

SenseCap S2120 LoRaWAN Weather Station integration with the WaziGate

The SenseCap S2120 Weather Station by Seeed Studio is an exceptional addition to your Internet of Things (IoT) solution. This weather station is engineered to provide comprehensive weather data with a focus on meteorological measurements. Equipped with a range of environmental sensors, the S2120 delivers real-time weather information and ensures data accuracy through advanced technology.

The SenseCap S2120 incorporates state-of-the-art sensors for temperature, humidity, air pressure, wind speed, wind direction, rainfall, and more. These sensors work together seamlessly to capture precise meteorological data, which is then transmitted wirelessly to your designated IoT server using LoRaWAN technology. LoRaWAN ensures long-range data communication with minimal power consumption, making it a robust choice for remote and wide-area deployments.

Powered by a long-lasting battery, the S2120 weather station can operate continuously for an extended period, minimizing maintenance and enabling uninterrupted data collection. Its exceptional design and quality construction make it a reliable choice for various environmental monitoring applications.

The SenseCap S2120 Weather Station is the ideal solution for applications such as weather monitoring, environmental research, precision agriculture, and smart city projects. With its accurate and timely weather data, this weather station empowers your IoT infrastructure, enabling data-driven decisions for a wide range of industries and applications. Experience the future of weather monitoring with the SenseCap S2120 Weather Station by Seeed Studio.

1) Connecting to WaziGate - OTAA mode

Step 1 : Setup a WaziGate & Seeedstudio SenseCap S2120

Recommended Hardware:

- **WaziGate**
- **RAK 2247 LoRa HAT:** For LoRa Over The Air Activation (OTAA) a multichannel LoRa module is needed. It has benefits in many areas like:Improved Network Accessibility, Robustness, Enhanced Network Capacity, Load Balancing, Compatibility, Future-Proofing
- **Antennas for the selected frequency**
- **Micro SD-Card**

Setup WaziGate:

Here is a **tutorial** on how to **setup a Wazigate**:

<https://lab.waziup.io/resources/waziup/wazigate#prepare-the-gateway-hardware>

Setup SenseCap S2120:

In the following there are the steps explained to **change the connection settings of the SenseCap weather station**. It is done by using an Android or iOS app. The name of the App is SenseCap, you will find it in [PlayStore](#) or [AppStore](#).

After installing you can make a connection to the sensor, it is done via bluetooth, so you need to be close to the weather station.

The screenshots below show the procedure:

Screenshot 1: Device Bluetooth Configuration

This screenshot shows the main configuration menu. A red circle highlights the "S2120 Weather Station" option under "Device Type".

Screenshot 2: S2120 Weather Station Configuration

This screen provides initial setup steps. Step 1 shows connecting via Bluetooth. Step 2 shows updating the sensor's firmware. A red circle highlights the "Setup" button.

Screenshot 3: ScanList

This screen shows a list of detected devices. An S2120 Weather Station entry is highlighted with a red circle. A red circle also highlights the "Scan" button at the bottom.

Screenshot 4: Bluetooth Pairing

This screen shows the pairing process. It displays a PIN code (202003536224200425) and a keyboard for entering it. A red circle highlights the PIN field.

