

# Connect a Decentlab DL-LP8P sensor over LoRaWAN to Ubidots

## Requirements

- [Decentlab: DL-LP8P](#)
- [The Things Network account](#)
- [Ubidots account](#)

## Step-by-Step

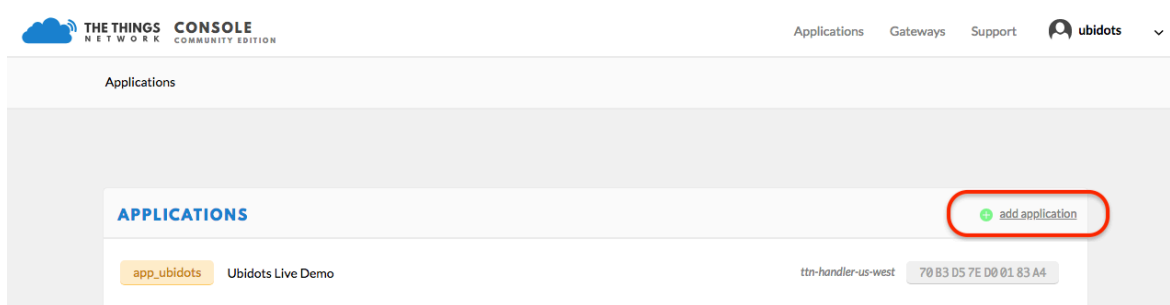
- 1 *TTN Device Registration*
- 2 *Uplink messages managementa. Custom payload setupb. Ubidots integration setup*
- 3 *Data visualization / Dashboard Creation*
- 4 *Summary*

### 1. TTN Device registration

**IMPORTANT NOTE:** To start managing your Decentlab devices' data with TTN as LoRaWAN network provider, please contact [support@decentlab.com](mailto:support@decentlab.com) to set up the requirements.

This **guide assumes your device is already connected and transmitting data to TTN** successfully. However, below you can find brief information on how to register a new device in TTN.

1. Go to the TTN console and enter to the application section to add a new application. To create the application, just press “**add application**”:



Then, in the following page, assign the parameters below and press “**Add application**” to continue:

- **Application ID:** The unique identifier of your application on the network.
- **Description (optional):** a human readable description.
- **Handler registration:** handler where you want to register the application.

**ADD APPLICATION**

**Application ID**  
The unique identifier of your application on the network

app-decentlab-lp8p

**Description**  
A human readable description of your new app

CO<sub>2</sub> Temperature, Humidity and Barometric Pressure Sensor

**Application EUI**  
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.

EUI issued by The Things Network

**Handler registration**  
Select the handler you want to register this application to

ttn-handler-us-west

CancelAdd application

Once the application is created, you will be redirected to the application overview.

THE THINGS  
NETWORK

CONSOLE  
COMMUNITY EDITION

ApplicationsGatewaysSupportubidots

Applications > app-decentlab-lp8p

OverviewDevicesPayload FormatsIntegrationsDataSettings

**APPLICATION OVERVIEW**

documentation

Application ID app-decentlab-lp8p

Description CO<sub>2</sub> Temperature, Humidity and Barometric Pressure Sensor

Created 30 seconds ago

Handler ttn-handler-us-west

2. To register a device, go to the device tab. Then press “**Register Device**”

Applications > app-decentlab-lp8p > Devices

OverviewDevicesPayload FormatsIntegrationsDataSettings

**DEVICES**

+ register device

Application app-decentlab-lp8p does not have any devices yet.

[Get started by registering one!](#)

Then, in the following page, assign the parameters below:

- **Device ID:** the unique identifier for the device in the application. The device ID will be immutable.
- **Device EUI:** the unique identifier for the device on the network.

The rest of the parameters (App Key, and App EUI) are automatically assigned by TTN. Once the device's registration is done, you will be redirected to the device overview. At this point, the device's status is "never seen" because it is waiting for its first message.

