

ThingsBoard Integration Guide - LRS20100

Table of Contents

1. How to connect LRS20100 LoRaWAN Temperature and Humidity Sensor to ThingsBoard?	1
1.1. Introduction	1
1.2. Prerequisites	1
1.3. Create device on ThingsBoard	6
1.4. Check data on ThingsBoard	9
1.5. Conclusion	13

1. How to connect LRS20100 LoRaWAN Temperature and Humidity Sensor to ThingsBoard?

1.1. Introduction

The LRS20100 LoRaWAN Temperature and Humidity Sensor is a high-performance, multi-parameter sensor designed to monitor ambient temperature and humidity. Ideal for different use cases like offices, classrooms, hospitals, and public spaces, cold-chain refrigerators.

It combines advanced sensing, long battery life and long-range wireless communication in compact form-factor.

Key Features:

- Measures ambient temperature and humidity accurately
- LoRaWAN connectivity
 - Long-range, low-power wireless communication
 - Ideal for smart buildings and IoT integrations
- More than 10 years of battery life
- Excellent platform compatibility:
 - Works well with major LoRaWAN network servers and platforms like ThingsBoard, ChirpStack, The Things Stack and LORIoT

1.2. Prerequisites

LRS20100 uses LoRaWAN technology to send data to the ThingsBoard platform. To start with, you need to configure the LoRaWAN gateway and network server and make sure the data from the sensor can reach the network server. This guide uses ChirpStack open-source LoRaWAN Network Server.

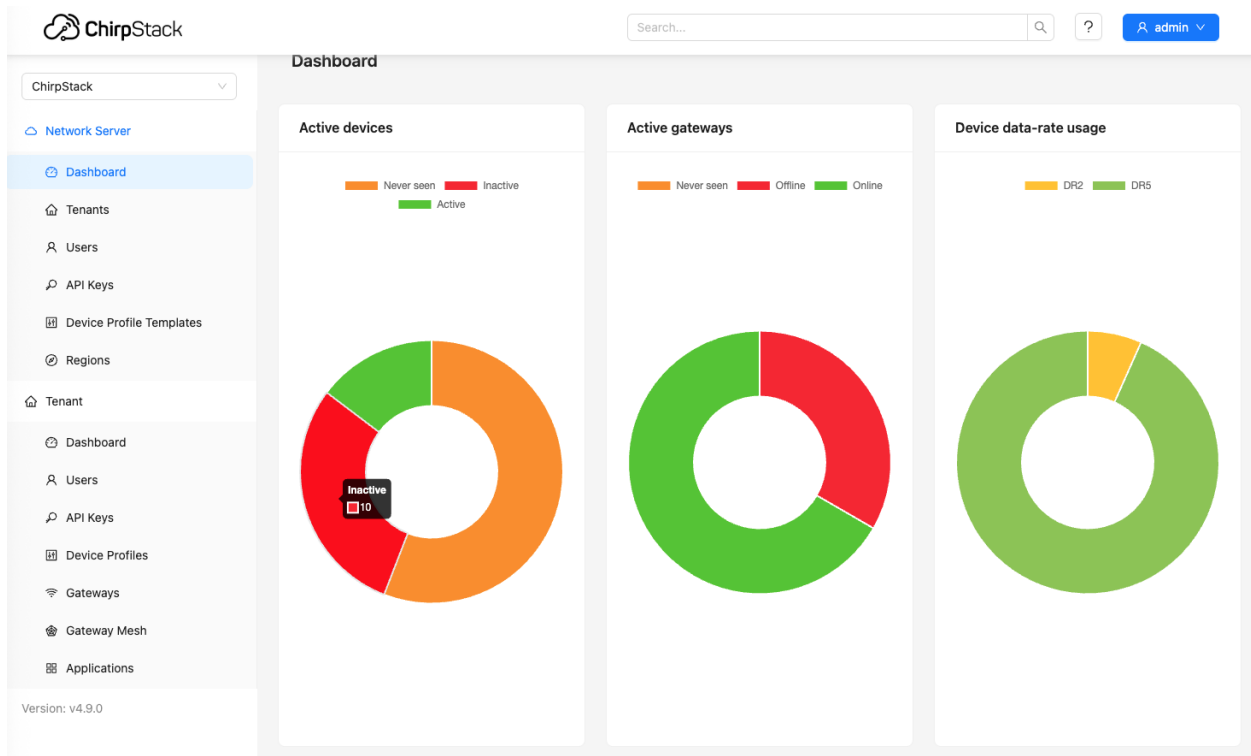
In this guide, we are using a sensor configured as OTAA as an example. To add a device to the