Screenshot 5: Advanced Configuration

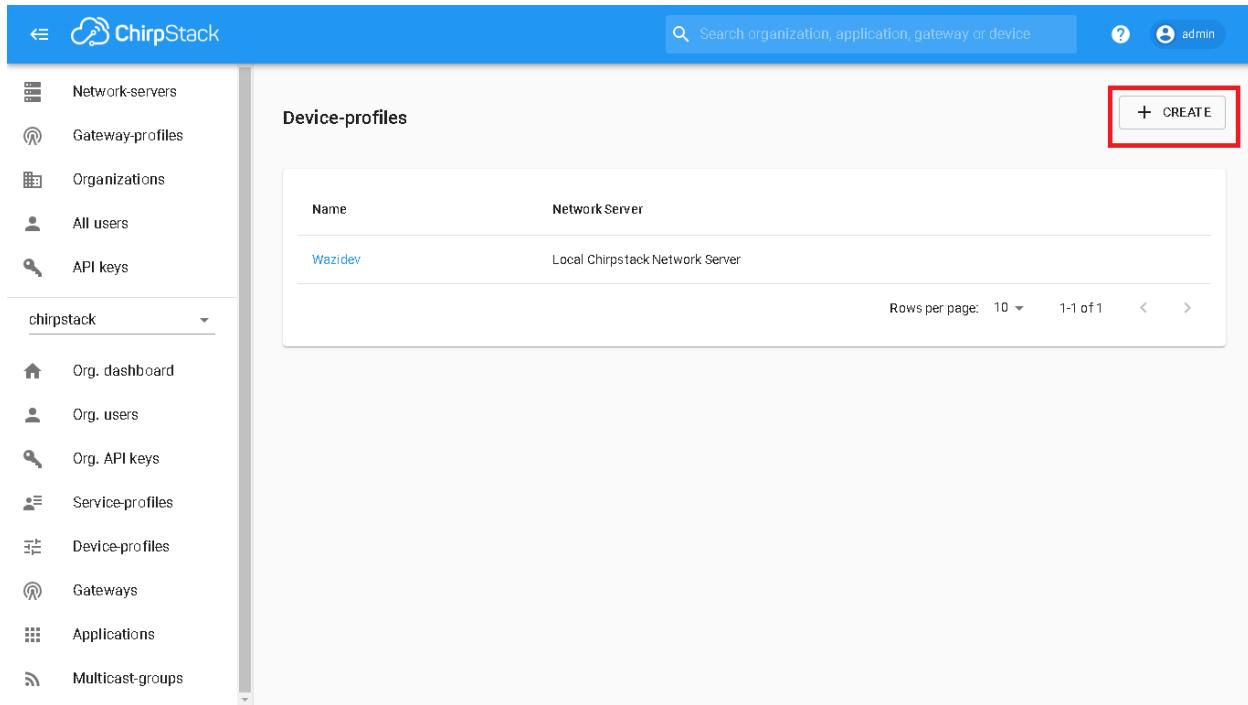
This screen shows detailed configuration settings for the device. A red circle highlights the "Advanced Configuration" mode, which is described as suitable for experienced users.

The default pin is : 000000

Step 2 : Create a new device profile on ChirpStack

Using a Web browser, open ChirpStack UI at wazigate.local:8080. The default username and password is admin admin.

Next, go to the “Device-profiles” and click the “+ Create” button.



The screenshot shows the ChirpStack UI interface. On the left, there is a sidebar with a tree view of the system structure. The main area is titled "Device-profiles" and contains a table with one row. The table has two columns: "Name" and "Network Server". The row shows "Wazidev" under "Name" and "Local Chirpstack Network Server" under "Network Server". At the top right of the main area, there is a search bar and a user account icon. Below the search bar is a red box highlighting the "+ CREATE" button, which is located in the top right corner of the table area. The bottom right of the table area shows pagination controls: "Rows per page: 10", "1-1 of 1", and navigation arrows.

Name	Network Server
Wazidev	Local Chirpstack Network Server

Now we need to enter the following details for the new device profile. Feel free to use a different “Device-profile name”

(a) For “GENERAL” enter the following information:

- Network-server * = Local ChirpStack Network Server
- LoRaWAN MAC version * = 1.0.3
- LoRaWAN Regional Parameters revision * = A
- Max EIRP * = 14
- Uplink interval (seconds) * = 1200

Device-profiles / Dragino_LDDS75_OTAA

GENERAL JOIN (OTAA / ABP) CLASS-B CLASS-C CODEC TAGS

Device-profile name *

Dragino_LDDS75_OTAA

A name to identify the device-profile.

LoRaWAN MAC version *

1.0.3

The LoRaWAN MAC version supported by the device.