**DEVICE OVERVIEW**

Application ID

app-decentlab-lp8p

Device ID

dl\_lp8p\_1

Activation Method

OTAA

Device EUI

<> ⇄ 00 04 A3 0B 00 22 CD A8 ⓘ

Application EUI

<> ⇄ 70 B3 D5 7E D0 01 C2 BB ⓘ

App Key

<> ⇄ ⓘ

Status

• never seen

Frames up

0 [reset frame counters](#)

Frames down

0

Once the device receives the first message, you will see how the device's status change, as well as the log of the data sent by the device.

Applications > app-decentlab-lp8p > Devices > dl\_lp8p\_1 > Data

Overview Data Settings

### APPLICATION DATA

|| pause || clear

Filters: uplink downlink activation ack error

	time	counter	port	
▲	12:30:51	2	1	payload: 02 05 78 00 0F 67 BD 61 8D 1C EDBD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C 25
▲	12:30:39	1	1	payload: 02 05 78 00 0F 67 BD 61 8D 1C EDBD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C 25
▲	12:30:27	0	1	retry payload: 02 05 78 00 0F 67 BD 61 8D 1C EDBD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C 25
⚡	12:30:22			dev addr: 26 02 23 2C app eui: 70 B3 D5 7E D0 01 C2 BB dev eui: 00 04 A3 0B 00 22 CDA8

## 2. Uplink messages management

To start sending data (uplink messages) to Ubidots, you need to establish some configurations in the TTN side. Please follow the steps below carefully to establish the proper communication between your device, TTN, and Ubidots.

### • Custom Payload

1. Go to the Application overview, and refer to the “**Payload Formats**” tab. Then, assign the following code into the decoder section to format the payload in a JSON objects, which is the one allowed by Ubidots.

Applications > app\_sodaq > Payload Formats

Overview Devices **Payload Formats** Integrations Data Settings

## PAYLOAD FORMATS

**Payload Format**  
The payload format sent by your devices

Custom

decoder converter validator encoder

[remove decoder](#)

```

1 function Decoder(bytes, port) {
2   // Decode an uplink message from a buffer
3   // (array) of bytes to an object of fields.
4   var decoded = {};
5
6   // if (port == 1) decoded.lcd = bytes[0];
7
8   return decoded;
9 }

```

```

function Decoder(bytes, port) {
  return decentlab_decoder.decode(bytes);
}

```

```
/* https://www.decentlab.com/support */
```

```

var decentlab_decoder = {
  PROTOCOL_VERSION: 2,
  SENSORS: [
    {length: 2,
      values: [{name: 'Air-temperature',
        convert: function (x) { return 175.72 * x[0] / 65536 - 46.85; },
        unit: '°C'},
        {name: 'Air-humidity',
        convert: function (x) { return 125 * x[1] / 65536 - 6; },
        unit: '%'}]},
    {length: 2,
      values: [{name: 'Barometer-temperature',
        convert: function (x) { return (x[0] - 5000) / 100; },
        unit: '°C'},
        {name: 'Barometric-pressure',
        convert: function (x) { return x[1] * 2; },
        unit: 'Pa'}]},
    {length: 8,
      values: [{name: 'CO2-concentration',
        convert: function (x) { return x[0] - 32768; },
        unit: 'ppm'},
        {name: 'CO2-concentration-LPF',
        convert: function (x) { return x[1] - 32768; },
        unit: 'ppm'}]},
  ]
}

```

```

    {name: 'CC2-sensor-temperature',
      convert: function (x) { return (x[2] - 32768) / 100; },
      unit: '°C'},
    {name: 'Capacitor-voltage-1',
      convert: function (x) { return x[3] / 1000; },
      unit: 'V'},
    {name: 'Capacitor-voltage-2',
      convert: function (x) { return x[4] / 1000; },
      unit: 'V'},
    {name: 'CO2-sensor-status',
      convert: function (x) { return x[5]; }},
    {name: 'Raw-IR-reading',
      convert: function (x) { return x[6]; }},
    {name: 'Raw-IR-reading-LPF',
      convert: function (x) { return x[7]; }}}],
  {length: 1,
    values: [{name: 'Battery-voltage',
      convert: function (x) { return x[0] / 1000; },
      unit: 'V'}]}
],

read_int: function (bytes) {
  return (bytes.shift() << 8) + bytes.shift();
},

decode: function (msg) {
  var bytes = msg;
  var i, j;
  if (typeof msg === 'string') {
    bytes = [];
    for (i = 0; i < msg.length; i += 2) {
      bytes.push(parseInt(msg.substring(i, i + 2), 16));
    }
  }

  var version = bytes.shift();
  if (version !== this.PROTOCOL_VERSION) {
    return {error: "protocol version " + version + " doesn't match v2"};
  }

  var deviceId = this.read_int(bytes);
  var flags = this.read_int(bytes);
  var result = {'Protocol-version': version, 'Device-ID': deviceId};
  // decode payload
  for (i = 0; i < this.SENSORS.length; i++, flags >>= 1) {
    if ((flags & 1) !== 1)
      continue;

    var sensor = this.SENSORS[i];
    var x = [];
    // convert data to 16-bit integer array
    for (j = 0; j < sensor.length; j++) {
      x.push(this.read_int(bytes));
    }
  }
}

