Qubitro Device Data - IoT Platform Series

Qubitro Device Data - IoT Platform Series

Difficulté **Facile** | Durée **1 heure(s)** | Catégories **Électronique** | Coût **0 USD ($)**

Sommaire

Introduction

Qubitro is an IoT (Internet of Things) platform that provides tools and services for connecting, managing, and analyzing IoT devices and data. It provides a cloud-based platform where users can securely connect their IoT devices and collect data from sensors and actuators.

It supports a wide range of communication protocols and provides device management capabilities, monitoring device data, linking with third-party webhooks, and creating rules to trigger based on conditions, etc. All of it with a Great UI ❤

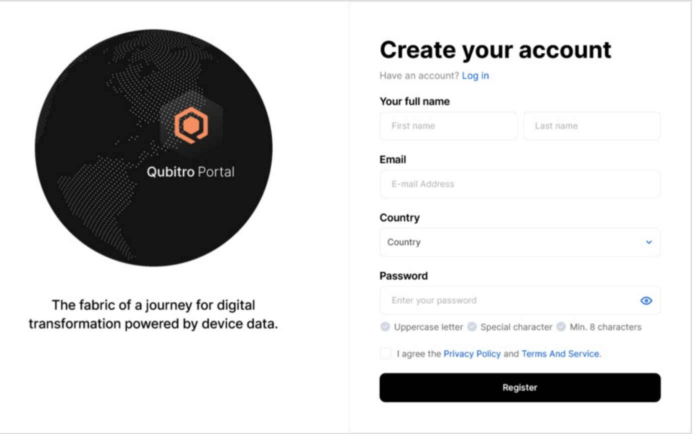
Matériaux

Outils

Étape 1 - Getting Started

To get started with Qubitro, we will first need to create an account. Go to the Qubitro website (https://www.qubitro.com/) and click on the "Sign Up" button. You will be prompted to enter *Full Name*, *Email Address,Country*, and *password* to create your account.

Once, we have created the account, we can log in from **https://portal.qubitro.com/login.** However, we shall automatically be logged in to our account.



Create your account

Have an account? Log in

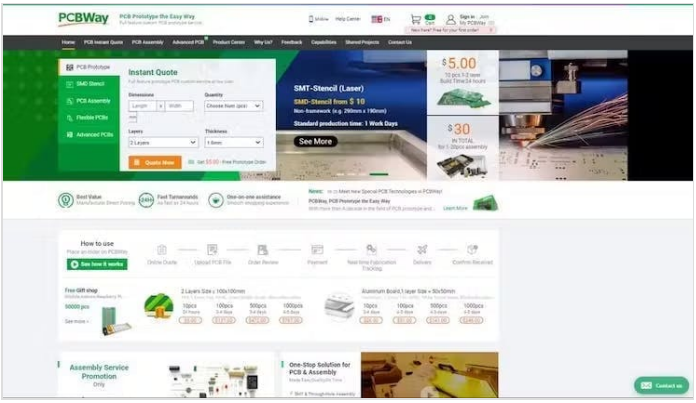
The fabric of a journey for digital transformation powered by device data.

I agree the Privacy Policy and Terms And Service.

Étape 2 - Get PCBs for Your Projects Manufactured

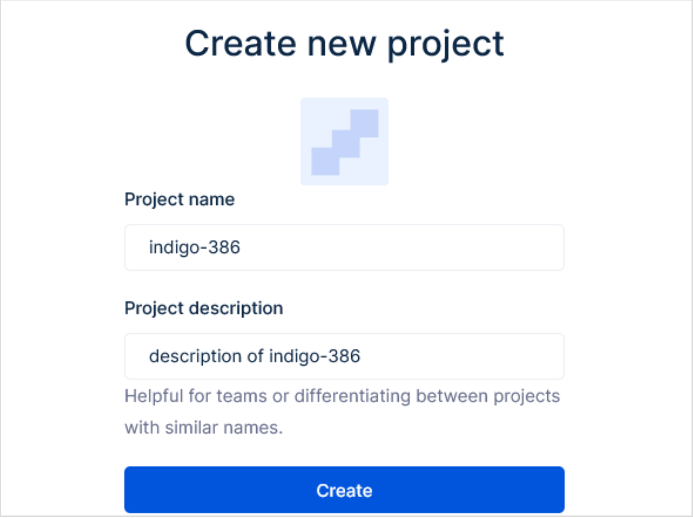
You must check out PCBWAY for ordering PCBs online for cheap!