LoRaWAN Regional Parameters revision *

A

Revision of the Regional Parameters specification supported by the device.

Max EIRP *

14

Maximum EIRP supported by the device.

Uplink interval (seconds) *

1200

The expected interval in seconds in which the device sends uplink messages. This is used to determine if a device is active or inactive.

UPDATE DEVICE-PROFILE

(b) For “JOIN (OTAA/ABP)” check the box “Device supports OTAA”

Device-profiles / Dragino_LDDS75_OTAA

GENERAL JOIN (OTAA / ABP) CLASS-B CLASS-C CODEC TAGS

Device supports OTAA

UPDATE DEVICE-PROFILE

(c) Leave the “CLASS-B” and “CLASS-C” fields unchecked

(d) For “CODEC” select “None”. This is because the latest Wazigate version only requires the LoRaWAN bytes for the Decoder function. This decoder function will be executed by WaziGate Edge and it will create the sensor values.

The screenshot shows the ChirpStack Device-profiles interface. On the left is a sidebar with navigation links like Network-servers, Gateway-profiles, Organizations, All users, API keys, and a dropdown for 'chirpstack'. The main area shows a device profile named 'Dragino_LDDS75_OTAA'. The 'CODEC' tab is currently selected. A section titled 'Payload codec' contains the instruction 'Select payload codec' and a note: 'By defining a payload codec, ChirpStack Application Server can encode and decode the binary device payload for you.' At the bottom right of this section is a blue 'UPDATE DEVICE-PROFILE' button. In the top right corner of the main area, there is a red 'DELETE' button.

However, if you intend to view the device data on ChirpStack you can select the “Custom JavaScript codec functions” and copy and paste this [code](#) into the custom codec section inside the codec tab of the device profile.

This screenshot shows the same ChirpStack Device-profiles interface as above, but with custom JavaScript code pasted into the 'Custom JavaScript codec functions' section of the 'CODEC' tab. The code is a function named 'Decode' that takes 'fPort' and 'bytes' as parameters and returns a JSON object. It includes logic to handle battery levels and distance calculations. Below the code, a note states: 'The function must have the signature function Decode(fPort, bytes) and must return an object. ChirpStack Application Server will convert this object to JSON.' The rest of the interface is identical to the first screenshot.

```

function Decode(fPort, bytes, variables) {
    // Decode an uplink message from a buffer
    // (array) of bytes to an object of fields.
    var len=bytes.length;
    var value=(bytes[0]<<8 | bytes[1]) & 0x3FFF;
    var batV=value/1000;//Battery,units:V
    var distance = 0;
    if(len==8)
    {
        value=bytes[2]<<8 | bytes[3];
        distance=(value)/distance,units:mm
        if(value<20)
            distance = "Invalid Reading";
    }
}
// Encode encodes the given object into an array of bytes.
// - fPort contains the LoRaWAN fport number
// - obj is an object, e.g. {"temperature": 22.5}

```

(e) Finally, click the “CREATE DEVICE-PROFILE” button. The new device profile will be listed under “Device-profiles”.

The screenshot shows the ChirpStack interface with the title "Device-profiles". On the left is a sidebar with various navigation options. The main area displays a table of device profiles:

Name	Description
Network Server	Local Chirpstack Network Server
Dragino_ABP	Local Chirpstack Network Server
Dragino_OTAA	Local Chirpstack Network Server
Wazidev	Local Chirpstack Network Server

At the bottom right of the table, there are pagination controls: "Rows per page: 10", "1-3 of 3", and navigation arrows.

Step 3 : Create a OTAA device for WaziGate using WaziGate Edge API

Using the WaziGate Edge API, we can [create a new device](#) that will be defined by a Device EUI and the “Other” LoRaWAN profile. We can run this [JavaScript code](#) to create the device on WaziGate. In the code, we give the Device EUI of the SenseCap S2120 LoRaWAN device. This key can be obtained from the sticker that is on the bottom of the arm that leads to the wind speed sensor. The credentials for connection can also be changed via the SenseCap Mate application, described in Step 1.



