

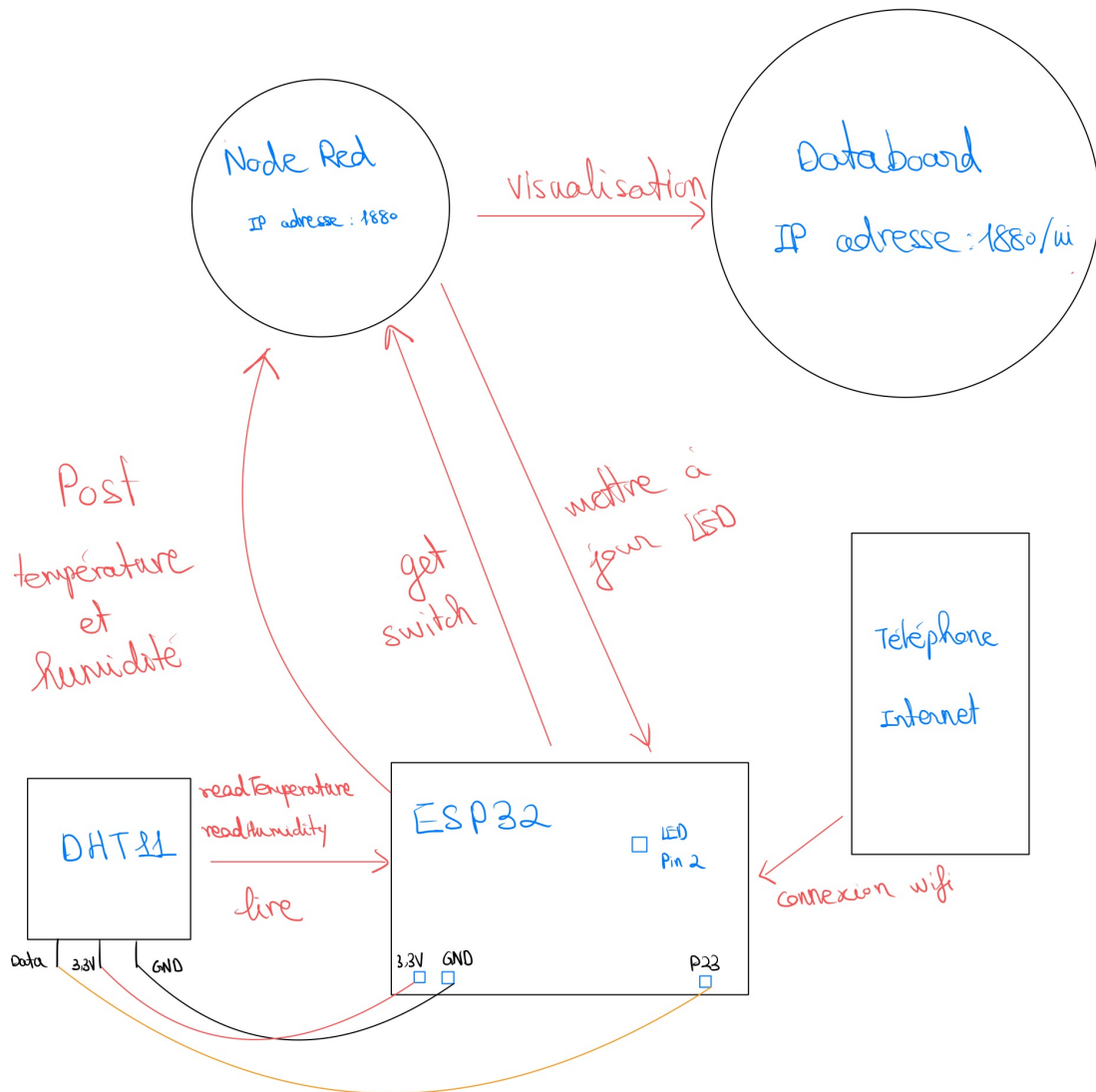
Notice

Guide de reproduction de notre système de communication bidirectionnelle basée sur le protocole REST

Description des éléments:

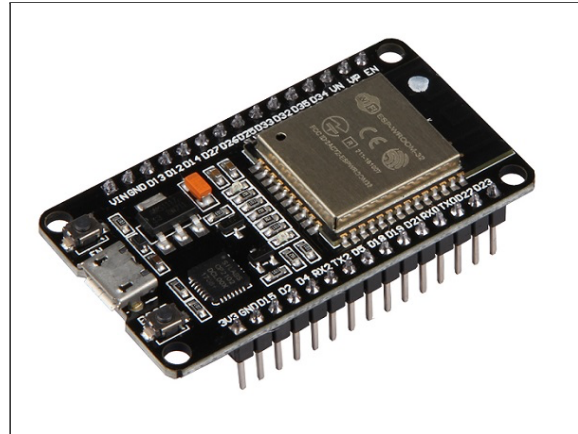
L'objectif de ce système est de réaliser une communication bidirectionnelle entre le microcontrôleur **ESP32** et le cloud (**Node RED**) basé sur le protocole **REST**. Ce système est composé comme l'architecture en dessous par les éléments suivants:

Architecture principale:

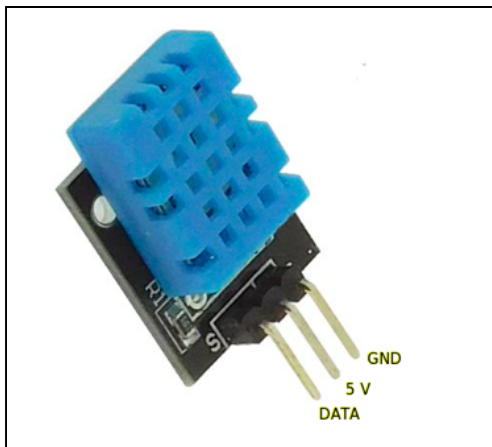


ESP32:

C'est une série de microcontrôleurs de type système sur une SoC intégrant la gestion du Wi-Fi et du Bluetooth. Elle est liée avec la capteur DHT11 et communique avec node red.



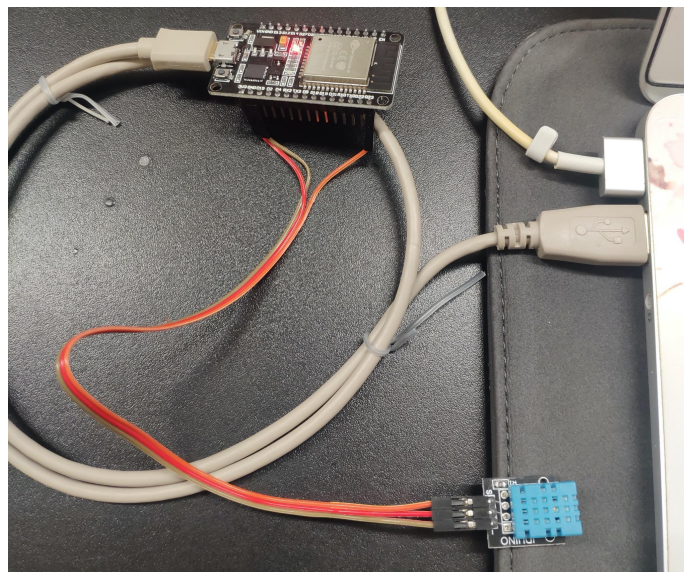
DHT11:



C'est la capteur qui permet de lire la température et l'humidité. Elle est liée avec la microcontrôleur ESP32. (GND -> GND, 5V -> 3.3V, DATA -> D23)

Installation du système:

Pour installer le système, tout d'abord il faut lier la capteur DHT11 avec la microcontrôleur ESP32 et lier la ESP32 avec l'ordinateur comme la figure suivant:



Utilisation du système:

Afin d'utiliser le système et réaliser la communication, il faut installer 2 logiciels, ce sont Node RED et Arduino.

