



# **AgroSense Valve & Water Flow LoRaWAN® Manual V1.0**



Author: Yuki

Time: 2025.09.12

# Contents

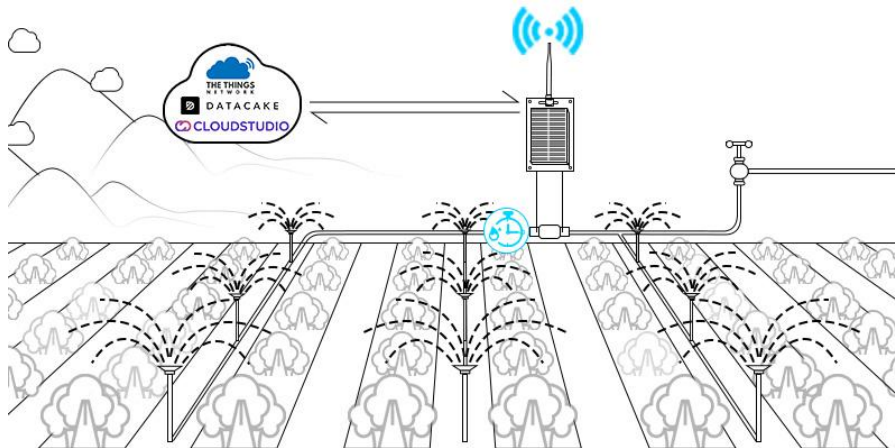
1 Product Description .....	1
1.1 Introduction .....	1
1.2 Feature .....	1
1.3 Parameter .....	2
2 Technical route .....	3
2.1 System Framework .....	3
2.2 Regional frequency band .....	4
3 Usage .....	5
3.1 Interface Specification .....	5
3.2 Usage with TTN &ThingSpeak .....	8
3.3 Usage with Datacake .....	19

# 1 Product Description

## 1.1 Introduction

The AgroSense Valve & Water Flow sensor combines an ultra-low-power pulse solenoid valve with a high-precision YF-S201C Hall Effect Flow Sensor, enabling remote irrigation control and real-time water monitoring over the LoRaWAN® network. The pulse valve only consumes power during switching, retains its ON/OFF state after power loss, and responds within milliseconds, making it ideal for solar + battery-powered systems. The flow sensor provides accurate flow rate and cumulative consumption measurement ( $\pm 3\%$  accuracy), ensuring reliable water usage data even at low flow rates.

This product is cased with IP67 case, solar panel powered, can be used long-term in field application.



## 1.2 Feature

- Includes one valve and one flow meter.
- LoRaWAN version: LoRaWAN Specification 1.0.3. OTAA **Class C**.
- Monitor data and upload **real-time** data regularly.
- Modify the product parameters through **AT commands**.
- Support **downlink** to modify the time interval and control relay switch.
- Integrated data logging capability with a storage capacity of up to 3300 records.
- Compatible with Worldwide **LoRaWAN® Networks**: Support the universal frequency bands EU868/ US915.
- **Long Range**: Up to 2 kilometers in the city, up to 10 kilometers in the wilderness, receive sensitivity -137dBm , transmit power up to 22dBm.

- **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.

## 1.3 Parameter

### 1. General Parameters

Product Model	AGLWVF
Valve Working Fluid	water
Valve Range	0.02~1Mpa
Valve Process Connection	G1/2 (20mm)
Valve Thread Length	14.5 mm
Valve Operating Temperature	1~80℃
Flow Working Fluid	water
Flow Range	0~1.75Mpa
Flow Process Connection	G1/2 (20mm)
Flow Thread Length	12 mm
Flow Operating Temperature	1~80℃

### 2. Wireless Parameters

Communication Protocol	Standard LoRaWAN® protocol V1.0.3
Network Access/Operating Mode	OTAA <b>Class C</b>
MAX Transmit Power	22dBm
Receiver Sensitivity	-137dBm/125kHz SF=12
Frequency Band	EU868/US915

### 3. Physical Parameters

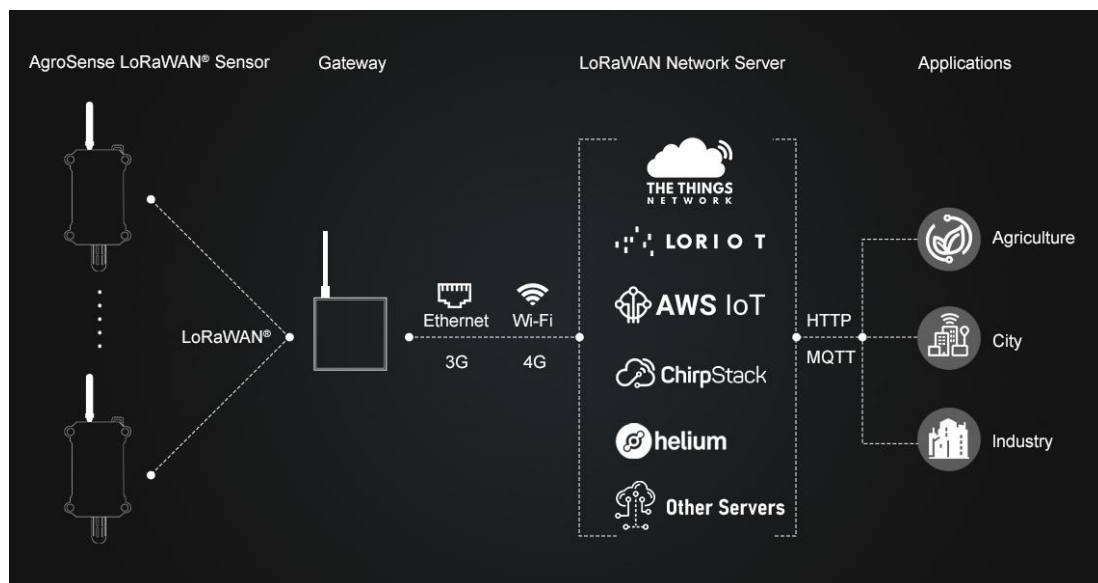
Batteries Power Supply	2 x 18650 3.7V Lion batteries
Solar Power Supply	6V1W
Operating Temperature	-40℃ ~85℃
Protection Class	IP68
Dimensions	115 × 85 × 35 mm
Mounting	Wall Mounting

## 2 Technical route

### 2.1 System Framework

AgroSense Valve & Water Flow 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.



#### Uplink:

##### 1.Data Collection & Transmission

Sensor data and transmits it to the Gateway via LoRaWAN® protocol.

##### 2.Gateway Forwarding

The Gateway packages the raw data and forwards to the Network Server.

##### 3.Data Decoding & Routing

The Network Server decodes the payload and forwards it to the designated Application Server.

##### 4.User Monitoring

The Application Server processes the data and updates the user interface (APP), allowing real-time monitoring of data.

## Downlink:

### 1.Command Generation

A downlink commands generated in the Network Server or Application Server through a predefined API/interface. (Example: Set sampling interval to 10 minutes; Control Valve ON/OFF.)

### 2.Gateway Transmission

The command is encapsulated into a downlink packet and sent to the Gateway via the network.

### 3.End Node Execution

The Gateway transmits the downlink command to the target End Node using the wireless protocol. The End Node parses the command and performs the corresponding action (e.g., activate valve, modify configuration).