You get 10 good-quality PCBs manufactured and shipped to your doorstep for cheap. You will also get a discount on shipping on your first order. Upload your Gerber files onto PCBWAY to get them manufactured with good quality and quick turnaround time. PCBWay now could provide a complete product solution, from design to enclosure production. Check out their online Gerber viewer function. With reward points, you can get free stuff from their gift shop.



Étape 3 - Create a New Project

Once you have logged in, you will be prompted to create a project. Enter a name for your project and mention a description for your project.



Create new project

Project name

indigo-386

Project description

description of indigo-386

Helpful for teams or differentiating between projects with similar names.

Étape 4 - Add Devices

Next, you will need to add devices to your application. Go to the Project (if not already open), there we can see a button **[+ New Source]**. Fron this section, we will have 3 major sections -

1. Communication Protocol

With a prompt to choose between LoRaWAN, MQTT, & Cellular. We can choose the protocol that best suits our use-case.

I choose MQTT to get started with the platform basics. And since I shall be using Arduino IDE for programming the board, I went ahead with the **MQTT Broker** (Qubitro has its own broker - we shall see it in the upcoming section). In case you wish to know how the Toit platform works, you can check my **Tutorial on Toit.io**

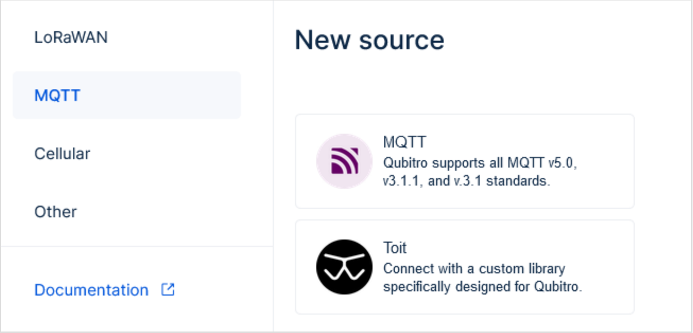
2. Device Details

I shall be using an ESP32 Dev Board, and therefore entered the details as per the image below -

3. Credentials

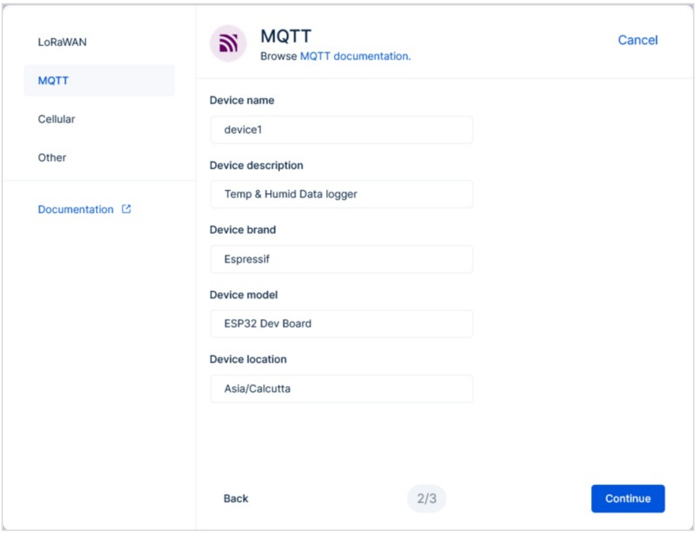
In the next step, we receive credentials, to connect to the MQTT Broker. We can use this detail to connect to the broker as a client - to Publish or Subscribe.