The JavaScript code also assigns a custom codec to this device using the WaziGate Edge codec's id. The codec has already been added to the WaziGate for you, you should replace the codec id in the JavaScript code to the one you will find under <http://wazigate.local/codecs>, if you get a message "Unauthorized", you just have to log in to the UI again (<http://wazigate.local>).

If you do not have a custom codec on WaziGate, you can still run the JavaScript code. Afterwards add a custom codec on WaziGate and assign this to the respective device.

Dashboard

wazigate.local says
new device.id: 64f85b6d68f3190768901b64

G Gateway dca632d397937c24 ID dca632d397937c24

W WaziDev ID 64f5c16768f319072f...

temperature 120 °C Thermometer Yesterday
relativehumidity 50 % Humidity sensor Yesterday
analogoutput 120 Yesterday

OK

Console

```
Welcome to Wazigate! This is a HEAD build. main.js:2
1.0.4
Connecting to mqtt... main.js:2
Apps loaded: > Array(3) main.js:2
Gateway ID: dca632d397937c24 main.js:2
MQTT Connected. main.js:2
> var resp = await fetch("/devices", {
  method: "POST",
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    // id: "5cede0d034b9f61" // let the server choose
    an id name: "Dragino_LDDS75_OTAA", // readable
device name
    "meta": {
      "codec": "64F5cab68f319072f52c149", // change with custom codec id: wazigate.local/codecs
      "lorawan": {
        "devEUI": "A84001498183B4D4", // enter the LoRaWAN device's EUI
        "profile": "Other" // we will use "Other" LoRaWAN profile
      }
    }
  });
// the device id will be returned
var deviceId = await resp.json();
alert(`new device.id: ${deviceId}`);
```

Dashboard

G Gateway dca632d397937c24 ID dca632d397937c24

W WaziDev ID 64f5c16768f319072f...

temperature 120 °C Thermometer Yesterday
relativehumidity 50 % Humidity sensor Yesterday
analogoutput 120 Yesterday

D Dragino_LDDS75_OTAA ID 64f85b6d68f3190768 ...

Console

```
Welcome to Wazigate! This is a HEAD build. main.js:2
1.0.4
Connecting to mqtt... main.js:2
MQTT Connected. main.js:2
Apps loaded: > (3) {<--, <--, <--} main.js:2
Gateway ID: dca632d397937c24 main.js:2
>
```

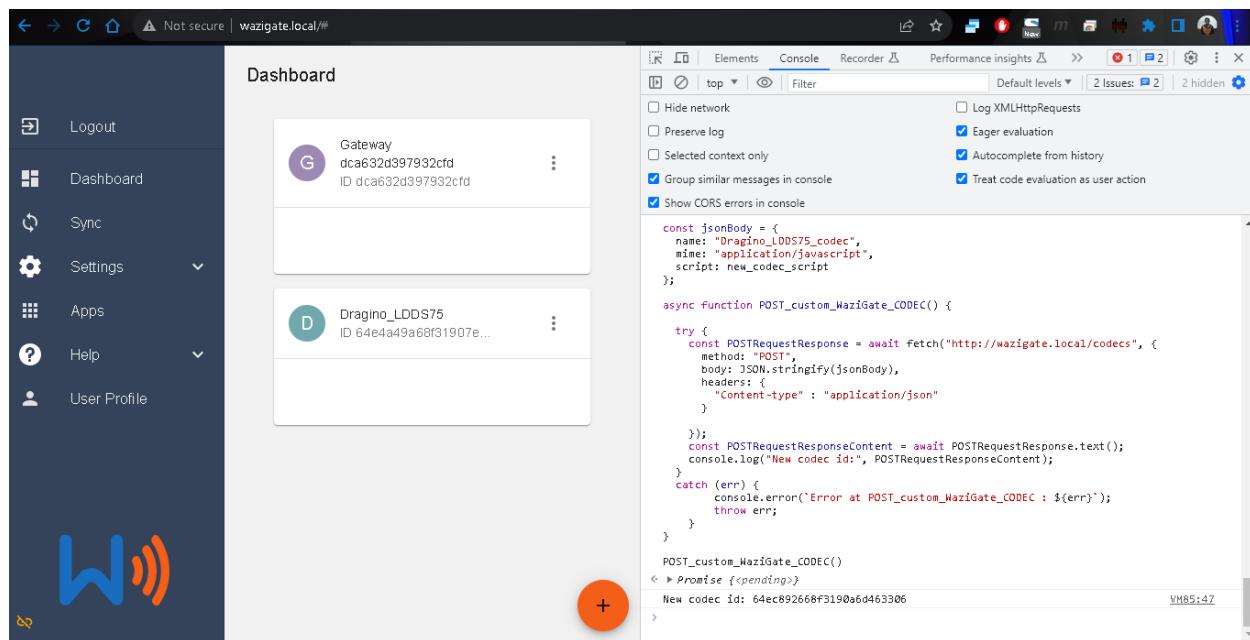
Step 4 : Select the custom WaziGate codec for the SenseCap