## 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

### 3.1 Interface Specification

#### 3.1.1 Plumbing the Flow Sensor and the Latching Solenoid Valve

**Tools & Materials: PTFE thread-seal tape, couplings, wrench**

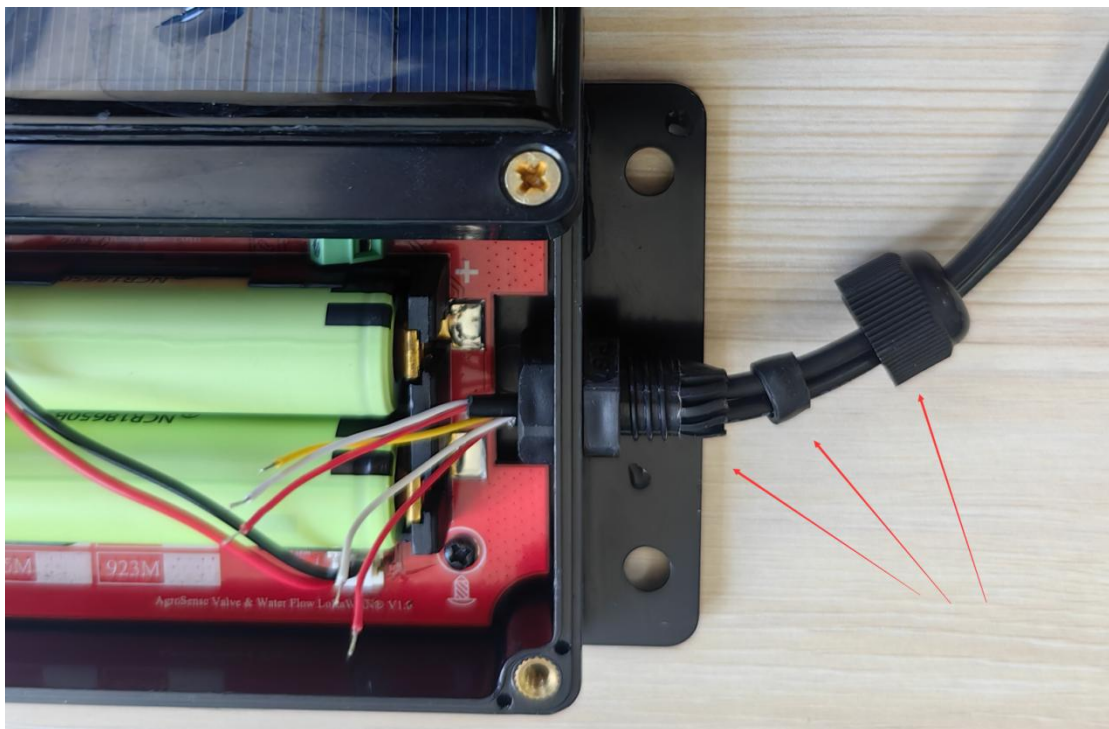
1. **Verify flow direction.** Align the device arrow with the actual flow. Reversed installation will degrade measurement accuracy and may affect valve operation.
2. Apply PTFE tape. Wrap the threads clockwise (in the tightening direction) with 6–10 turns of PTFE tape, keeping the bore unobstructed.
3. Make up the joints. Tighten by hand first, then use a wrench for additional  $\frac{1}{4}$ – $\frac{1}{2}$  turn. Do not overtighten to avoid damaging threads or seals.



### 3.1.2 Wiring Two Sensors (Cable Routing & Termination)

**Goal:** Route both harnesses through the cable gland and connect them to the internal terminals

1. **Open the enclosure.** Remove the **four screws** in a criss-cross pattern and lift the cover.
2. **Prepare the cable gland.** Loosen the gland cap and seal.
3. **Feed the cables.** Route the **flow sensor** and **latching solenoid valve** harnesses **through the cable gland** into the enclosure. Avoid nicking the jacket; keep a proper bend radius.



#### 4. Flow sensor (3-wire) connections:

- WATER FLOW (signal) — **Yellow**
- GND — **White**
- VCC — **Red**

#### 5. Latching solenoid valve (2-wire) connections:

- OUT1 — **Red**
- OUT2 — **White**

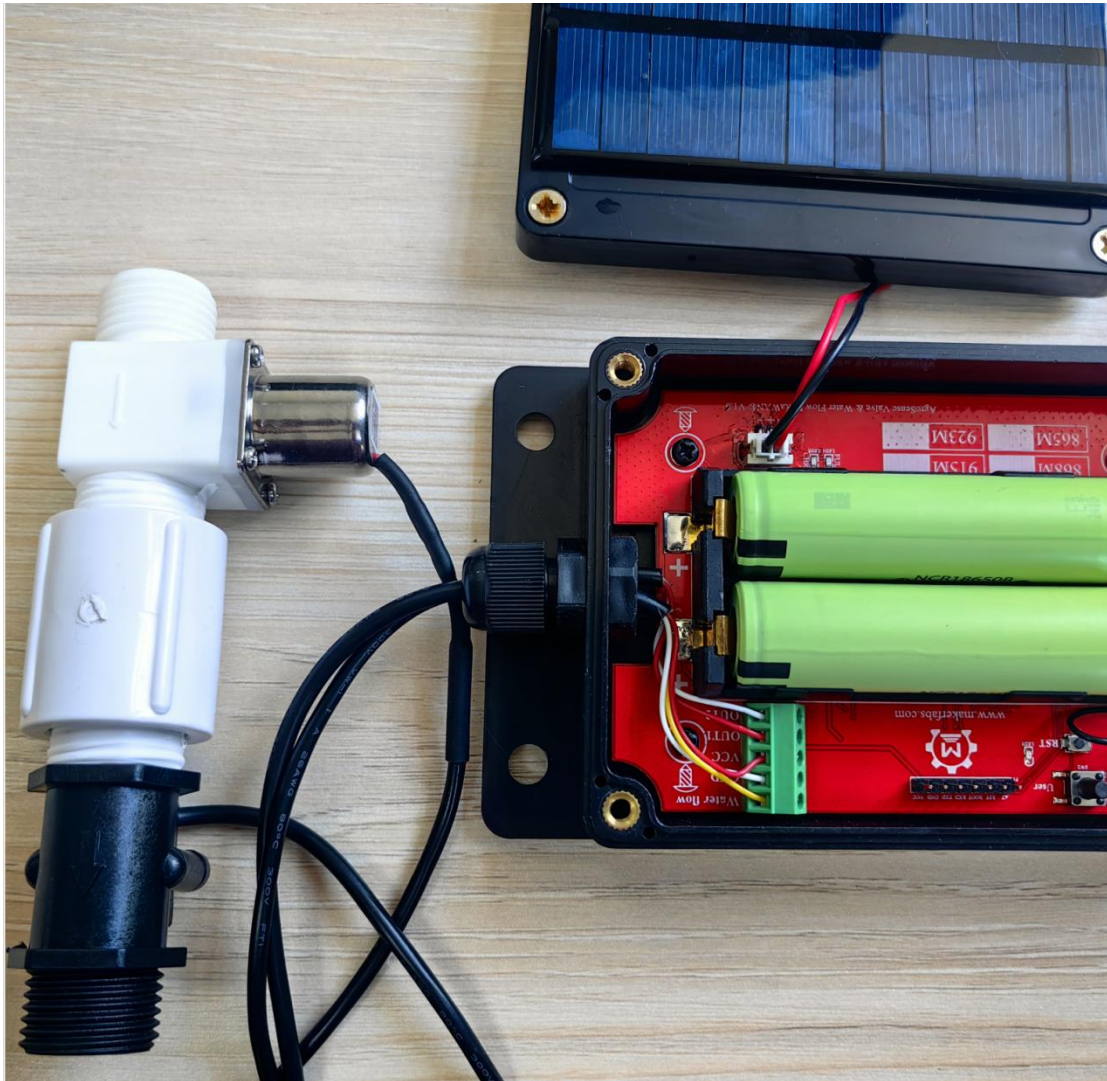
The **polarity** defines **OPEN/CLOSE**. Connect only to OUT1/OUT2 on the controller; **do not** wire directly to VCC/GND.

6. **Strain relief & sealing.** Secure the harness with a tie inside the enclosure and **tighten the**



**gland** until the cable cannot slip and the seal is firm.

7. **Verification.** Cross-check each terminal; ensure no mis-wiring, no exposed strands, and all screws are tight.



### 3.1.3 Platform Registration, Solar Connection & Mounting

**Goal:** Register OTAA credentials and verify communication, connect the solar panel and close the enclosure, then secure the unit in place.

1. **Register on the platform and verify communication**
  - In your LNS/platform, register **DevEUI / JoinEUI (AppEUI) / AppKey (OTAA)**.
  - Power on the device and confirm a successful **join** (Join Accept) and the **first uplink** on the platform.
2. **Connect the solar panel and close the enclosure**
  - Plug the **solar panel connector** into the device **PV input**, observing **polarity (+ / -)**.