Now that we have the server, port, username and password we are all ready to send data to the Qubitro Cloud. **Copy these details in a safe place** (*We can view them later in the device settings as well though*)



MQTT
Qubitro supports all MQTT v5.0, v3.1.1, and v.3.1 standards.

Toit
Connect with a custom library specifically designed for Qubitro.



Device name: device1

Device description: Temp & Humid Data logger

Device brand: Espressif

Device model: ESP32 Dev Board

Device location: Asia/Calcutta

Étape 5 - Hardware - From Device to Cloud

Once you have configured your devices, you can start collecting data. Qubitro provides a range of tools for data collection and analysis, including real-time data visualization, data logging, and data filtering.

We shall upload a code on ESP32 using Arduino IDE to send data to Qubitro -

```
#include <WiFiClientSecure.h>
```

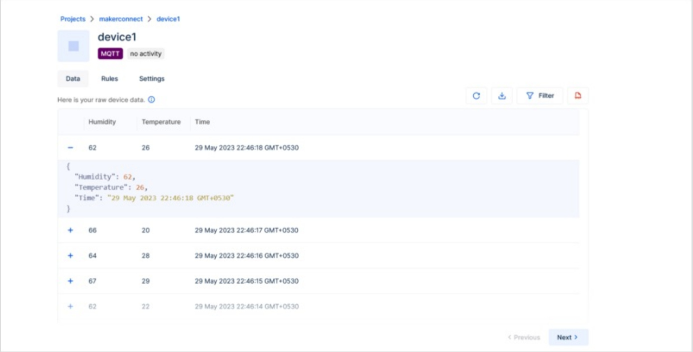
```
#include <PubSubClient.h>
```

```
#include <HTTPClient.h>
```

```
#include <ArduinoJson.h>
```

These are the necessary libraries for establishing an MQTT connection, handling HTTP requests, and working with JSON data.

```
const char* ssid = "xxxxxxxxx";
```



Here is your raw device data.

	Humidity	Temperature	Time
−	62	26	29 May 2023 22:46:18 GMT+0530
+	66	20	29 May 2023 22:46:17 GMT+0530
+	64	28	29 May 2023 22:46:16 GMT+0530
+	67	29	29 May 2023 22:46:15 GMT+0530
+	62	22	29 May 2023 22:46:14 GMT+0530

```
{
  "Humidity": 62,
  "Temperature": 26,
  "Time": "29 May 2023 22:46:18 GMT+0530"
}
```

```
const char* password = "xxxxxxxxx";
```

```
String topic = "xxxxx";
```

```
String mqtt_server = "broker.qubitro.com";
```

```
String mqttuser = "xxxxxxxxxxxxxxxxxxxxxx";
```

```
String mqttpass = "xxxxxxxxxxxxxxxxxxxxxx";
```

```
String clientId = "xxxxxxxxxxxxxxxxxxxxxx";
```

ssid These variables store the Wi-Fi credentials (and), MQTT password mqtt_server broker server address (), MQTT authentication credentials (and mqttuser mqttpass clientId), MQTT client ID (), and the topic MQTT topic () to which the data will be published.

```
WiFiClientSecure espClient;
```

```
PubSubClient client(espClient);
```

```
float humidity = 0;
```

```
float temp = 0;
```

WiFiClientSecure PubSubClient Create an instance of and classes to establish a secure connection with the MQTT broker. Also, initializing default value of temperature and humidity.

```
#define MSG_BUFFER_SIZE (500)
```

```
char msg[MSG_BUFFER_SIZE];
```

```
char output[MSG_BUFFER_SIZE];
```

Define the size of the message buffer for storing MQTT messages.

```
void device_setup() {
```

```
// ... Wi-Fi connection setup ...
```

```
}
```