```

```

// decode sensor values
for (j = 0; j < sensor.values.length; j++) {
  var value = sensor.values[j];
  if ('convert' in value) {
    result[value.name] = {value: value.convert(x)};
  }
}
}
return result;
}
};

```

**IMPORTANT NOTE:** If you are using a different device [check out the decoders](#) provided by Decentlab and adapt it based on the [Ubidots API Requirements](#) to start sending data.

2. [OPTIONAL] Once you have the decoder code defined in TTN, you can test the payload which is being sent from the device to verify if the decoded value is right.

Let's say the value sent from the device is:  
 020578000F67BD618D1CEDBD10810D981F4895B0BD80BB500009598953900C25

Assign the same payload sent and the result should be in a JSON format payload:

```

1 function Decoder(bytes, port) {
2   return decentlab_decoder.decode(bytes);
3 }
4
5
6 /* https://www.decentlab.com/support */
7
8 var decentlab_decoder = {
9   PROTOCOL_VERSION: 2,
10  SENSORS: [

```

decoder has unsaved changes [undo changes](#)

**Payload**

02 05 78 00 0F 67 BD 61 8D 1C ED BD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C 25 32 bytes 1 [Test](#)

```

{
  "Air-humidity": {
    "value": 41.63221740722656
  },
  "Air-temperature": {
    "value": 24.35660461425781
  },
  "Barometer-temperature": {
    "value": 24.05
  }
}

```

• Payload was valid

3. To finish, press the **“Save payload functions”** button. At this point, if you refer to the data section you will notice how the data is being automatically decoded after saving the decoder function.

Applications > app-decentlab-lp8p > Devices > dl\_lp8p\_1 > Data

Overview Data Settings

### APPLICATION DATA

Filters: uplink downlink activation ack error

time	counter	port	payload
12:43:17	63	1	02 05 78 00 0F 67 BD 61 8D 1C ED BD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C

**Uplink**

**Payload**

02 05 78 00 0F 67 BD 61 8D 1C ED BD 10 81 0D 98 1F 48 95 B0 BD 80 BB 50 00 09 59 89 53 90 0C 25

**Fields**

```
{
  "Air-humidity": {
    "value": 41.63221740722656
  },
  "Air-temperature": {
    "value": 24.35660461425781
  },
  "Barometer-temperature": {
    "value": 24.05
  },
  "Barometric-pressure": {
    "value": 96800
  },
  "Battery-voltage": {
    "value": 36.876
  }
}
```

- **Ubidots integration setup**

1 Go to the “**Integration**” tab to add a new integration. To create a new integration, just press “**add integration**”.