- Tighten the **four cover screws** in a **criss-cross pattern**; check the gasket is seated to ensure **waterproofing**.
- 3. **Mount the device**
- Use **screws or cable ties** to secure the unit at the intended location; form a **drip loop** in the cable to prevent water ingress.
- Aim the **solar panel** toward good sunlight, avoiding shading; keep the **antenna** away from large metal surfaces ( $\geq 10$  cm).
- Re-verify wiring and fasteners—no loose connections or exposed strands.

### 3.2 Usage with TTN & ThingSpeak

In the phase, We use The Things Network(TTN) as data server, and Thingspeak as console to display data& control the valve.

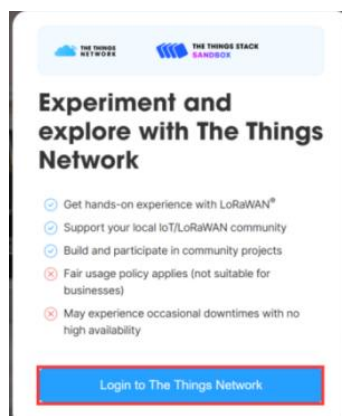
we need to configuration the country/area frequency, inputting DEV EUI/ APP EUI/ APP Key, decodes, and connect to ThingSpeak.

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

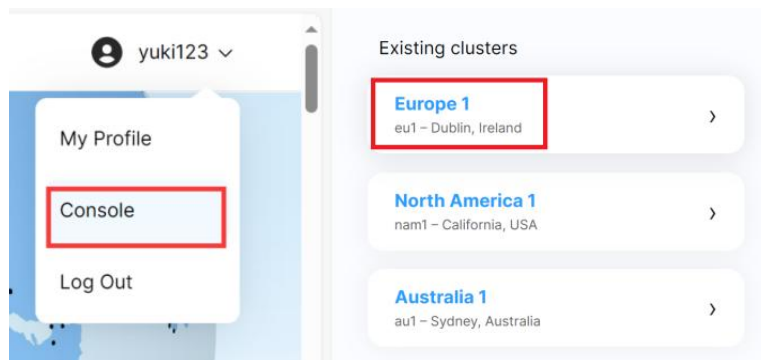
- End Nodes and Gateway: AgroSense Valve & Water Flow.(The AgroSense series is applicable)
- Network Server: The Things Network. ( Datacake, Lorient, AWS IoT, ChirpStack, ect)
- Application Server: ThingSpeak.(Datacake, Blockbax, akenza, ect)

#### 3.2.1 Network Server configuration

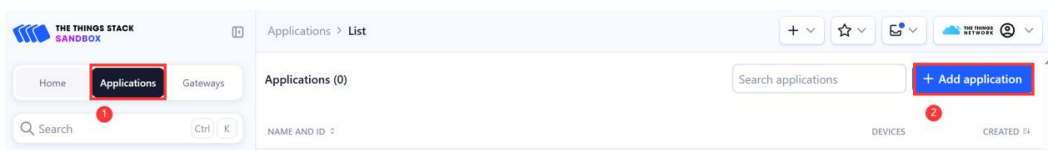
- Open The Things Network in your browser and login it. (Or register an account)



- 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

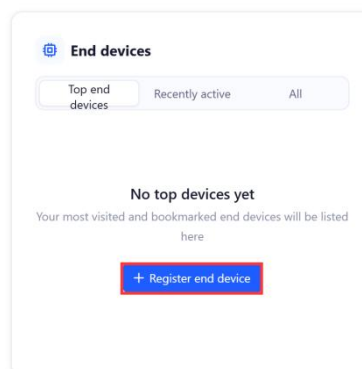
Description

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, **select class C** 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 ⓘ \*

United States 902-928 MHz, FSB 2 (used by TTN) 2

LoRaWAN version ⓘ \*

LoRaWAN Specification 1.0.3 3

Regional Parameters version ⓘ \*

RP001 Regional Parameters 1.0.3 revision A

Show advanced activation, LoRaWAN class and cluster settings ^ 4

Activation mode ⓘ

☒ Over the air activation (OTAA)

☐ Activation by personalization (ABP)

☐ Define multicast group (ABP & Multicast)

Additional LoRaWAN class capabilities ⓘ

Class C (Continuous) 5

Network defaults ⓘ

☒ Use network's default MAC settings

Cluster settings ⓘ

☐ Skip registration on Join Server

### 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.

Device overview
Live data
Messaging
Location
Payload formatters
Settings

TIME TYPE DATA PREVIEW Verbose stream Export as JSON Pause Clear

10:19:32 Schedule data download for... DevAddr: 26 0B EA 1A Rx1 Delay: 5

10:19:31 Update end device [ "activated\_at" ]

10:19:31 Forward uplink data message DevAddr: 26 0B EA 1A 00 01 06 34 1D 00 36 EE 00 00 FPort: 2 Data rate: SF12BW125 SNR: 7

10:19:31 Successfully processed dat... DevAddr: 26 0B EA 1A

10:19:23 Forward join-accept message DevAddr: 26 0B EA 1A JoinEUI: 48 FF 00 00 00 00 02 00 DevEUI: 48 E6 63 FF FE 30 02 00

10:19:21 Successfully processed joi... JoinEUI: 48 FF 00 00 00 00 02 00 DevEUI: 48 E6 63 FF FE 30 02 00

10:19:21 Accept join-request DevAddr: 26 0B EA 1A JoinEUI: 48 FF 00 00 00 00 02 00 DevEUI: 48 E6 63 FF FE 30 02 00

10:19:07 Create end device

### 3.2.2 Decoder

- Now, we need to decode the data.

Overview
Live data
Messaging
Location
Payload formatters
General settings

TIME Type Data preview Verbose stream Export as JSON Pause Clear

09:52:56 Schedule data download for tr... DevAddr: 26 0C DA 7C Rx1 Delay: 5

09:52:56 Fail to send webhook Request

09:52:55 Forward uplink data message DevAddr: 26 0C DA 7C 00 01 06 31 1C 00 00 00 00 ... FPort: 2 Data rate: SF10BW125 SNR: 12.5 RSSI: -80

09:52:55 Successfully processed data m... DevAddr: 26 0C DA 7C

09:52:49 Forward join-accept message DevAddr: 26 0C DA 7C JoinEUI: 48 FF 00 00 00 00 00 16 DevEUI: 48 E6 63 FF FE 30 00 16

09:52:48 Successfully processed join-r... DevAddr: 26 0C 28 1F JoinEUI: 48 FF 00 00 00 00 00 16 DevEUI: 48 E6 63 FF FE 30 00 16

09:52:48 Accept join-request DevAddr: 26 0C DA 7C JoinEUI: 48 FF 00 00 00 00 00 16 DevEUI: 48 E6 63 FF FE 30 00 16

Data length	Data description	Value range	Explanation
-------------	------------------	-------------	-------------

**AgroSense Valve & Water Flow LoRaWAN®**

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 amplified by a factor of 10. To get the actual value, divide it by 10. For example, if the value is 0x21 (33), the actual voltage is 3.3 V
byte 3	Valve state	0/1	0 is OFF, 1 is ON.
byte 4	Flow One-second pulse bits 8 to 15		Number of pulses per second of change in water flow velocity For example, if the value is 0x0021 = 33, then the value is 33.
byte 5	Flow One-second pulse bits 0 to 7		
byte 6	Valve open-close duration bits 24 to 31		Timing begins when the solenoid valve is opened, and once it closes, the recorded opening duration is uploaded. For example, if the value is 0x000000A8 = 168, then the valve open duration: 168 seconds.
byte 7	Valve open-close duration bits 16 to 23		
byte 8	Valve open-close duration bits 8 to 15		
byte 9	Valve open-close duration bits 0 to 7		
byte 10	data transmission interval bits 24 to 31		The time interval for data transmission has been increased by a factor of 1000. The unit is seconds.
byte 11	data transmission interval bits 16 to 23		
byte 12	data transmission interval bits 8 to 15		
byte 13	data transmission interval bits 0 to 7		
Downlink			
Fport 1	Change the data sending interval		10S-1440min
Fport 5	Upload the quantity of the latest local logged data		
Fport 6	Change the Valve	0/1	0 is OFF, 1 is ON.