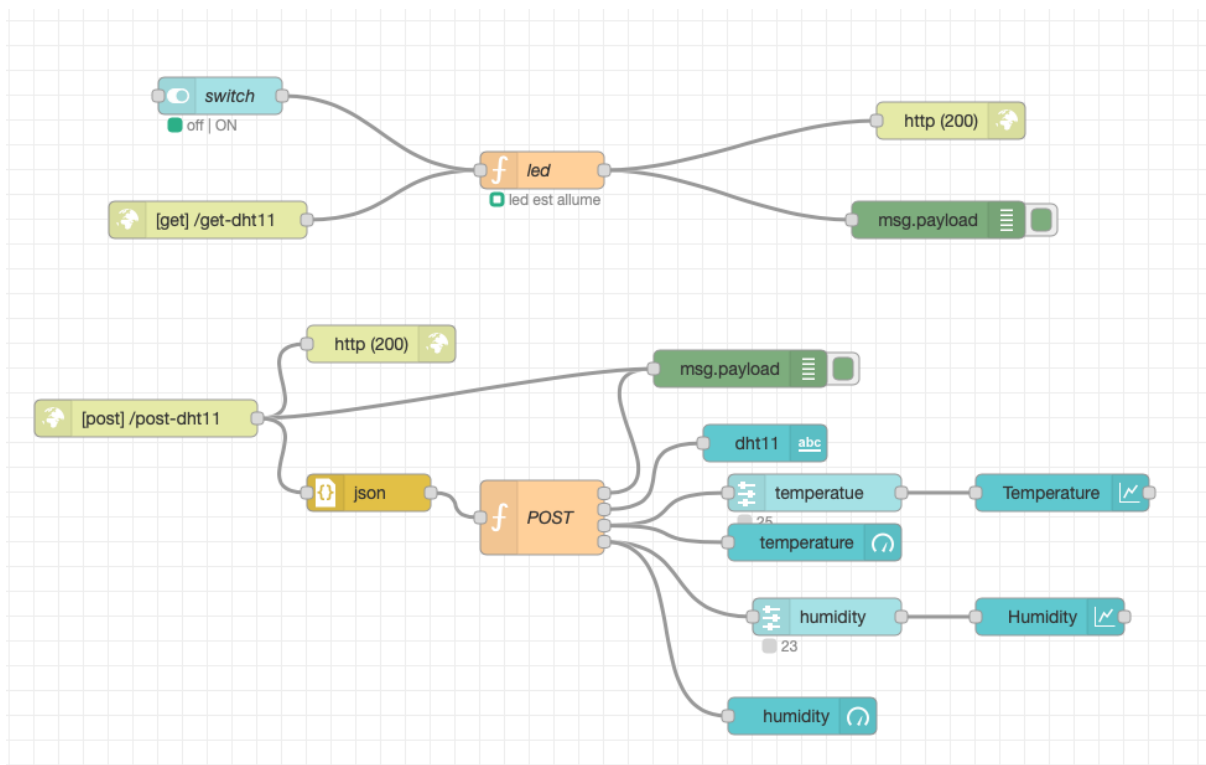
Intsallation du Node RED

Node RED est un logiciel permettant de gérer des flows d'événements, des suites de traitements à effectuer suite à la réception de messages ou événements. Pour installer Node RED, il est similaire pour les différents systèmes de l'ordinateur:

- **Mac/Linux:** tapez la commande `sudo npm install -g --unsafe-perm node-red` dans la **terminal**, une fois que l'installation terminée, tapez la commande `node-red` pour lancer la logiciel.
- **Windows:** Ouvrir **PowerShell**, tapez la commande `npm install -g --unsafe-perm node-red`, Puis placez dans le répertoire de l'installation : `c:\Users\<Utilisateur>\AppData\Roaming\npm\node_modules` en tapant `cd C:\Users\<Utilisateur>\AppData\Roaming\npm\node_modules\node-red`. A la fin, lancez le logiciel avec la commande `node red.js`

Interface du Node RED

Après lancement du Node RED, on peut entrer dans l'interface de Node RED. Tout d'abord ouvrir votre navigateur web et taper "**votre adresse IP + ":1880"**", par exemple `10.188.96.135:1880`. Puis importer le fichier **REST.json** pour obtenir le flow suivant. Ensuite, cliquez sur le bouton **Deploy** pour mettre à jour le flow.



Installation du Arduiono

Arduino est une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques matériellement libres sur lesquelles se trouve un microcontrôleur. Pour installer Arduino, ouvrir le site <https://www.arduino.cc/en/software> et choisir le système adapté pour installer.