network server, you need to the following information:

- Device EUI
- Application EUI
- Application Key

The information is provided on the purchase of the sensor. Contact your sales or distributor for the information if needed.

To add the device on the ChirpStack, you need to login to the network server.



Go to Device profiles and click Add device profile

The screenshot displays the ChirpStack web interface. On the left is a sidebar with a navigation menu. The 'Device Profiles' link is highlighted with a green circle and a callout bubble labeled '1. Go to Device Profiles'. The main content area is titled 'Device profiles' and shows a table with columns: Name, Region, MAC version, Revision, Supports OTAA, Supports Class-B, and Supports Class-C. The table is currently empty, displaying 'No data'. A blue button labeled 'Add device profile' is located in the top right corner of the main content area, with a callout bubble labeled '2. Add device profile' pointing to it. The top of the interface features the ChirpStack logo, a search bar, and a user profile dropdown showing 'admin'.

Input the parameters on the Device Profile page and click Save. For Profile name, input 'LRS20100-OTAA' or 'LRS20100-ABP' depending on the sensor activation mode configured. It will be used on the data converter to determine the sensor model.

Tenants / SENSE08 / Device profiles / Add

Add device profile

General Join (OTAA / ABP) Class-B Class-C Codec Relay Tags Measurements [Select device-profile template](#)

* Name **3. Input a profile name**
 LRS10701-EU868-OTAA

Description

* Region **4. Select the region that match your sensor**
 EU868

Region configuration **5. Select the region configuration**
 EU868

* MAC version **6. Select LoRaWAN 1.0.3**
 LoRaWAN 1.0.3

* Regional parameters revision **7. Select A**
 A

* ADR algorithm **8. Leave the other parameters as default**
 Default ADR algorithm (LoRa only)

Flush queue on activate ☒ Allow roaming ☐

* Expected uplink interval (secs) **9. Click Submit**
 3600

Device-status request frequency (req/day)
 1

RX1 Delay (0 = use system default)
 0

[Submit](#)

Make sure the Payload codec under Codec for the Device Profiles is selected as 'None'.

General Join (OTAA / ABP) Class-B Class-C Codec Relay Tags Measurements [Select device-profile template](#)

Payload codec **9. Click Submit**
 None

[Submit](#)

Go to Applications, select an existing application or Add application.

ChirpStack

SENS08

Network Server

- Dashboard
- Tenants
- Users
- API Keys
- Device Profile Templates
- Regions
- Tenant**
 - Dashboard
 - Users
 - API Keys
 - Device Profiles
 - Gateways
 - Gateway Mesh
 - Applications**

Version: v4.9.0

Tenants / SENS08 / Applications

Applications

Add application

Name	Description
Arwin	

< 1 > 10 / page

11. Go to Applications

12. Select an application or Add application

Click Add device to add the new device.

ChirpStack

SENS08

Network Server

- Dashboard
- Tenants
- Users
- API Keys
- Device Profile Templates
- Regions
- Tenant**
 - Dashboard
 - Users
 - API Keys
 - Device Profiles
 - Gateways
 - Gateway Mesh
 - Applications**

Version: v4.9.0

Tenants / SENS08 / Applications / SENS08

SENS08 application id: c21135f8-f88c-4fd0-a29c-365d501de900

Delete application

Devices Multicast groups Relays Application configuration Integrations

13. Click Add device

Add device Selected devices

☐	Last seen	DevEUI	Name	Device profile	Battery
No data					

Input the sensor information and click Submit to add the sensor.