	ON/OFF		
--	--------	--	--

Example: 0x00 0x01 0x24 0x01 0x00 0x21 0x00 0x00 0x00 0x00 0x00 0x09 0x27 0xC0

Data parsing:

Battery voltage is 3.6V.

Valve status is ON.

Flow One-second pulse change is 33.

Valve open-close duration is 0. (The timing will be uploaded when the valve closes)

Data transmission interval value is 600s.

- Know how to decode it after, we need to write it in code. (You can check it out on [Github](#))

```
function decodeUplink(input) {
  var bytes = input.bytes;
  var num = bytes[0] * 256 + bytes[1];
  var bat = input.bytes[2]/10;
  var Valve = input.bytes[3]; //Valve state    :0--OFF; 1--ON
  var flow_pulse_1s_diff = input.bytes[4] * 256 + input.bytes[5]; //Pulse changes within 1 second
  var flow_velocity = flow_pulse_1s_diff*60/450; // L/min
  var flow_rate = flow_pulse_1s_diff/450; //L
  var Valve_on_all_time = input.bytes[6]* 16777216 + input.bytes[7]* 65536 + input.bytes[8] * 256 +
input.bytes[9]; //Time from opening to closing of the latest valve
  var interval = (input.bytes[10]* 16777216 + input.bytes[11]* 65536 + input.bytes[12] * 256 + input.bytes[13]) /
1000; //interval when valve is open
  return {
    data: {
      field1: num,
      field2: bat,
      field3: Valve,
      field4: flow_velocity,
      field5: flow_rate,
      field6: Valve_on_all_time,
      field7: interval,
    },
    warnings: [],
    errors: []
  };
}
```

- Select “Payload formatters” and follow the steps.

### 3.2.3 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)

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



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

Channel ID: 2599652  
 Author: mwa0000034232775  
 Access: Private

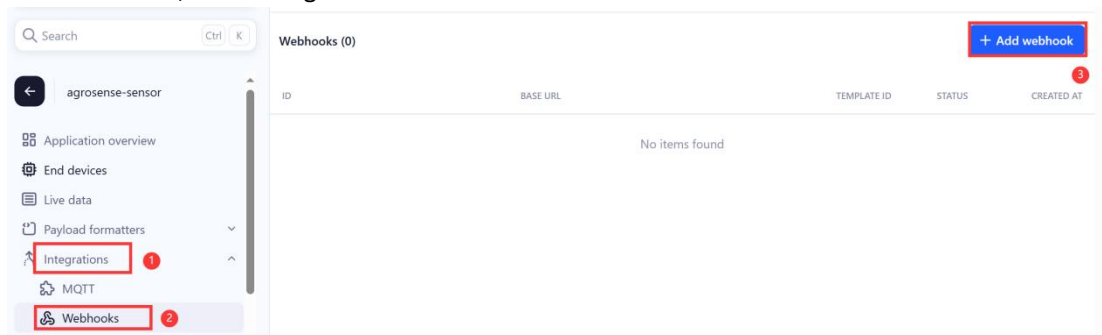
Private View Public View Channel Settings Sharing API Keys

Write API Key

Key N9IBFTBI3J36T779

### 3.2.4 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.)





### 3.2.5 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 (Fport5)

Users can view previous data based on this feature.

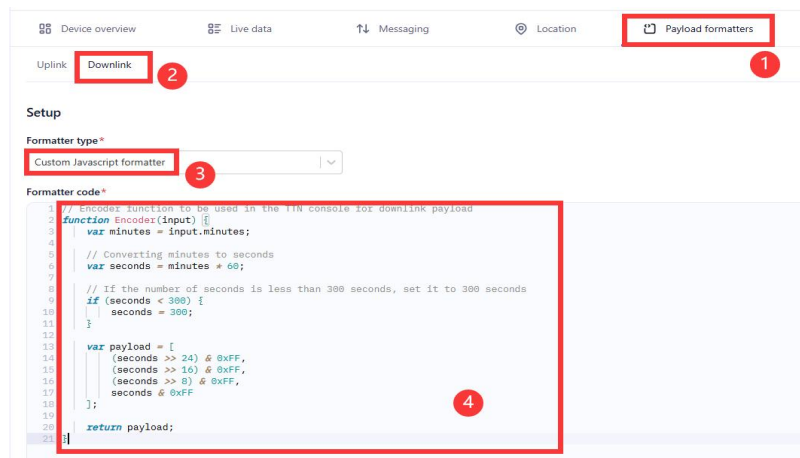
- Change the Valve ON/OFF (Fport6)

Modify the valve status, 0 is OFF, 1 is ON.

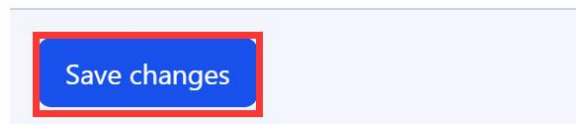
### Modify the time interval :

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.

### Change the Valve ON/OFF:

1、Click “Payload formatters-->Downlink” and follow the steps.

Formatter code you can find in [Github](#).

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

Uplink **Downlink**

Setup

Formatter type\*

Custom Javascript formatter

Formatter code\*

```
1 function encodeDownlink(input) {
2   // Get the relay status from user input; the relay is identified by the fPort
3   var relayStatus = input.RELAY; // User input: 0 or 1
4   var fPort = input.fPort; // Get fPort
5
6   // Ensure that relayStatus only accepts 0 or 1
7   relayStatus = relayStatus === 1 ? 1 : 0;
8
9   // Determine the relay state based on fPort
10  var payload = [relayStatus];
11
12  // Return the downlink message in the format required by TTN
13  return {
14    bytes: payload, // Byte array to send
15    fPort: fPort, // Dynamically set fPort
16    warnings: [],
17    errors: []
18  };
19 }
```

2、Click “Save changes”.

**Save changes**

2、Click “Messaging-->Schedule downlink”. The valve will ON/OFF immediately after the modification. ( 00 is OFF, 01 is ON )

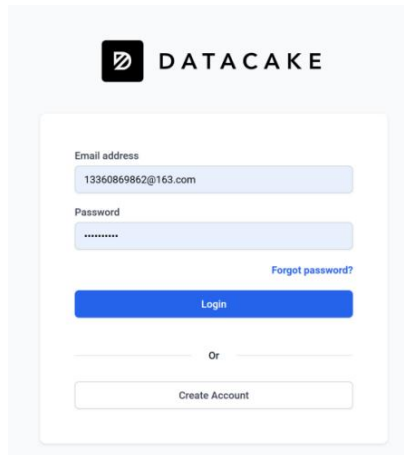
The screenshot displays the 'Messaging' tab in a web interface. At the top, there are three tabs: 'Device overview', 'Live data', and 'Messaging'. The 'Messaging' tab is active and highlighted with a red box. Below the tabs, there are two buttons: 'Schedule downlink' (highlighted with a red box) and 'Simulate uplink'. The 'Schedule downlink' section contains the following fields and options:

- Schedule downlink**
- Insert Mode**
  - ☒ Replace downlink queue
  - ☐ Push to downlink queue (append)
- FPort \***
  - Input field containing the value '6' (highlighted with a red box).
- Payload type**
  - ☒ Bytes
  - ☐ JSON
- Payload**
  - Input field containing the value '01' (highlighted with a red box).
- The desired payload bytes of the downlink message
- ☐ Confirmed downlink
- Schedule downlink** (button at the bottom, highlighted with a red box)

### 3.3 Usage with Datacake

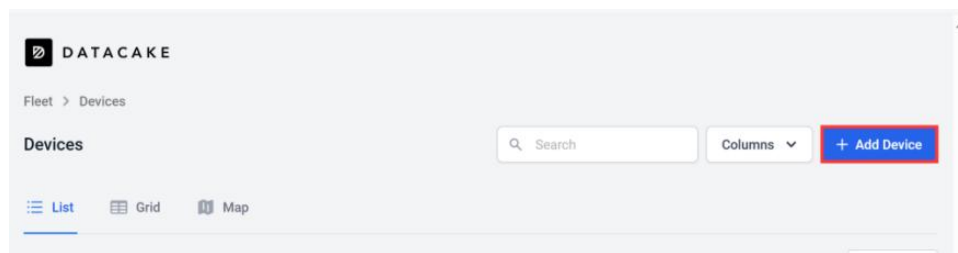
In this phase, we use DataCake(<https://datacake.co/>) as the data server & console.