Applications > app\_sodaq > Integrations

Overview Devices Payload Formats **Integrations** Data Settings

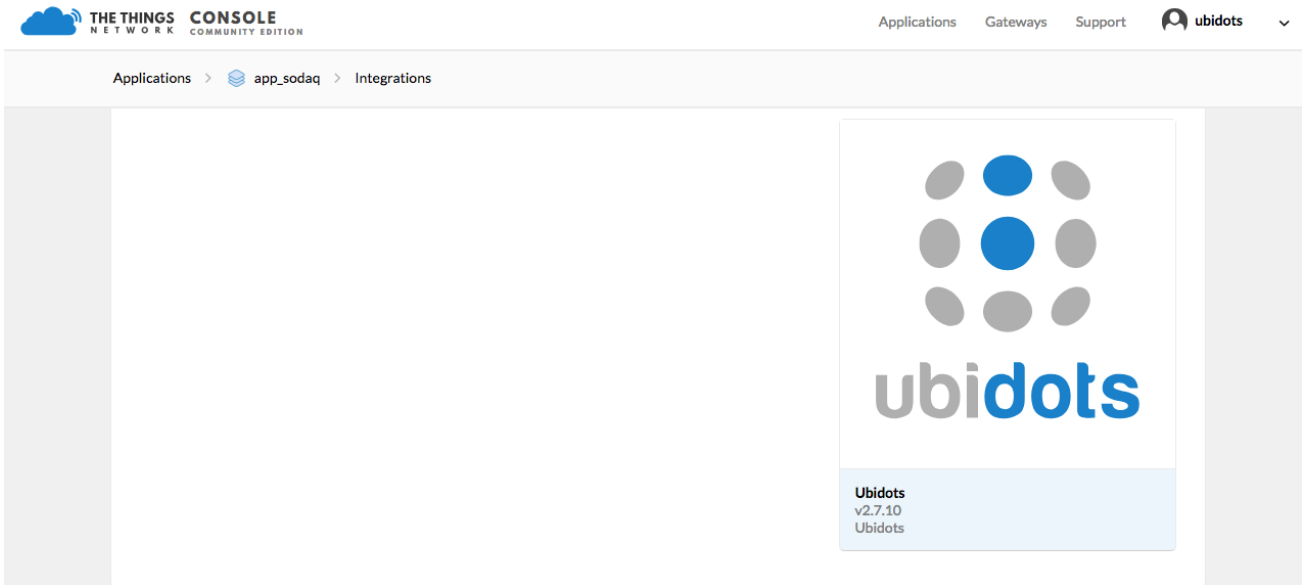
### INTEGRATIONS

[+ add integration](#)

There are no integrations for application app\_sodaq.  
[Get started by creating one!](#)

Then, select **Ubidots** as integration:





In the next window, assign the following parameters:

- **Process ID:** The unique identifier desired for the integration process.
- **Access Key:** The app access key.
- **Token:** Your Ubidots account Token. ([refer to this guide to know where to find it.](#))

With all the parameters assigned, your integration should look like the one below:

 The screenshot shows the 'Ubidots (v2.7.10)' integration configuration form. At the top, there are tabs for 'Overview', 'Devices', 'Payload Formats', 'Integrations' (which is active), 'Data', and 'Settings'. Below the tabs is a section titled 'ADD INTEGRATION' with the Ubidots logo and name. A description states: 'Learn to handle your The Things Network's account data with Ubidots to launch your IoT Control or Monitoring App.' with a link to 'documentation'. The form contains three input fields:
 


- Process ID:** Labeled 'The unique identifier of the new integration process', with the value 'decentlab\_uplink\_messages' entered and a green checkmark on the right.
- Access Key:** Labeled 'The app access key', with a dropdown menu showing 'default key' and options for 'devices' and 'messages'.
- Token:** Labeled 'Ubidots token', with the value 'BBFF-FWFRcjZns9rZQqPSnU72erdn6pSger' entered and a green checkmark on the right.

To finish, press **"Add integration"**. You'll be redirected to the integration overview:

## INTEGRATION OVERVIEW

Process ID `decentlab_uplink_messages`

Status ● Running

Platform  Ubidots (v2.7.10) [documentation](#)