Tenants / SENS08 / Applications / SENS08 / Add device

Add device

Device Tags Variables

14. Input a name for the sensor

* Name

Description

15. Input the Device EUI of the sensor

* Device EUI (EUI64)

Join EUI (EUI64) ?

16. Select the profile just created

* Device profile

Device is disabled ?

Disable frame-counter validation ?

17. Click Submit to add the sensor

Submit

It will then jump to the OTAA keys page. Input the Application Key of the sensor and click Submit.

Tenants / SENS08 / Applications / SENS08 / Devices / Office IAQ sensor

Office IAQ sensor

device eui: 70b3d58c90001961

Delete device

Dashboard Configuration **OTAA keys** Activation Queue Events LoRaWAN frames

* Application key ?

18. Input the Application Key

Submit

19. Click Submit

1.3. Create device on ThingsBoard

Follow the guide on this page <https://thingsboard.io/docs/user-guide/integrations/chirpstack/> to create ChirpStack Integration on our Thingsboard account and configure integration on your ChirpStack application.

Import the JSON file `SENSO8_data_converter_for_chirpstack_integration.json` or copy the codes below for the payloadDecoder function.

```
var deviceType = metadata.deviceProfileName;
var fPort = metadata.fPort;
```

```
/**
Decodes the incoming payload and returns a structured object containing telemetry
data and attributes.

@param {byte[]} input - The raw payload received as an array of bytes.
@returns {Object} output - The structured output with decoded telemetry and
attributes.
*/

function decodePayload(input) {
  // Initialize the output object with empty attributes and telemetry for clarity.
  var result = { attributes: {}, telemetry: {} };

  // Decode serial number (SN) from the first 4 bytes of the payload.
  // Press '?' icon in the top right corner to learn more about built in helper
  functions and capabilities.
  result.attributes.sn = parseBytesToInt(input, 0, 4);

  // Extract the timestamp from metadata (represented in milliseconds).
  var timestamp = metadata.ts; // ts is the timestamp parsed from the incoming
  message's time, or returns the current time if it cannot be parsed.

  // Initialize an object to store decoded key/value telemetry data.
  var values = {};
  if ((deviceType === 'LRS10701-OTAA') || (deviceType === 'LRS10701-ABP')) {
    if (fPort === 10) {
      // Decode event from the 1st byte of the payload.
      values.event = parseBytesToInt(input, 0, 1);

      // Decode AQI, CO2 and temperature from the 2nd to 5th byte of the payload.
      var AQI_CO2_T = parseBytesToInt(input, 1, 4);
      values.aqi = AQI_CO2_T >> 23;
      values.co2 = (AQI_CO2_T >> 10) & 0x1FFF;
      values.temperature = ((AQI_CO2_T & 0x3FF) - 300) * 0.1;

      // Decode humidity from the 6th byte of the payload.
      values.humidity = parseBytesToInt(input, 5, 1) * 0.5;

      // Decode battery level and EC sensor resolution from the 11th byte of the
      payload.
      var ecr_bat = parseBytesToInt(input, 10, 1);
      var ec_res = ecr_bat >> 7;
      if (ec_res === 1)
        ec_scale = 10;
      else
        ec_scale = 1000;
      values.battery = ecr_bat & 0x7F;
    }
  }
}
```

```
// Decode gas sensor readings from the 7th to 8th and 9th to 10th byte of
the payload.
values.gas1 = parseBytesToInt(input, 6, 2)/ec_scale;
values.gas2 = parseBytesToInt(input, 8, 2)/ec_scale;
} else if (fPort === 11) {
  // Decode TVOC from the 1st and 2nd byte of the payload.
  values.tvoc = parseBytesToInt(input, 0, 2);
  // Decode PM1.0 from the 3rd to 5th byte of the payload.
  values["pm1.0"] = parseBytesToInt(input, 2, 3)/1000;
  // Decode PM2.5 from the 6th to 8th byte of the payload.
  values["pm2.5"] = parseBytesToInt(input, 5, 3)/1000;
  // Decode PM10 from the 8th to 11th byte of the payload.
  values.pm10 = parseBytesToInt(input, 8, 3)/1000;
}
}

if ((deviceType === 'LRS20100-OTAA') || (deviceType === 'LRS20100-ABP')) {
  if (fPort === 10 && parseBytesToInt(input, 0, 1) === 1) {
    // Decode event from the 2nd byte of the payload.
    values.event = parseBytesToInt(input, 1, 1);

    // Decode temperature from the 4th and 5th byte of the payload.
    var temperature = parseBytesToInt(input, 3, 2);
    if (temperature > 32767)
      temperature = temperature - 65536;
    values.temperature = temperature/10;

    // Decode humidity from the 6th and 7th byte of the payload.
    values.humidity = parseBytesToInt(input, 5, 2)/10;

    // Decode battery level the 3rd byte of the payload.
    values.battery = parseBytesToInt(input, 2, 1);
  }
}

// Combine the timestamp with values and add it to the telemetry.
result.telemetry = {
  ts: timestamp,
  values: values
};

// Return the fully constructed output object.
return result;
}

var result = decodePayload(payload);
```

```
return result;
```