#### 1、Login datacake or Create Account



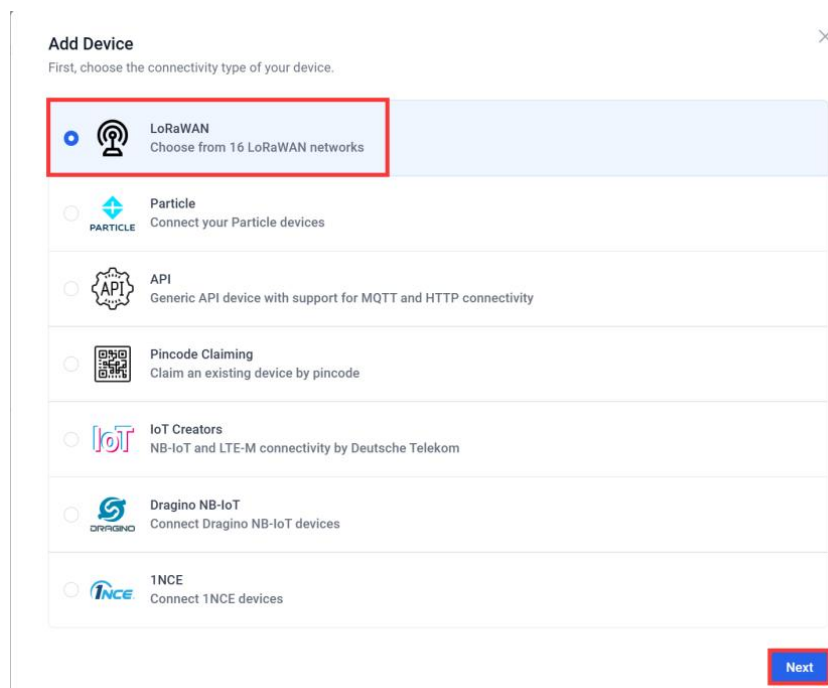
The image shows the DataCake login and registration interface. At the top is the DataCake logo. Below it is a form with two input fields: 'Email address' (containing '13360869862@163.com') and 'Password' (masked with dots). A 'Forgot password?' link is next to the password field. Below the fields is a blue 'Login' button. Underneath is an 'Or' separator, followed by a 'Create Account' button.

#### 2、Click “Add Device”



The image shows the DataCake dashboard. The top bar includes the DataCake logo, a breadcrumb 'Fleet > Devices', a search bar, a 'Columns' dropdown, and a red-bordered '+ Add Device' button. Below the top bar, there are tabs for 'List', 'Grid', and 'Map'. The 'List' tab is selected.

#### 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 connectivity options, each with a radio button and an icon:
 

- LoRaWAN** (selected, highlighted with a red box): Choose from 16 LoRaWAN networks
- 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
- 1NCE: Connect 1NCE devices

 At the bottom right of the dialog is a red-bordered '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

STEP 2

STEP 3

STEP 4

Product

Network Server

Devices

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

STEP 2

STEP 3

STEP 4

Product

Network Server

Devices

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

20

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

The 'Add LoRaWAN Device' form consists of four steps: STEP 1 Product, STEP 2 Network Server, STEP 3 Devices, and STEP 4 Plan. In STEP 3, users are prompted to provide one or multiple LoRaWAN device EUIs along with the corresponding names they should have on Datacake. Alternatively, they can upload a CSV file. The form includes fields for DEVEUI, NAME, APPEUI, APPKEY 1.0 / NWKEY 1.1, FREQUENCY, and DEVICE CLASS. Red boxes highlight the input fields in both screenshots.

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

The 'Add LoRaWAN Device' form shows four pricing options: Free (€0.00 / month), Light (€1.00 / month), Standard (€3.00 / month), and Plus (€5.00 / month). Each option lists its data retention and datapoint limits. The 'Free' option is highlighted with a red box. Below the pricing options, there is a blue banner for consolidated billing and a section for applying a code. The 'Add 1 device' button is highlighted with a red box.

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

**DATA CAKE**

Fleet > Devices

Devices

Search Columns + Add Device

List Grid Map

Actions

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	👁️ ⋮

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 an 'agrosense sensor'. The 'Configuration' tab is selected. Below the configuration options, the 'Payload Decoder' section is visible. A red box highlights the code editor area, and a red circle with the number '1' is next to the code. At the bottom right, a red circle with the number '2' is next to the 'Save' button.

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

    var num = input.bytes[0] * 256 + input.bytes[1]
    var bat = input.bytes[2]/10
    var valve = input.bytes[3] //Valve state    :0--OFF; 1--ON
    var flow_pulse_1s_diff = input.bytes[4] * 256 + input.bytes[5] //Pulse changes within 1 second
    var flow_velocity = flow_pulse_1s_diff*60/450;    // L/min
    var flow_rate = flow_pulse_1s_diff/450; //L
    var Valve_on_all_time =    input.bytes[6]* 16777216 + input.bytes[7]* 65536 + input.bytes[8] * 256 +
input.bytes[9] //Time from opening to closing of the latest valve
    var interval = (input.bytes[10]* 16777216 + input.bytes[11]* 65536 + input.bytes[12] * 256 + input.bytes[13])
/ 1000 //interval when valve is open
    var decoded =
    {
        NUM:num
        BAT:bat
        VALVE:valve
        //FLOW_PULSE_1S_DIFF:flow_pulse_1s_diff
        FLOW_VELOCITY:flow_velocity
        FLOW_RATE:flow_rate
    }
}
```