Arduino IDE 1.8.16

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 [Get](#)

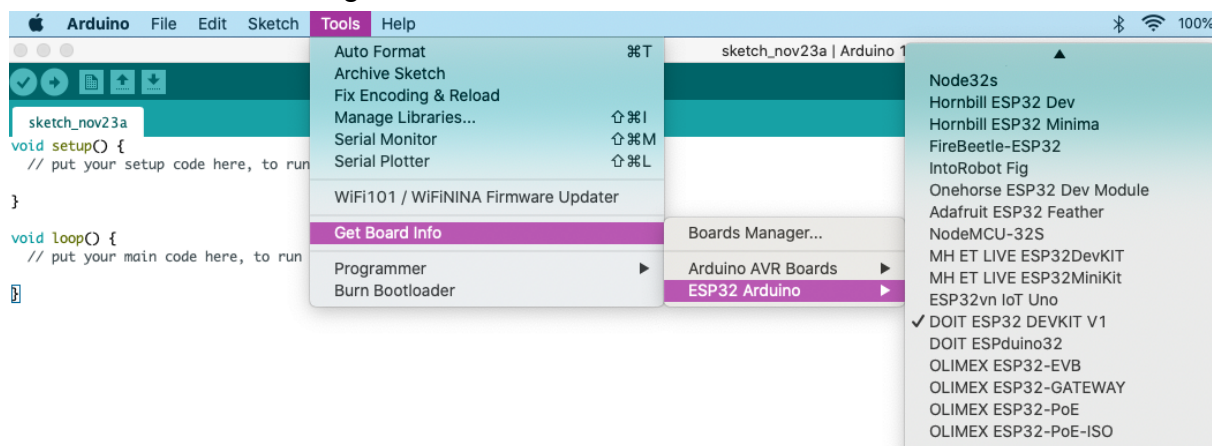
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#) [Checksums \(sha512\)](#)

Configuration du Arduino

Dans Arduino, il faut d'abord configurer le board du Arduino en **DOIT ESP32 DEVKIT V1** comme la figure suivante.



Pour Windows, il est simple de le faire, mais pour linux ou mac, il faut un peu plus de étapes. Tout d'abord, ouvrez la fenêtre des préférences de l'IDE Arduino. Allez dans **Arduino > Préférences**. Ensuite, entrez dans le champ "**Additional Boards Managers URLs**": https://dl.espressif.com/dl/package_esp32_index.json. Puis, cliquez sur le bouton **OK**. Après, choisir le board comme la figure au dessus.

Plus de détaille sur la site suivante:

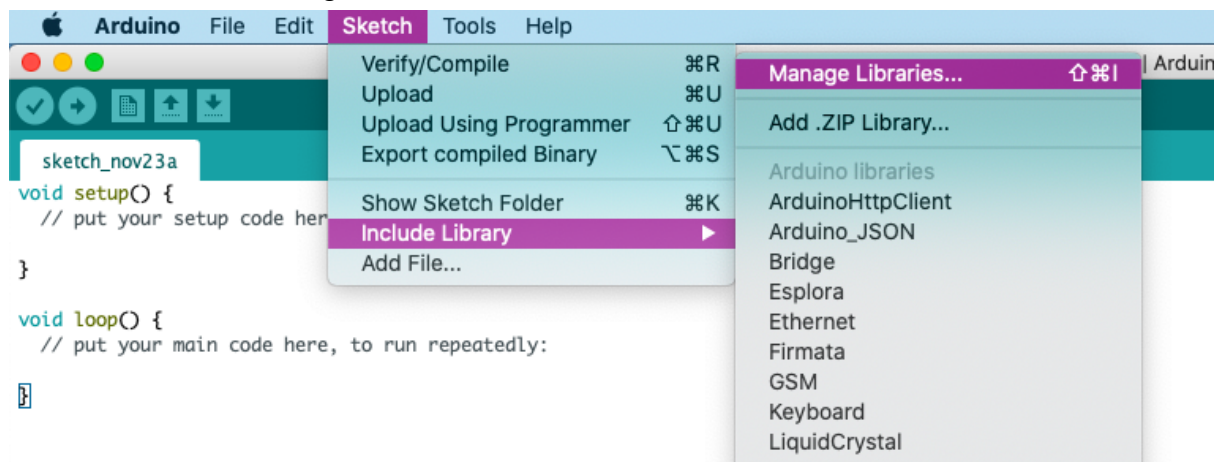
<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/>

Vérification du librairie

Avant d'exécuter le programme, il faut inclure les librairies montrées par la figure suivante et la librairies du ESP32.

```
#include <Arduino_JSON.h>
#include <WiFi.h>
#include <HttpClient.h>
#include <DHT.h>
```

Normalement, il est déjà inclus dans le dossier, s'il indique qu'il manque de librairie, vous pouvez l'installer. Tout d'abord, allez dans **Sketch > Include Library > Manage Libraries** comme la figure suivante.