And for advance decoding parameters, please refer to the screenshot below.

Advanced decoding parameters

Device profile name \$deviceProfileName	Device label \$deviceName
Customer name	Device group name

Telemetry

fCnt × data × rssi × snr × channel × fPort ×

Attributes

eui × tenantId × tenantName × applicationId × applicationName × adr ×

Update only keys list

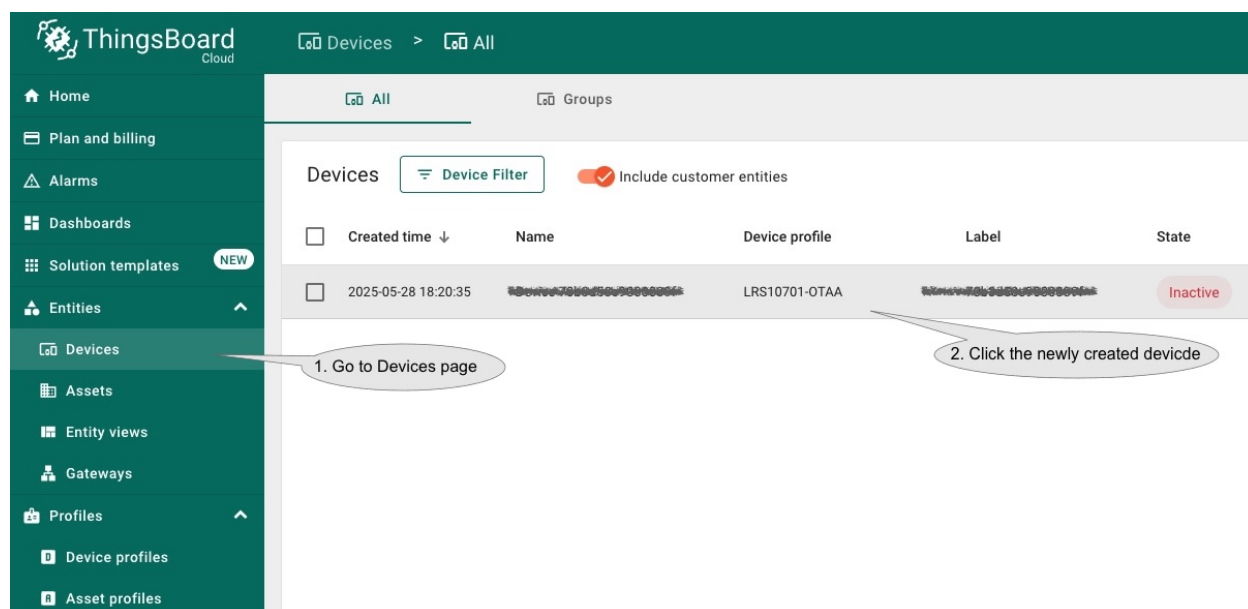
eui × devAddr × tenantId × tenantName × applicationId × applicationName × adr ×

The values associated with the provided keys will be saved to the database only if they are different from the corresponding values in the previous converted message. This functionality applies to both attributes and telemetry in the converter output.