Since the SenseCap S2120 Codec is included in the WaziGate you just have to select it after creating the device.

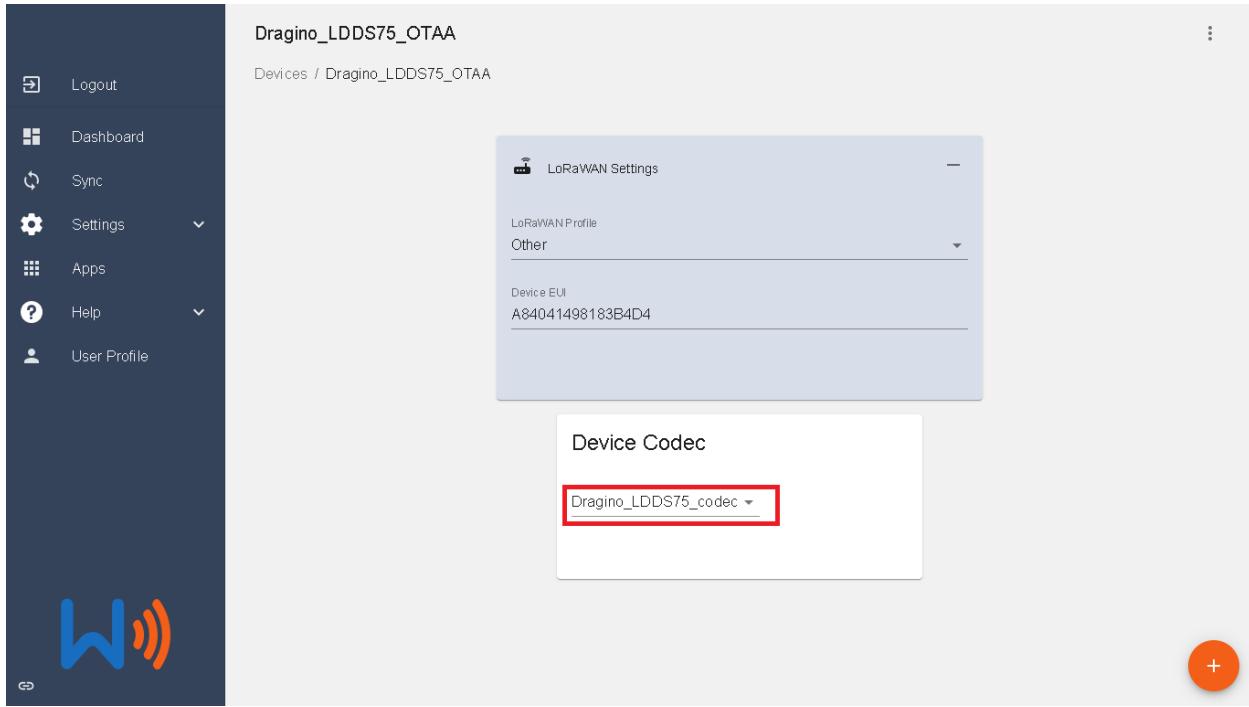
Upload codecs via the WaziGates API (**optional**, not needed for Seeedstudio SenseCap S2120):