Ensuite trouver les librairies nécessaire comme les figures suivantes:



WiFi

Built-In by **Arduino** Version **1.2.7** **INSTALLED**

Enables network connection (local and Internet) using the Arduino WiFi shield. For all Arduino boards. With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The shield can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.

[More info](#)

DHT sensor library

by **Adafruit** Version **1.4.3** **INSTALLED**

Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors

[More info](#)

ESP32 BLE Arduino

by **Neil Kolban** Version **1.0.1** **INSTALLED**

BLE functions for ESP32 This library provides an implementation Bluetooth Low Energy support for the ESP32 using the Arduino platform.

[More info](#)

Exécution du programme

Pour le code, il est dans le fichier **REST.ino**. Dans le code, il y a une partie que vous devez changer selon votre cas, ce sont le nom et le mot de passe du WiFi que vous allez utiliser et votre adresse IP pour les deux serverName montrés dans la figure suivante.

```
// spécifier le nom et le mot de passe de wifi
const char* ssid = "Mi";
const char* password = "huihui123";

// définir un nom de device
String device_label = "ESP32_DHT11_JY";

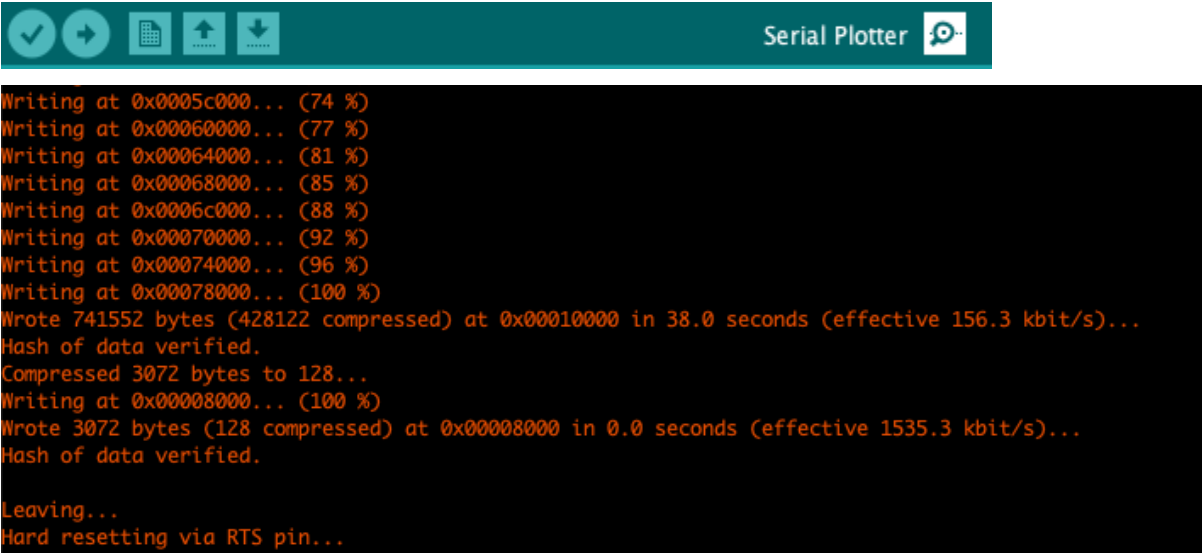
// Entrer votre adresse IP + 1880 + nom de URL
String serverNamePost = "http://192.168.43.20:1880/post-dht11";
String serverNameGet = "http://192.168.43.20:1880/get-dht11";
```

Après le changement du code, vous pouvez l'exécuter en cliquant sur le bouton **Upload**. En même temps, il faut appuyer sur le bouton **BOOT** du ESP32 jusqu'à qu'il affiche ces informations dans la console en bas.



```
Sketch uses 741442 bytes (56%) of program storage space. Maximum is 1310720 bytes.
Global variables use 38424 bytes (11%) of dynamic memory, leaving 289256 bytes for local variables. Maximum is 327680 bytes.
esptool.py v3.0-dev
Serial port /dev/cu.usbserial-0001
Connecting.....
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:6f:28:7b:76:14
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
```