This function sets up the Wi-Fi connection by connecting to the `ssid` `password` specified Wi-Fi network (and).

```
void reconnect() {
```

```
// ... MQTT reconnection logic ...
```

```
}
```

This function handles the reconnection to the MQTT broker in case of disconnection.

```
void setup() {
```

```
// ... Initialization code ...
```

```
}
```

`setup()` The function is the entry point of the code. It initializes the serial communication, sets up the device, establishes a connection with the MQTT broker, and prepares the secure `WiFiClientSecure` `PubSubClient` connection using and objects.

```
void loop() {
```

```
// ... Main code loop ...
```

```
}
```

`loop()` The function is the main execution loop of the code. It checks the MQTT connection, publishes the simulated temperature and humidity data to the MQTT topic, and then waits for a delay of 1 second before repeating the process.

`loop()` Inside the function, you'll notice the following steps:

- `if (!client.connected())` checks if the MQTT client is connected. If not, it calls the `reconnect()` function to establish the connection.
- `client.loop()` allows the MQTT client to maintain the connection and handle any incoming messages.
- `temp` `humidity` The and variables are randomly generated simulated values.
- A JSON document is created using the ArduinoJson library to store the temperature and humidity data.
- The JSON document is serialized into a string format using the

function and stored in the variable. `serializeJson()` `output`

- `client.publish()` The function is used to publish the serialized JSON data to the specified MQTT topic.
- The serialized JSON data is printed to the serial monitor using .
- `Serial.println()` A delay of 1 second is added before repeating the loop.

Full version of the code available in the **Code Section.**
Now that we have written the code, upload it to the ESP32 board and wait for it to send data to cloud.
To check data, go to Device Name that you created, and check for any incoming data in the table. (refresh the table in case data not retrieved)

Étape 6 - Create Dashboard

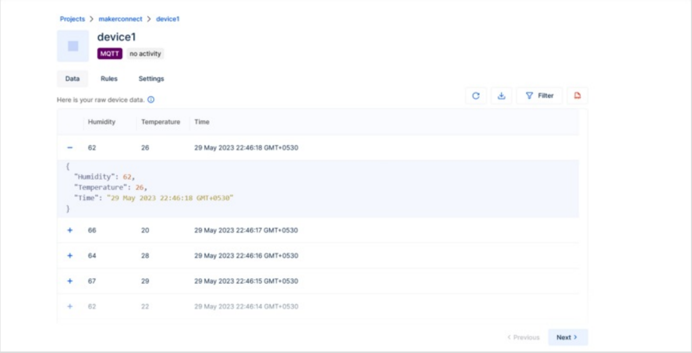
Now that we were able to fetch for real-time data from the ESP32 board and view it on the table of Qubitro. Let us use the visualization feature to plot a graph of the data. Trust me, it takes seconds to setup the whole thing.

- Go to Dashboards, and create a New Dashboard. Give it a name.
- Once created, open it and go to Edit > Add Widget > Charts.
- Click on the new widget > three dots (settings) > Customization.
- Accordingly, select the data source, chart type and colour for data variables. Follow the below images for reference, and final Graph.

Data source example above
Data Point example above
Finally, I received the above graph based on a 30-minute data logging. If we head back to the main dashboard page, we can have a proper view, and with a view configuration, receive live data in realtime on Qubitro.

- In the dashboard, click on the chart widget we created, click on edit and drag it to the middle.
- Stretch and play with the widget according to the need. Resizing it for proper viewing. Remember to save it.

If you are facing trouble with viewing the data with 4 points in the graph period, you can change it in the View Mode's configuration of the graph widget.
Now, using this we can view the data of our device based on our needs!