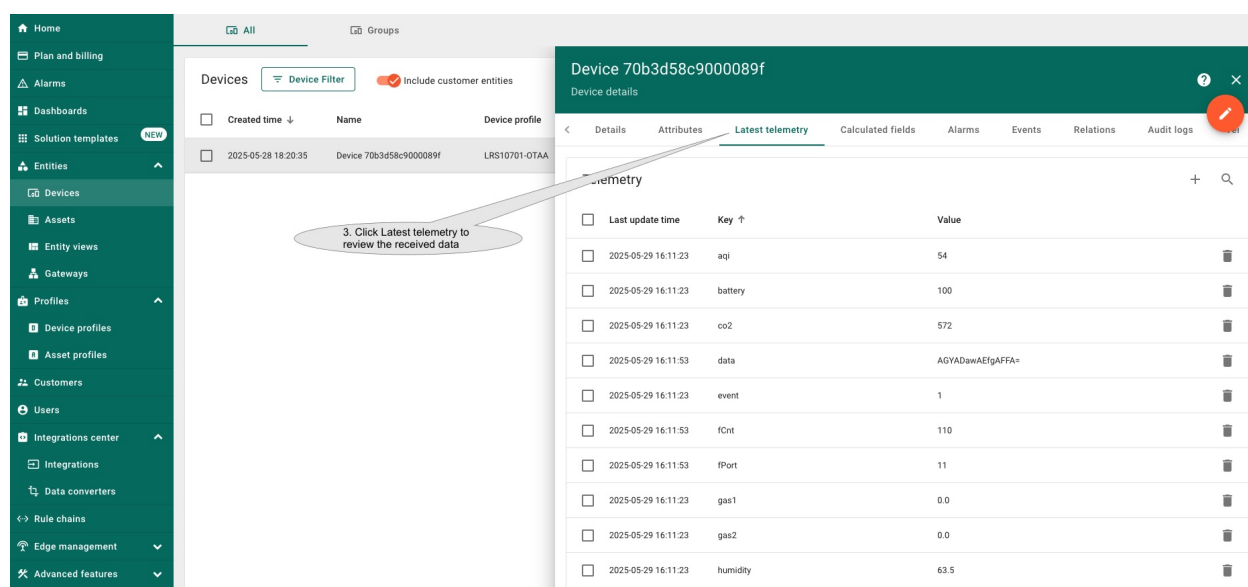
After the device has been added to ChirpStack and integration has been setup on both ThingsBoard and ChirpStack, a new device will appear on the ThingBoard user interface when the device sends any data.

1.4. Check data on ThingsBoard

To check the received data, you can go to Devices page under the Entitles. Then select the newly created device from the Device list.



The received data can be reviewed on Attributes and Latest telemetry pages.

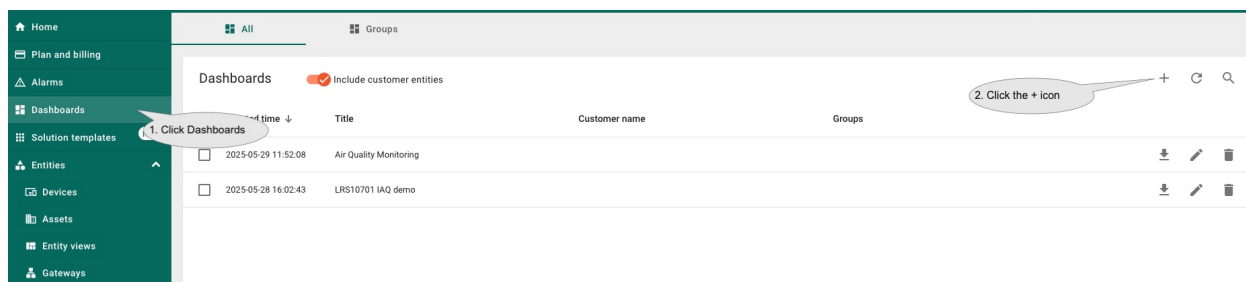


ThingsBoard provides a more user-friendly way to visualize your data with dashboards.