Quand l'écriture atteint 100% montré comme la figure suivante, vous pouvez cliquer sur le bouton **Serial Monitor** pour ouvrir le terminal.



```
Writing at 0x0005c000... (74 %)
Writing at 0x00060000... (77 %)
Writing at 0x00064000... (81 %)
Writing at 0x00068000... (85 %)
Writing at 0x0006c000... (88 %)
Writing at 0x00070000... (92 %)
Writing at 0x00074000... (96 %)
Writing at 0x00078000... (100 %)
Wrote 741552 bytes (428122 compressed) at 0x00010000 in 38.0 seconds (effective 156.3 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1535.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

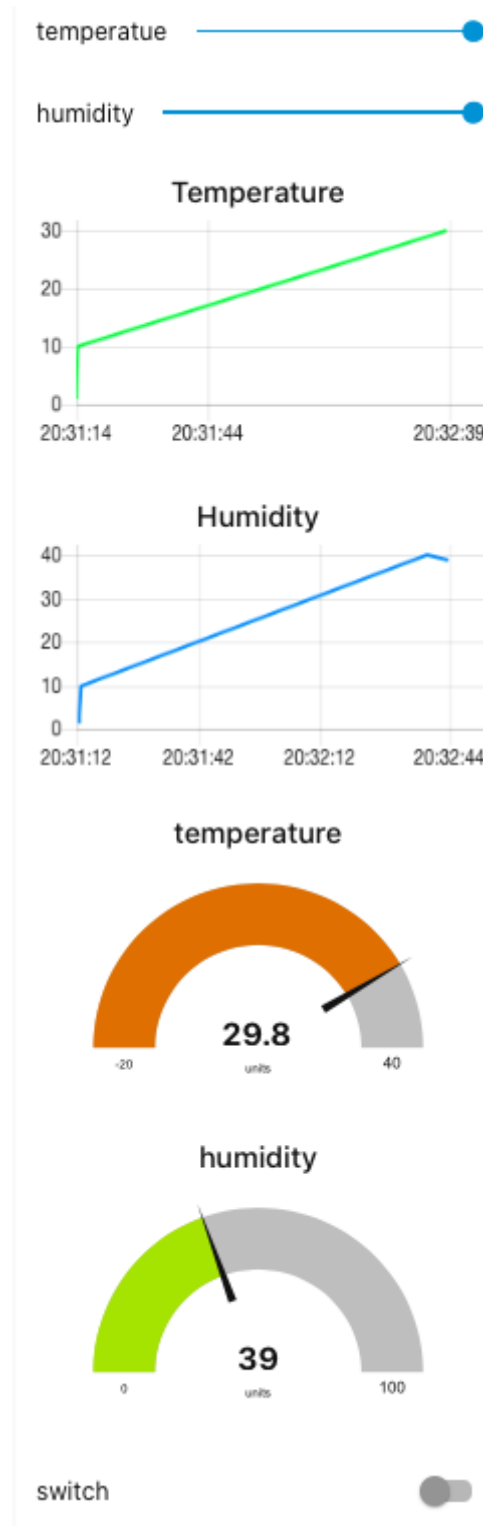
Quand on a bien exécuté le programme, il va afficher ces informations dans le terminal:

```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
Humidity:
28.00
Temperature:
26.30
{"api_key":"ESP32_DHT11_JY","sensor":["DHT11","DHT11"],"humidity":"28.00","temperature":"26.30"}
HTTP Response code for get: 200
{"switch":0}
JSON object = {"switch":0}
"switch" = 0
```

Visualisation des données

Pour visualiser les données, ouvrez votre navigateur, entrez “**votre adresse IP + “:1880/ui”**”, par exemple **10.188.96.135:1880/ui**. Vous allez voir un databoard comme la figure en dessous.



Le **switch** permet de contrôler la **LED interne** du ESP32 en pin 2.

Remarque:

Pour bien réaliser la communication entre Node RED et ESP32, il vaut mieux se connecter sur le même réseau. Sinon, il va peut-être y avoir des problèmes.