```

        VALVE_ON_ALL_TIME:Valve_on_all_time
        INTERVAL:interval
    };
    // Test for LoRa properties in normalizedPayload
    try {
        if (normalizedPayload.gateways && normalizedPayload.gateways.length > 0) {
            decoded.LORA_RSSI = normalizedPayload.gateways[0].rssi || 0;
            decoded.LORA_SNR = normalizedPayload.gateways[0].snr || 0;
        } else {
            decoded.LORA_RSSI = 0;
            decoded.LORA_SNR = 0;
        }
        decoded.LORA_DATARATE = normalizedPayload.spreading_factor
            || normalizedPayload.data_rate
            || (normalizedPayload.networks && normalizedPayload.networks.lora &&
normalizedPayload.networks.lora.dr)
            || "unknown";
    } catch (error) {
        console.log('LoRa property parsing error:', error);
        decoded.LORA_RSSI = 0;
        decoded.LORA_SNR = 0;
        decoded.LORA_DATARATE = "unknown";
    }
    return decoded;
}

```

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: Boolean

Name: valve

Identifier: VALVE

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

Assign roles to highlight key measurement fields across the platform; each role is limited to one field per product.

Semantic: None

Assign semantic fields to standardize data across devices.

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.

Cancel Add Field

NAME	IDENTIFIER	TYPE
Interval	INTERVAL	Integer
Lora Datarate	LORA_DATARATE	String
Lora Rssi	LORA_RSSI	Integer
Lora Snr	LORA_SNR	Integer
bat	BAT	Float
num	NUM	Integer
Valve_on_all_time	VALVE_ON_ALL_TIME	Integer
valve	VALVE	Boolean
flow_rate	FLOW_RATE	Float
Flow Velocity	FLOW_VELOCITY	Float

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

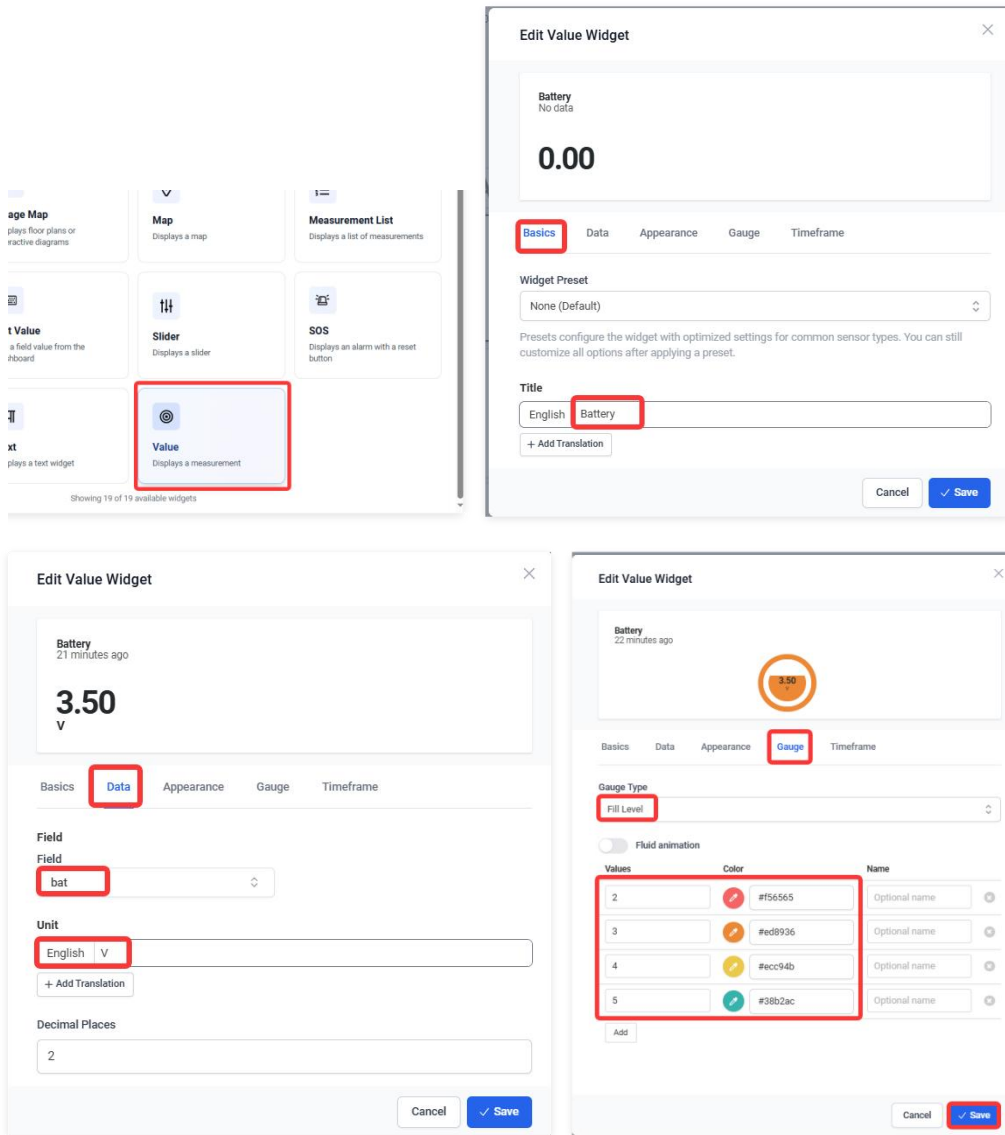
NAME	IDENTIFIER	TYPE	ROLE / SEMANTIC	CURRENT VALUE	LAST UPDATE
Interval	INTERVAL	Integer		3,600 s	23 seconds ago
Lora Datarate	LORA_DATARATE	String		SF10BW125.0	23 seconds ago
Lora Rssi	LORA_RSSI	Integer	Signal	-85 dBm	23 seconds ago
Lora Snr	LORA_SNR	Integer	Signal-to-Noise Ratio	6 dB	23 seconds ago
bat	BAT	Float		3.5	23 seconds ago
num	NUM	Integer		41	23 seconds ago
Valve_on_all_time	VALVE_ON_ALL_TIME	Integer		0	23 seconds ago
valve	VALVE	Boolean		False	23 seconds ago
flow_rate	FLOW_RATE	Float		0	23 seconds ago