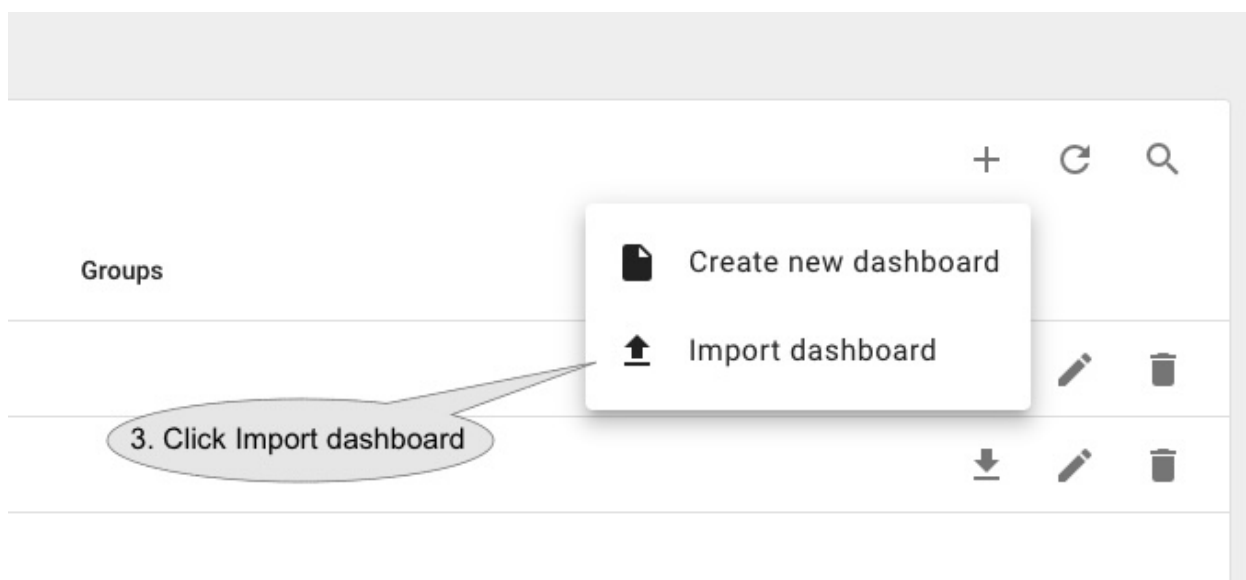
ThingsBoard allows you to create and customize your own dashboard to suit your needs.

If you want to experience the dashboard quickly, you can download this file and import it to your ThingsBoard to get a glimpse of the dashboard experience.