Before we can save and view the sensor data on WaziGate, we need to add a custom decoder function for the bytes that WaziGate is receiving from ChirpStack. The WaziGate Edge GitHub repository has a [codec documentation](#) that shows how we can update the codecs. Note that the custom codec should have a Decoder function that is TTN format (chirpstack codec formats are also supported now).

First we need to add a custom codec to WaziGate. We can use this [Javascript code](#) for that. We can run the Javascript code on a web browser's console provided that the WaziGate is connected to the same network as the PC being used.



This will add a new codec called **SenseCap S2120** to WaziGate. The last step is to select this codec for the Dragino device that was setup on the WaziGate Dashboard.



Step 5 : Add the WaziGate device to ChirpStack

After creating the LoRaWAN device on WaziGate Dashboard, it does not appear on ChirpStack. To fix this we can use ChirpStack REST API to create the device for the Wazigate application.

First, open ChirpStack Application Server REST API Swagger documentation at
<http://wazigate.local:8080/api>

ChirpStack Application Server REST API

JWT TOKEN

ChirpStack Application Server REST API

For more information about the usage of the ChirpStack Application Server (REST) API, see <https://www.chirpstack.io/application-server/api/>.

ApplicationService		ShowHide List Operations Expand Operations
GET	/api/applications	List lists the available applications.
POST	/api/applications	Create creates the given application.
PUT	/api/applications/{application.id}	Update updates the given application.
GET	/api/applications/{application_id}/integrations	ListIntegrations lists all configured integrations.
DELETE	/api/applications/{application_id}/integrations/aws-sns	DeleteAWSNSIntegration deletes the AWS SNS application-integration.
GET	/api/applications/{application_id}/integrations/aws-sns	GetAWSNSIntegration returns the AWS SNS application-integration.
DELETE	/api/applications/{application_id}/integrations/azure-service-bus	DeleteAzureServiceBusIntegration deletes the Azure Service-Bus application-integration.
GET	/api/applications/{application_id}/integrations/azure-service-bus	GetAzureServiceBusIntegration returns the Azure Service-Bus application-integration.
DELETE	/api/applications/{application_id}/integrations/gcp-pub-sub	DeleteGCPPubSubIntegration deletes the GCP PubSub application-integration.
GET	/api/applications/{application_id}/integrations/gcp-pub-sub	GetGCPPubSubIntegration returns the GCP PubSub application-integration.
DELETE	/api/applications/{application_id}/integrations/http	DeleteIntegration deletes the HTTP application-integration.
GET	/api/applications/{application_id}/integrations/http	GetHTTPIntegration returns the HTTP application-integration.
DELETE	/api/applications/{application_id}/integrations/integration_id	DeleteIntegrationIntegrationId deletes the integration with the specified ID.

Next, scroll to the section “InternalService” and click “POST /api/internal/login”. Here, we need to provide the login details and a JWT token will be provided if the login credentials are correct. For default login credentials, we can paste the following JSON data in the POST request body.

Unset

```
{ "email": "admin", "password": "admin" }
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	{ "email": "admin", "password": "admin" }		body	Model Example Value

Parameter content type:

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ "email": "admin", "password": "admin" }'
```

Request URL

Response Body

```
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJheyIsImV4cCI6NTY5NDM4NTA5NCwiaQjEiMlzcyci6imFzIiwibmJmIjoxNjkzOTk4NjIwMTIwMDAwMDAwMAwzIiwiZW1haWQiOiJsb2dpbi5jb20udHJhZG9tYWluLmNvbSIsImF1ZGUiOiJsb2dpbi5jb20udHJhZG9tYWluLmNvbSJ9"
}
```

Response Code

Response Headers

Next we need to copy the JWT token and replace the api_token in this [python script](#) with it. Also make sure to change the values of the variables: device_name, dev_eui, app_key and device_profile_id accordingly. The device_profile_id should be the ID of the OTAA device profile that was previously created on ChirpStack.