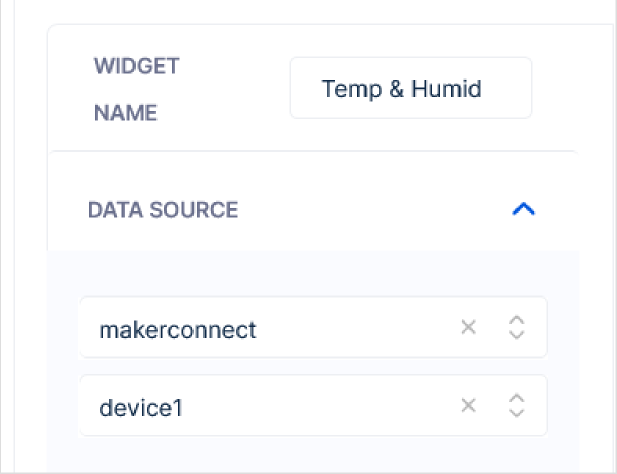


Here is your raw device data.

	Humidity	Temperature	Time
−	62	26	29 May 2023 22:46:18 GMT+0530

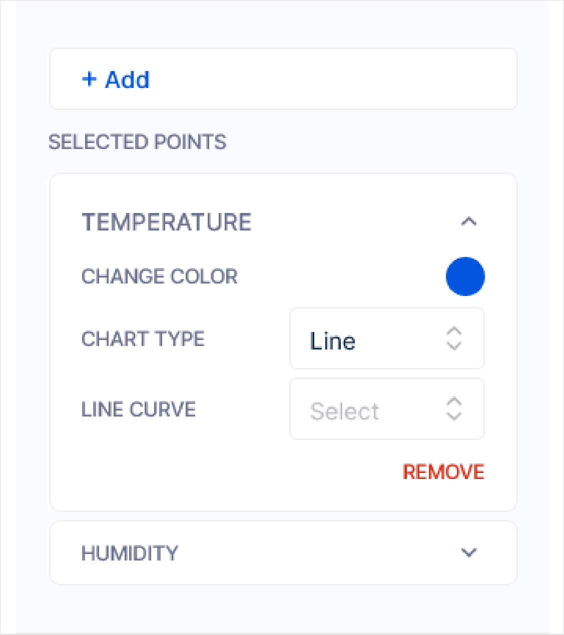
```
{
  "Humidity": 62,
  "Temperature": 26,
  "Time": "29 May 2023 22:46:18 GMT+0530"
}
```

	Humidity	Temperature	Time
+	66	20	29 May 2023 22:46:17 GMT+0530
+	64	28	29 May 2023 22:46:16 GMT+0530
+	67	29	29 May 2023 22:46:15 GMT+0530
+	62	22	29 May 2023 22:46:14 GMT+0530