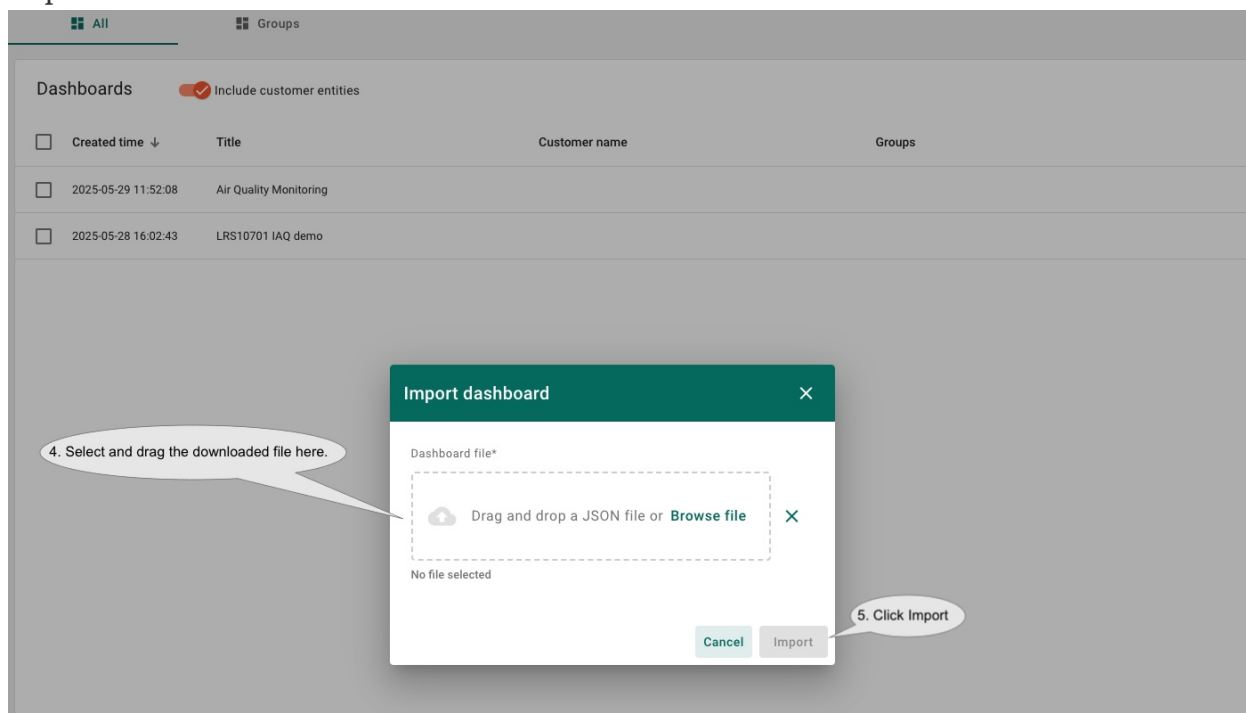
To import the dashboard, go to Dashboards and click the + icon.



Select Import dashboard



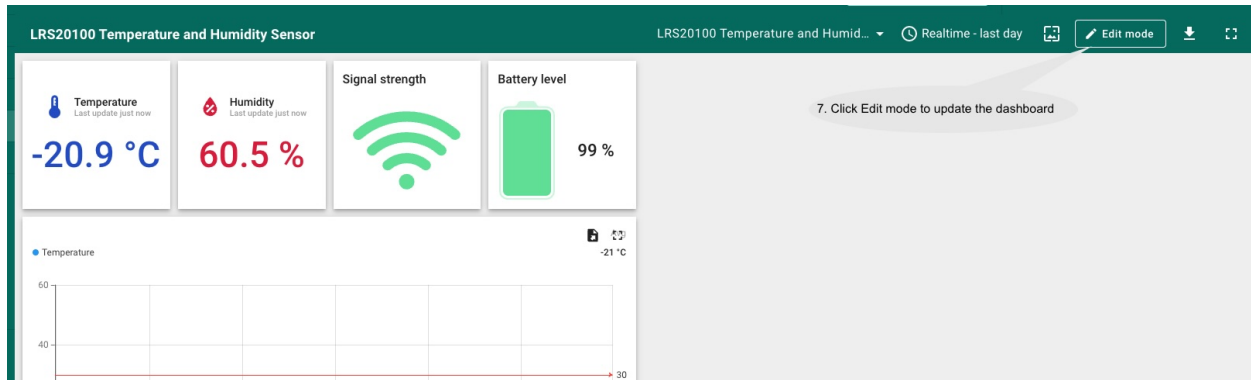
Select the download file or drag the file to the dialog box and click Import.