Flow Velocity	FLOW_VELOCITY	Float		0	23 seconds ago

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

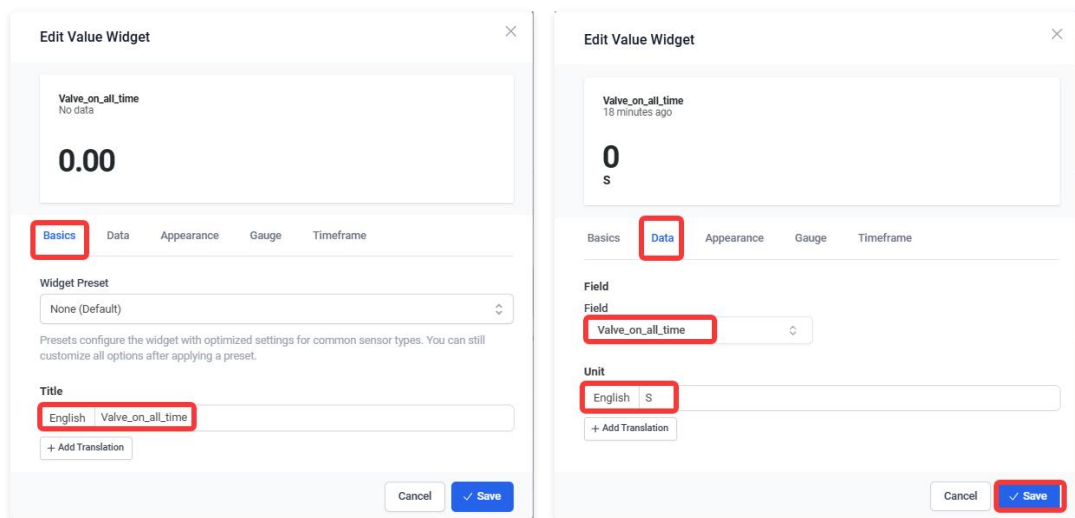
Click “Dashboard-->switch-->+ Add Widget”.



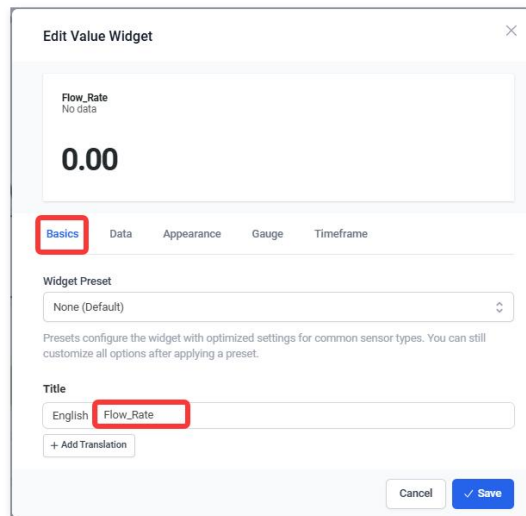
13、 Add “Battery” field, Select “Value” and set Title.



14、Add “Valve\_on\_all\_time” field, Select “Value” and set Title.



15、Add “Flow\_Rate” field, Select “Value” and set Title.



**Edit Value Widget**

Flow\_Rate  
No data

0.00

Basics Data Appearance Gauge Timeframe

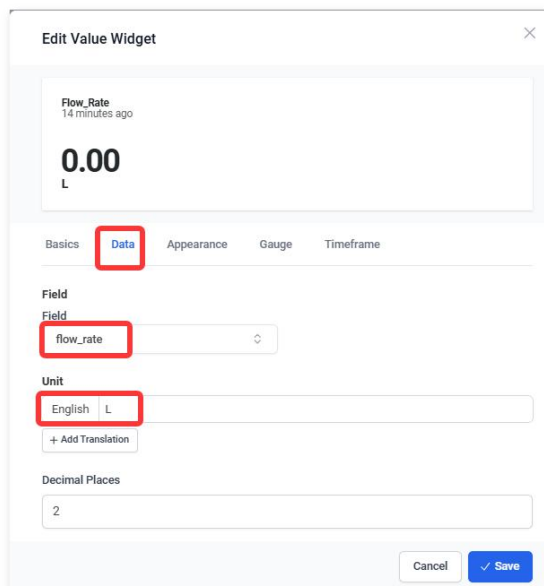
Widget Preset  
None (Default)

Presets configure the widget with optimized settings for common sensor types. You can still customize all options after applying a preset.

Title  
English Flow\_Rate

+ Add Translation

Cancel Save



**Edit Value Widget**

Flow\_Rate  
14 minutes ago

0.00  
L

Basics Data Appearance Gauge Timeframe

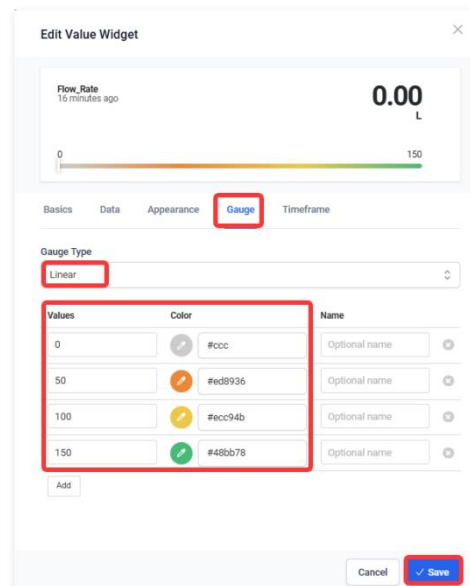
Field  
flow\_rate

Unit  
English L

+ Add Translation

Decimal Places  
2

Cancel Save



**Edit Value Widget**

Flow\_Rate  
15 minutes ago

0.00  
L

0 150

Basics Data Appearance Gauge Timeframe

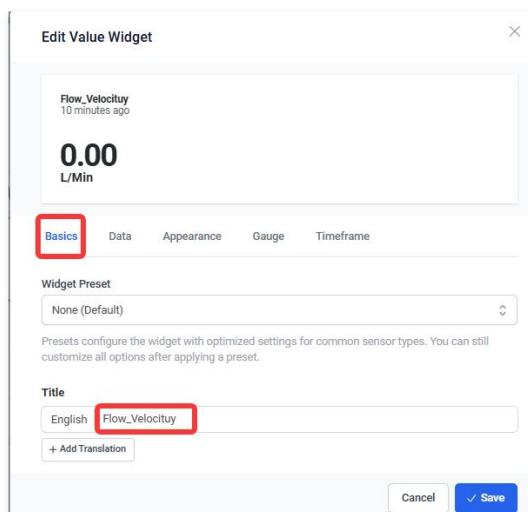
Gauge Type  
Linear

Values	Color	Name
0	#ccc	Optional name
50	#ed936	Optional name
100	#ecc94b	Optional name
150	#48bb78	Optional name

Add

Cancel Save

16、Add “Flow\_Velocity” field, Select “Value” and set Title.



**Edit Value Widget**

Flow\_Velocity  
10 minutes ago

0.00  
L/Min

Basics Data Appearance Gauge Timeframe

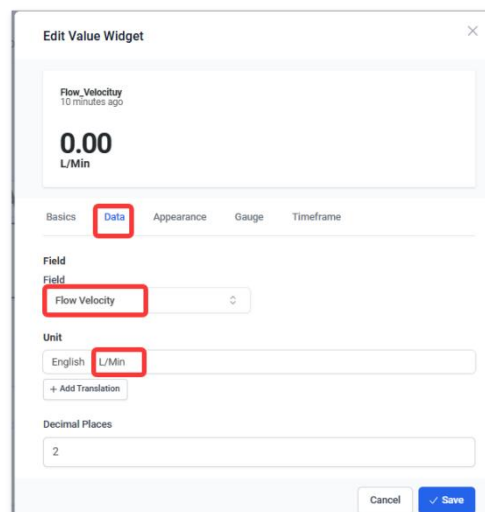
Widget Preset  
None (Default)

Presets configure the widget with optimized settings for common sensor types. You can still customize all options after applying a preset.

Title  
English Flow\_Velocity

+ Add Translation

Cancel Save



**Edit Value Widget**

Flow\_Velocity  
10 minutes ago

0.00  
L/Min

Basics Data Appearance Gauge Timeframe

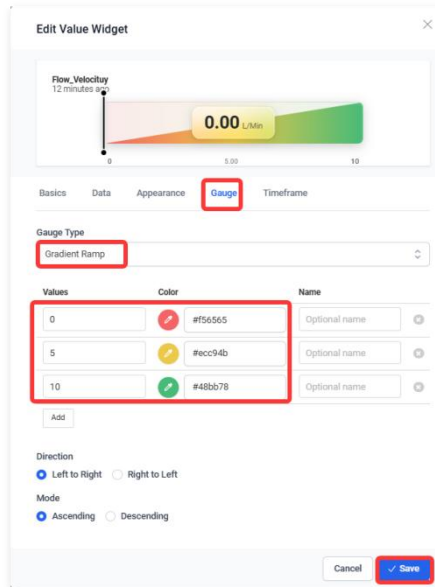
Field  
Flow Velocity

Unit  
English L/Min

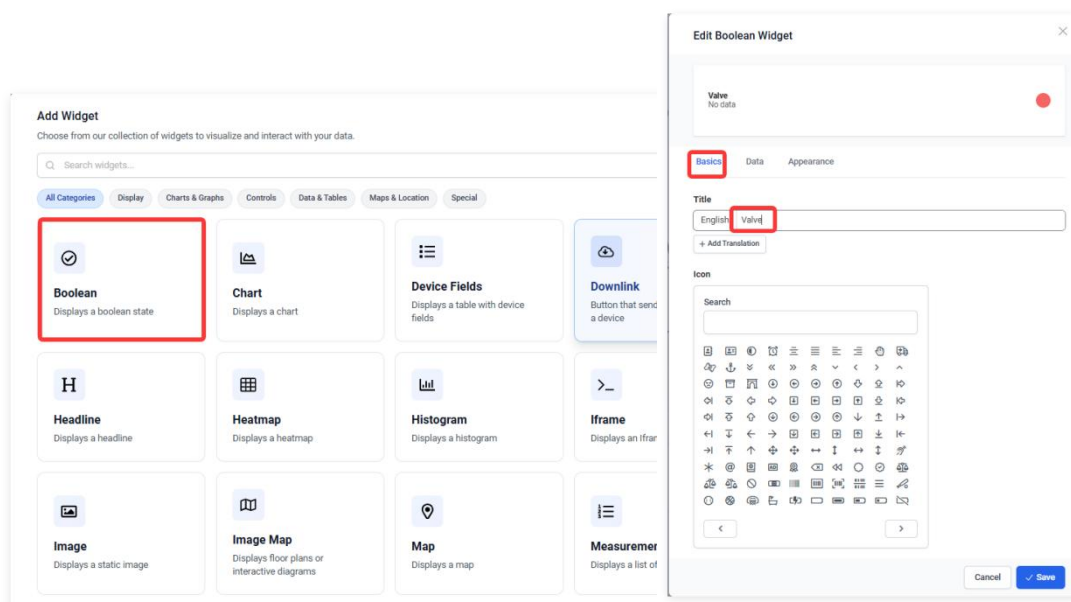
+ Add Translation

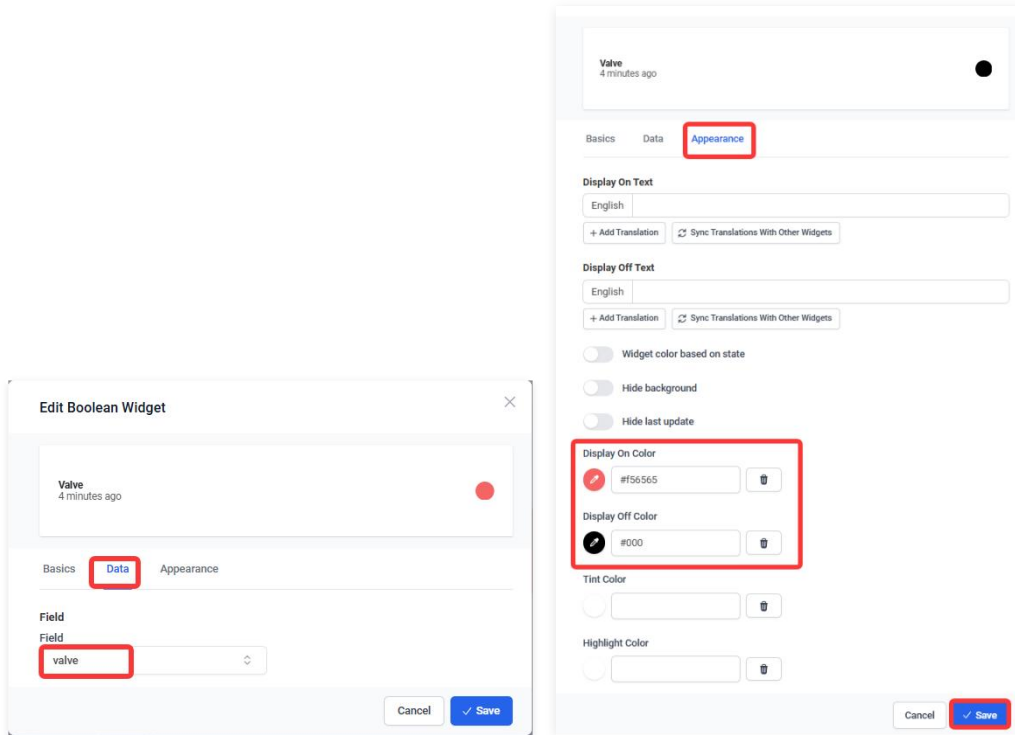
Decimal Places  
2

Cancel Save



17、Add “Valve” field, Select “Boolean” and set Title, Field as well as the status color.





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