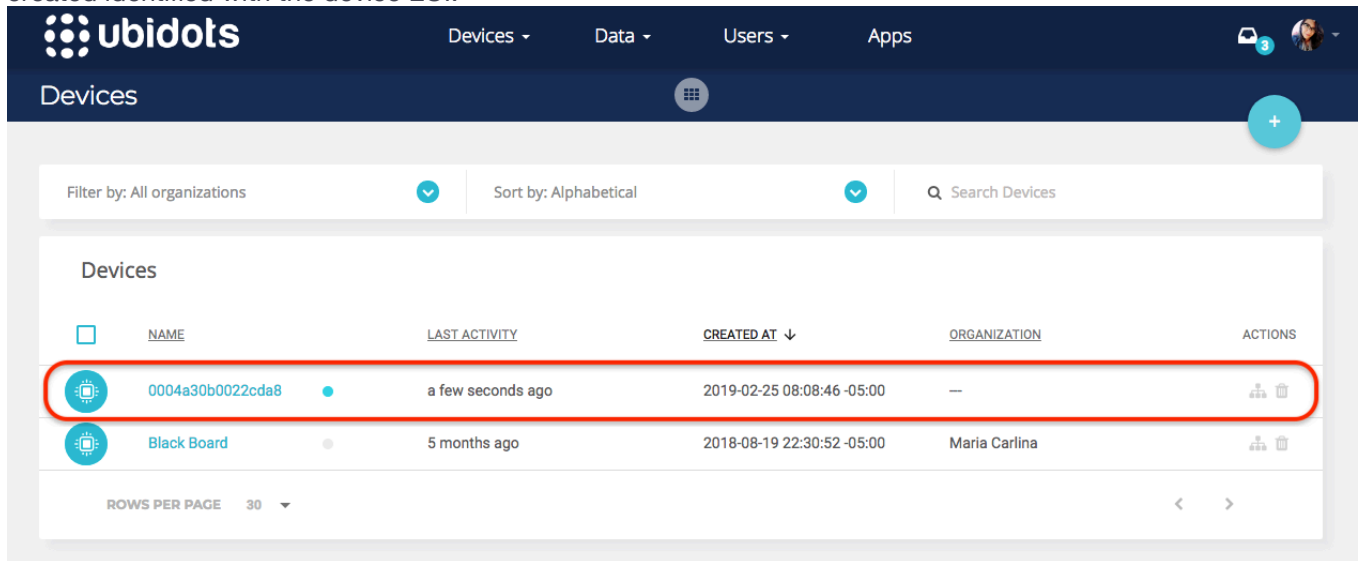
Author Ubidots

Description Learn to handle your The Things Network's account data with Ubidots to launch your IoT Control or Monitoring App.





At this point, once a new value is received in TTN from the device, a new device is going to be automatically created into your Ubidots account.

- **Data verification**

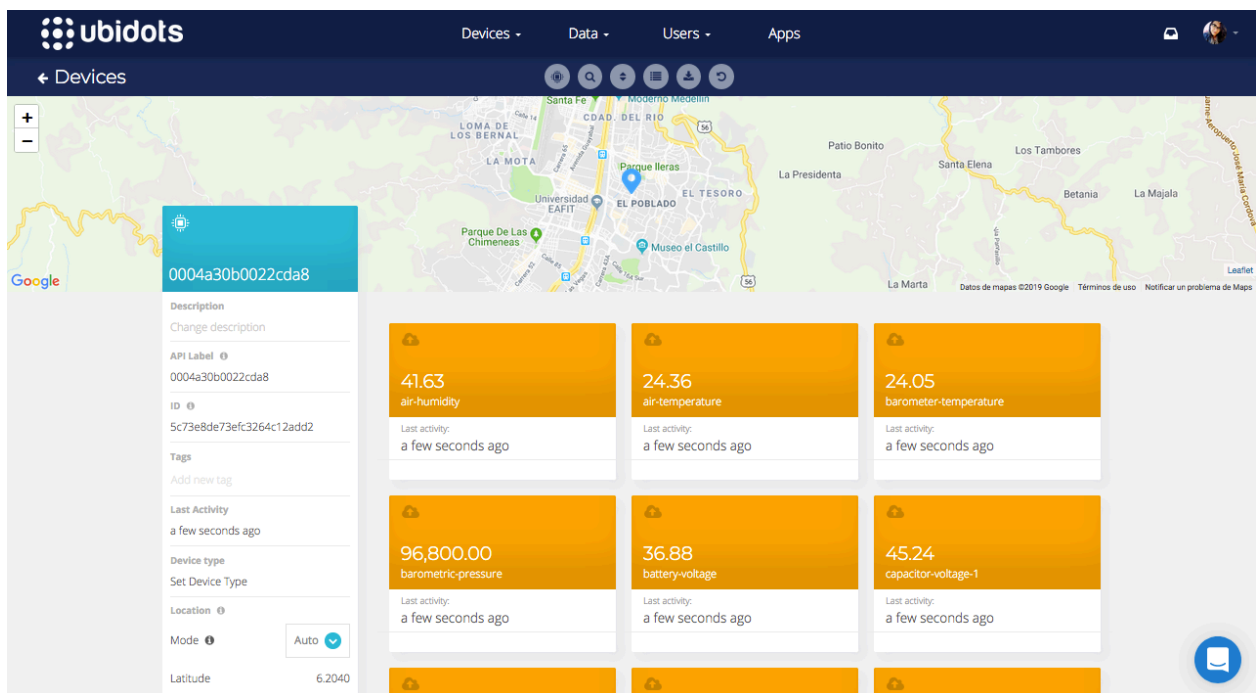
1. Go to the device section of your Ubidots account (**Devices > Devices**) to verify the new device created identified with the device EUI.



