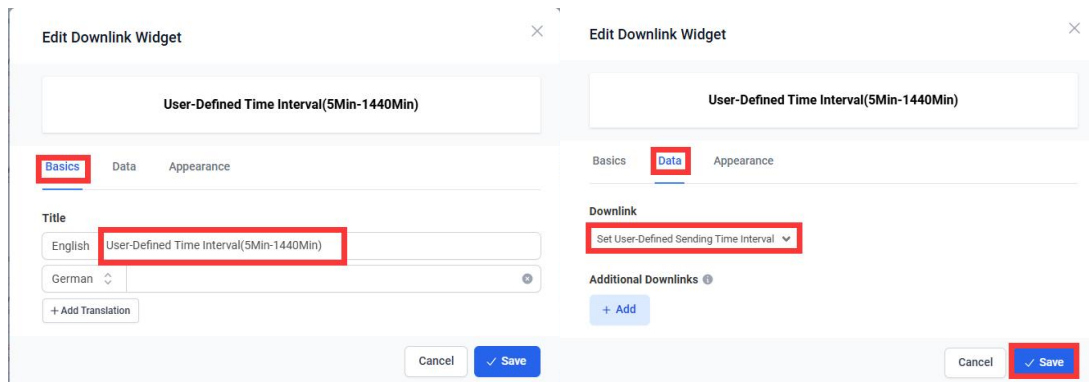
```

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.



4、Click the switch to save, and you can click to change your time Interval.