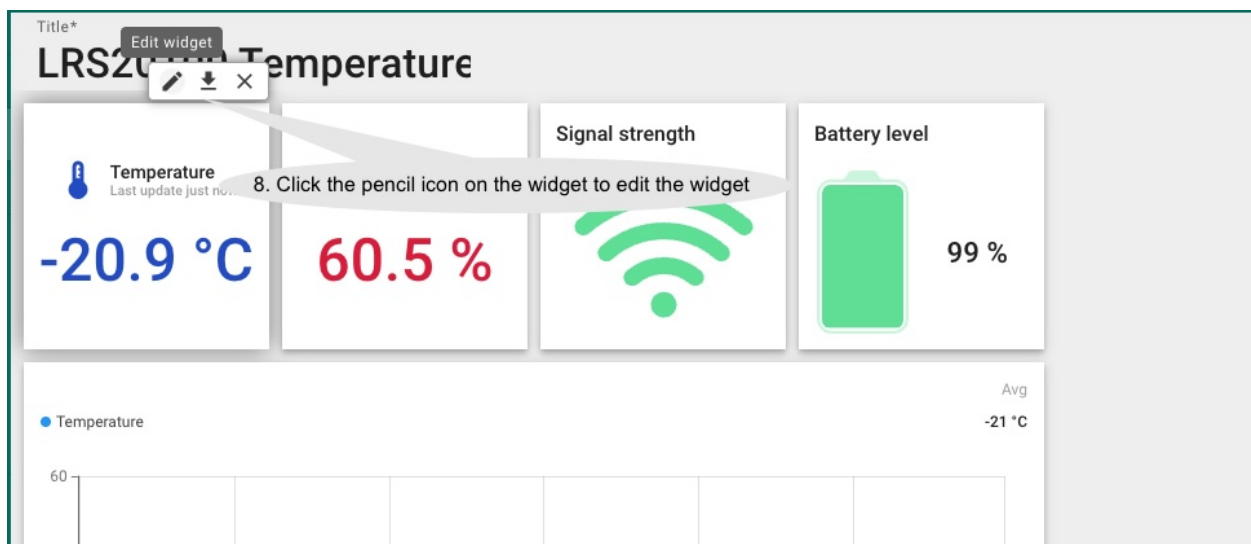
After importing the dashboard, it will go back to the dashboards list, click to select and open the imported dashboard.

Dashboards Include customer entities				
☐	Created time ↓	Title	Customer name	Groups
☐	2025-06-11 15:55:31	Temperature & Humidity		Customer dashboards
☐	2025-06-10 17:39:47	LRS20100 Temperature and Humidity Sensor		
☐	2025-05-30 15:55:18	LRS10701 Air Quality Monitor		
☐	2025-05-29 11:52:08	Air Quality Monitoring		
☐	2025-05-28 16:02:43	LRS10701 IAQ demo		

Click Edit mode to update the dashboard.



Move the cursor on one of the widget, a box will appear on the upper right corner. Click the pencil icon to edit it.



A dialog will pop up for entering the widget properties. Click Device under Datasource to select the device to be monitored. Then click Apply.

Air Quality Index

Air quality index chart card

Basic Advanced ? X

Preview Decline Apply

Time window

Use dashboard time window Use widget time window

Display time window

History - Current month so far

Datasource

9. Click Device under Datasource and select the device you want to monitor from the list

Device Entity alias

Device*
Device 70b3d58c9000089f X

Device 70b3d58c9000089f

aqi X

Appearance

Auto scale

Repeat the previous steps for other widgets.

Lastly, click Save to store the changes and the dashboard is then ready for monitoring the new device.

States Layouts + Add widget ⌚ ⚙️ 📄 ≡ ↺ X Cancel ✓ Save

Title*
LRS20100 Temperature

Temperature
Last update just now
-20.9 °C

Humidity
Last update just now
60.5 %

Signal strength

Battery level
99 %

Temperature
Avg
-21 °C

60

10. Click Save to save the changes

1.5. Conclusion

With the knowledge outlined in this guide, you can easily connect your LRS20100 LoRaWAN temperature and humidity sensor and visualize the data on ThingsBoard.

Explore the platform [documentation](#) to learn more about the key concepts and features of ThingsBoard for configuring [alarm rules](#) or [dashboards](#).