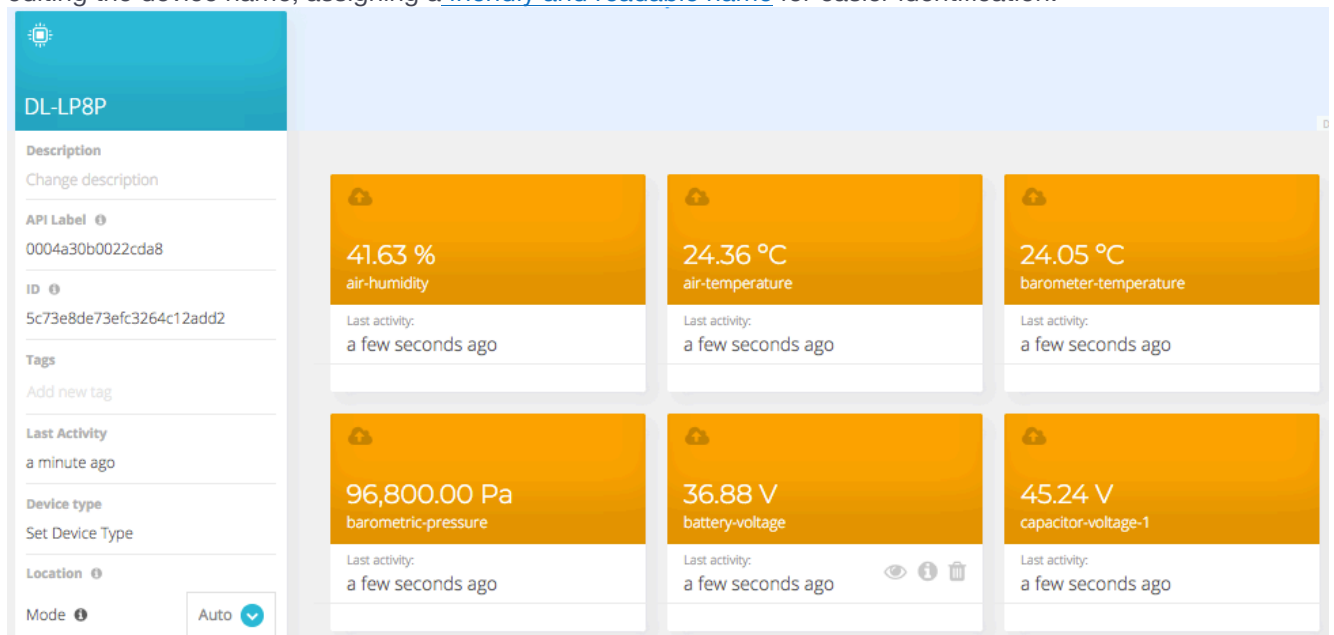
The screenshot shows the Ubidots web interface. The top navigation bar includes the Ubidots logo and links for Devices, Data, Users, and Apps. The 'Devices' section is active, showing a list of devices. The list has columns for NAME, LAST ACTIVITY, CREATED AT, ORGANIZATION, and ACTIONS. The first device, with EUI '0004a30b0022cda8', is highlighted with a red rectangle. The second device is 'Black Board'.