The screenshot shows the ChirpStack web interface. The left sidebar has navigation links: Organizations, All users, API keys, chirpstack (dropdown), Org. dashboard, Org. users, Org. API keys, Service-profiles, Device-profiles (selected), Gateways, Applications, Multicast-groups. The main content area shows the Device-profiles / Dragino_LDDS75_OTAA page. The page has tabs: GENERAL, JOIN (OTAA / ABP), CLASS-B, CLASS-C, CODEC, TAGS. The GENERAL tab is active. It contains fields: Device-profile name * Dragino_LDDS75_OTAA (text input), LoRaWAN MAC version * 1.0.3 (text input), LoRaWAN Regional Parameters revision * A (text input), Max EIRP * 14 (text input), Maximum EIRP supported by the device. The top right of the page has a DELETE button.

Finally, run the python script on any computer that is connected to the same subnetwork as the WaziGate. The script will create a device on ChirpStack with the same Device EUI used for the WaziGate device. Afterwards, the script sets the Application Key for this device.

```
● PS E:\Work\1. Waziup\WaziGate\Third-party devices test\Dragino_LDDS75> python python_ChirpStack_API_integration.py
○ create_device_response status code : 200
● OTAA Device created successfully.
  set_otaa_key_response status code : 200
  OTAA AppKey set successfully.
○ PS E:\Work\1. Waziup\WaziGate\Third-party devices test\Dragino_LDDS75>
```

Last seen	Device name	Device EUI	Device profile	Link margin	Battery
2 days ago	AA555A0021011D01	aa555a0021011d01	Wazidev	n/a	n/a
n/a	Dragino_LDDS75_OTAA	a84041498183b4d4	Dragino_LDDS75_OTAA	n/a	n/a

Step 6 : Power on SenseCap S2120 Weather Station

To power the device insert the batteries. To restart the device, press the reset button. The device will send join requests to the gateway afterwards.

Step 7 : View SenseCapS2120 Weather Station on the WaziGate Dashboard

The SenseCapS2120 Weather Station join mode should be OTAA. This is the default join mode, but if it was changed, we can use the SenseCAP app to change it.

Once the SenseCapS2120 Weather Station turns on, it will join the WaziGate network and start transmitting sensor data.

ChirpStack

Network-servers
Gateway-profiles
Organizations
All users
API keys

chirpstack

Org. dashboard
Org. users
Org. API keys
Service-profiles
Device-profiles
Gateways
Applications
Multicast-groups

Applications / Wazigate / Devices / Dragino_LDDS75_OTAA

DETAILS CONFIGURATION KEYS (OTAA) ACTIVATION DEVICE DATA LORAWAN FRAMES

2:40:35 PM up
2:40:21 PM up
2:40:07 PM up
2:39:54 PM up
2:39:39 PM up
2:39:25 PM up
2:39:11 PM up
2:39:11 PM join

DELETE HELP PAUSE DOWNLOAD CLEAR

Logout
Dashboard
Sync
Settings
Apps
Help
User Profile

W

Dashboard

Gateway dca632d39793d0f6 ID dca632d39793f877
Gateway dca632d397937c24 ID dca632d397937c24

WaziDev ID 64f5c16760f319072f...
Dragino_LDDS75_OTAA ID 64f65b6d60f3190768...

temperature Thermometer 120 °C Yesterday
relativehumidity Humidity sensor 50 % Yesterday
analogoutput 120 Yesterday
Bat 3.404 23 seconds ago
Distance 20 23 seconds ago
Interrupt_status 1 23 seconds ago

+