### 3.3.1 Downlink

The downlink has the following 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 (Fport5)

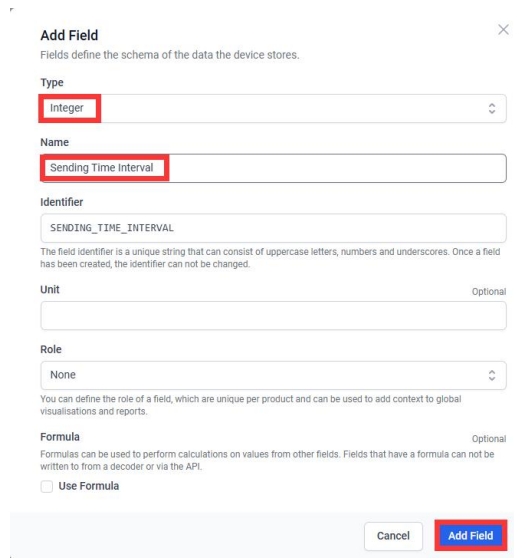
Users can view previous data based on this feature.

- Change the Valve ON/OFF (Fport6)

Modify the valve status, 0 is OFF, 1 is ON.

**Modify the time interval :**

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.

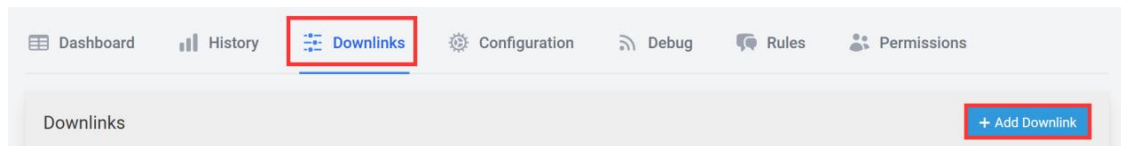
Formula: Optional

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.

**Edit Downlink Widget**

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

**Basics** | Data | Appearance

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

**Sending Time Interval**

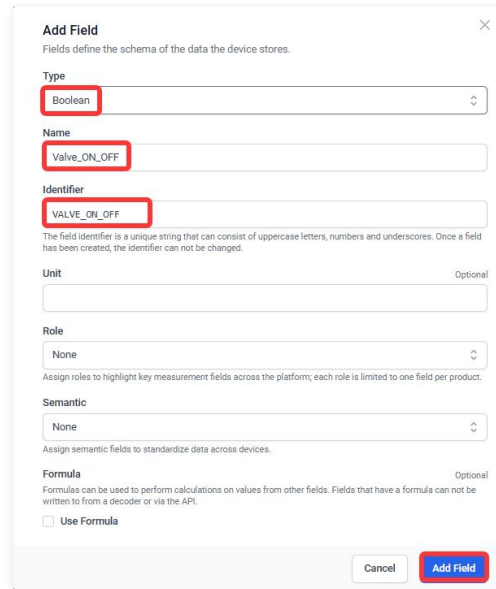
1

Cancel | **Save measurements and send downlink**

**Change the Valve ON/OFF:**

1、Click “Configuration-->Fields-->+Add Field”.





**Add Field**  
Fields define the schema of the data the device stores.

Type: **Boolean**

Name: **Valve\_ON\_OFF**

Identifier: **VALVE\_ON\_OFF**

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**

Assign roles to highlight key measurement fields across the platform; each role is limited to one field per product.

Semantic: **None**

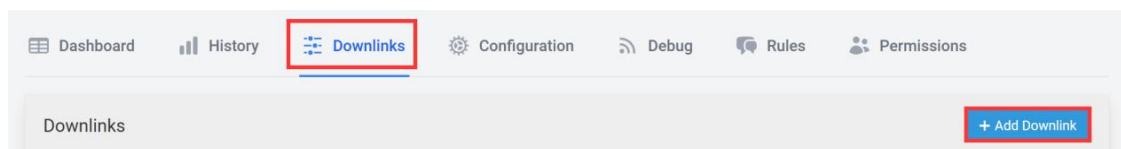
Assign semantic fields to standardize data across devices.

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

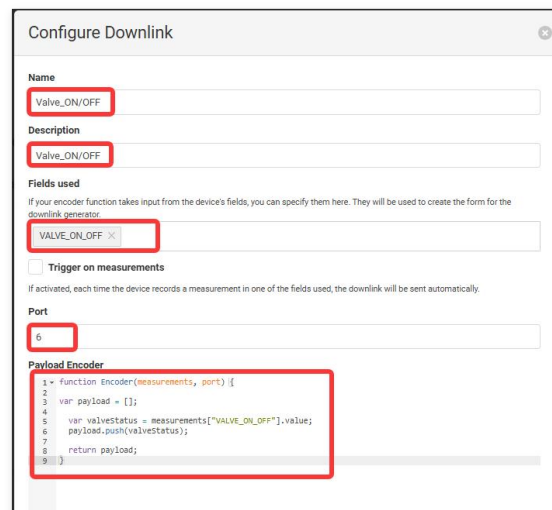
Cancel Add Field

2、Click "Downlink-->Add Downlink".



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

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



**Configure Downlink**

Name: **Valve\_ON/OFF**

Description: **Valve\_ON/OFF**

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.

**VALVE\_ON\_OFF** X

☐ Trigger on measurements  
If activated, each time the device records a measurement in one of the fields used, the downlink will be sent automatically.

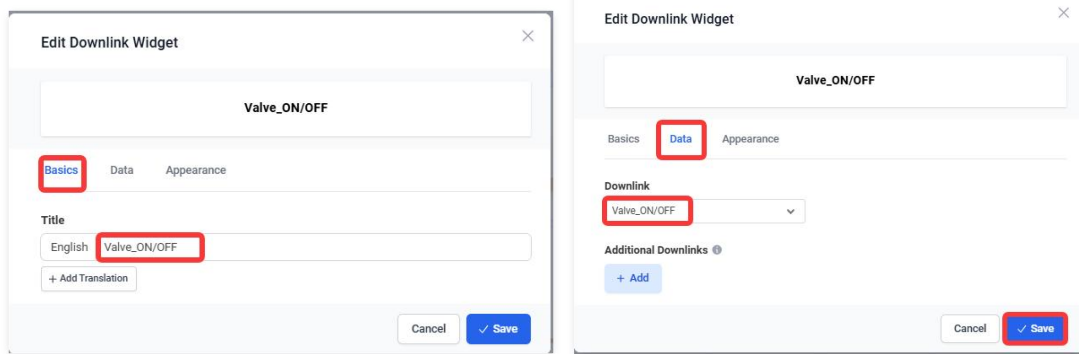
Port: **6**

Payload Encoder

```
1 * function Encoder(measurements, port) {
2
3   var payload = [];
4   var valvestatus = measurements["VALVE_ON_OFF"].value;
5   payload.push(valvestatus);
6   return payload;
7 }
8
9 }
```

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 Valve\_ON/OFF.

(Check the box to turn it on, otherwise it's off.)