NAME	LAST ACTIVITY	CREATED AT	ORGANIZATION	ACTIONS
0004a30b0022cda8	a few seconds ago	2019-02-25 08:08:46 -05:00	—	 
Black Board	5 months ago	2018-08-19 22:30:52 -05:00	Maria Carlina	 

Select the device created to verify all the variables received:



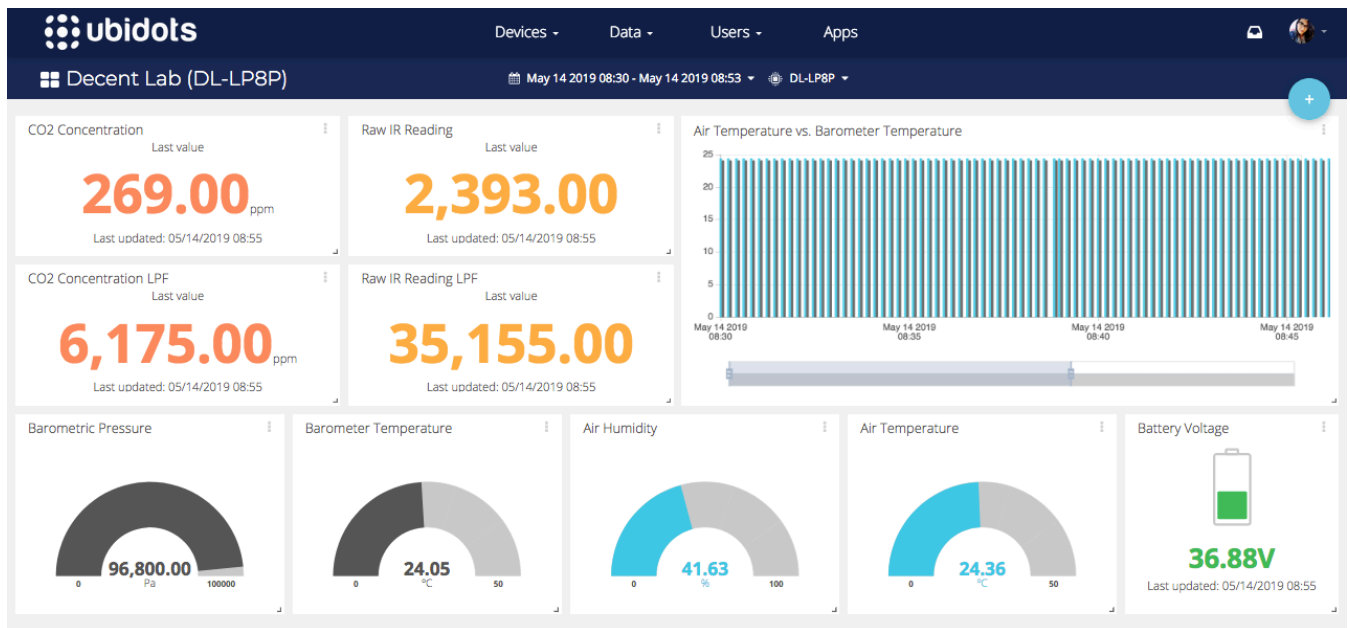
2. [ OPTIONAL - **PRO TIP**]: If you are using multiple devices to transmit data, we recommend editing the device name, assigning a [friendly and readable name](#) for easier identification:



### 3. Data visualization

Now it's time to build your own dashboard to start monitoring the data sensed by decentlab devices

1. **Go to the Dashboard (Data > Dashboards)** section of your Ubidots account.
2. Select the **plus (+)** icon located at the right-upper side of the page. Then select the widget types desired to display your data. [Learn more about Ubidots' Dashboards.](#)



## 6. Summary

LPWAN applications are having a huge impact on the IoT ecosystem, even more if the integrations required have a rapid set-up, like we just did with Decentlab sensor devices and Ubidots. We aim to enable the management of “things” easier for the IoT community, empowering more IoT engineers to [grow their deployments fast and reliably](#).