WIDGET NAME: Temp & Humid

makerconnect
device1



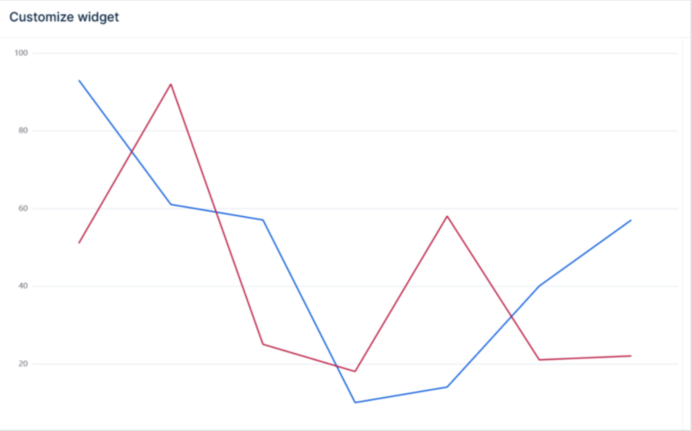
SELECTED POINTS

TEMPERATURE

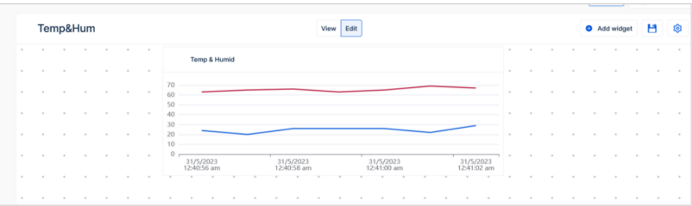
CHANGE COLOR ●

CHART TYPE: Line

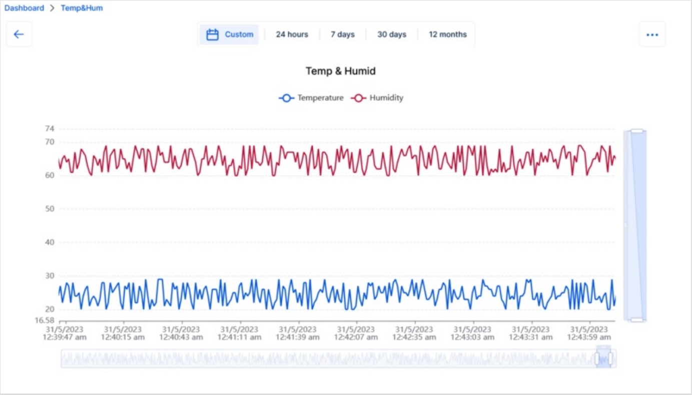
LINE CURVE: Select



Customize widget



Temp&Hum



Étape 7 - Rules to Trigger and Integration Services

Finally, Qubitro allows you to integrate with other services such as Twilio, Slack, MailGun, and SendGrid. We can also use the trigger for Webhooks (RAW HTTP request) triggering, You can do this by clicking on the "Rules" tab in the Device section and selecting the service you want to integrate with.

Congratulations! You have now completed the Qubitro IoT Platform documentation tutorial. We hope that this tutorial has provided you with the information you need to get started with Qubitro and create your own IoT application.

If you have any questions or need further assistance, please visit the Qubitro website or contact their support team.
Hurray! ?
We have learned another IoT Platform - Qubitro Device Data Platform

Étape 8 - Code (ESP32_MQTT_Qubitro)

```
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

// WiFi SSID and Password
const char* ssid = "xxxxxxxxx";
const char* password = "xxxxxxxxx";
String topic = "xxxxx";
String mqtt_server = "broker.qubitro.com";
String mqttuser = "xxxxxxxxxxxxxxxxxxxxxx";
String mqttpass = "xxxxxxxxxxxxxxxxxxxxxx";
String clientId = "xxxxxxxxxxxxxxxxxxxxxx";

WiFiClientSecure espClient;
PubSubClient client(espClient);

float humidity = 0;
float temp = 0;



#define MSG_BUFFER_SIZE (500)
char msg[MSG_BUFFER_SIZE];
char output[MSG_BUFFER_SIZE];

void device_setup() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

}

void reconnect() {
  // Loop until we’re reconnected
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection…");

    if (client.connect(clientId.c_str(), mqttuser.c_str(), mqttpass.c_str())) {
      Serial.print("MQTT connected");
    } else {
      Serial.print("failed, rc = ");
      Serial.print(client.state());
      Serial.println(", try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {
  delay(500);
  // When opening the Serial Monitor, select 9600 Baud
  Serial.begin(115200);
  delay(500);
  device_setup();
  espClient.setInsecure();//skip verification
  espClient.setInsecure();//skip verification
  //Serial.println(mqtt_server);
  client.setServer(mqtt_server.c_str(), 8883);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  temp = random(20,30);
  humidity = random(60,70);

  StaticJsonDocument<200> doc;
  doc["Temperature"] = temp;
  doc["Humidity"] = humidity;
  serializeJson(doc, output);
  doc.garbageCollect();

  client.publish(topic.c_str(), output);
  Serial.println(output);

  delay(1000);
}
```