IUT
CERGY-PONTOISE

CY IUT CERGY

GE&II Neuville
BUT3-S5 ESE
R5-10 ESE
Système Embarqué
Internet Des Objets version UnoR4 WiFi

A.DeCarvalho
Octobre 2024

IUT
CERGY-PONTOISE

Internet des objets (IOT)

**L'internet des objets (Internet of things ou IOT)
est l'interconnexion entre l'internet et des objets, des lieux et des environnements physiques.**

- **Objets qui se connectent à l'internet**
- **Machines qui sont en capacité de communiquer entres elles et de traiter des systèmes d'informations par elles-mêmes.**
- **Appareils tels que les téléphones intelligents ou des tablettes.**

IUT CERGY-PONTOISE

Internet des objets (IOT)

L’internet des objets se retrouve dans différents environnements:

En logistique pour le traçage des produits et ou la gestion des stokes.

Développement durable, pour mesurer la qualité de l’air, en météorologie ou la détection des risques environnementaux

Domotique dans de nombreux appareils d’électroménager.

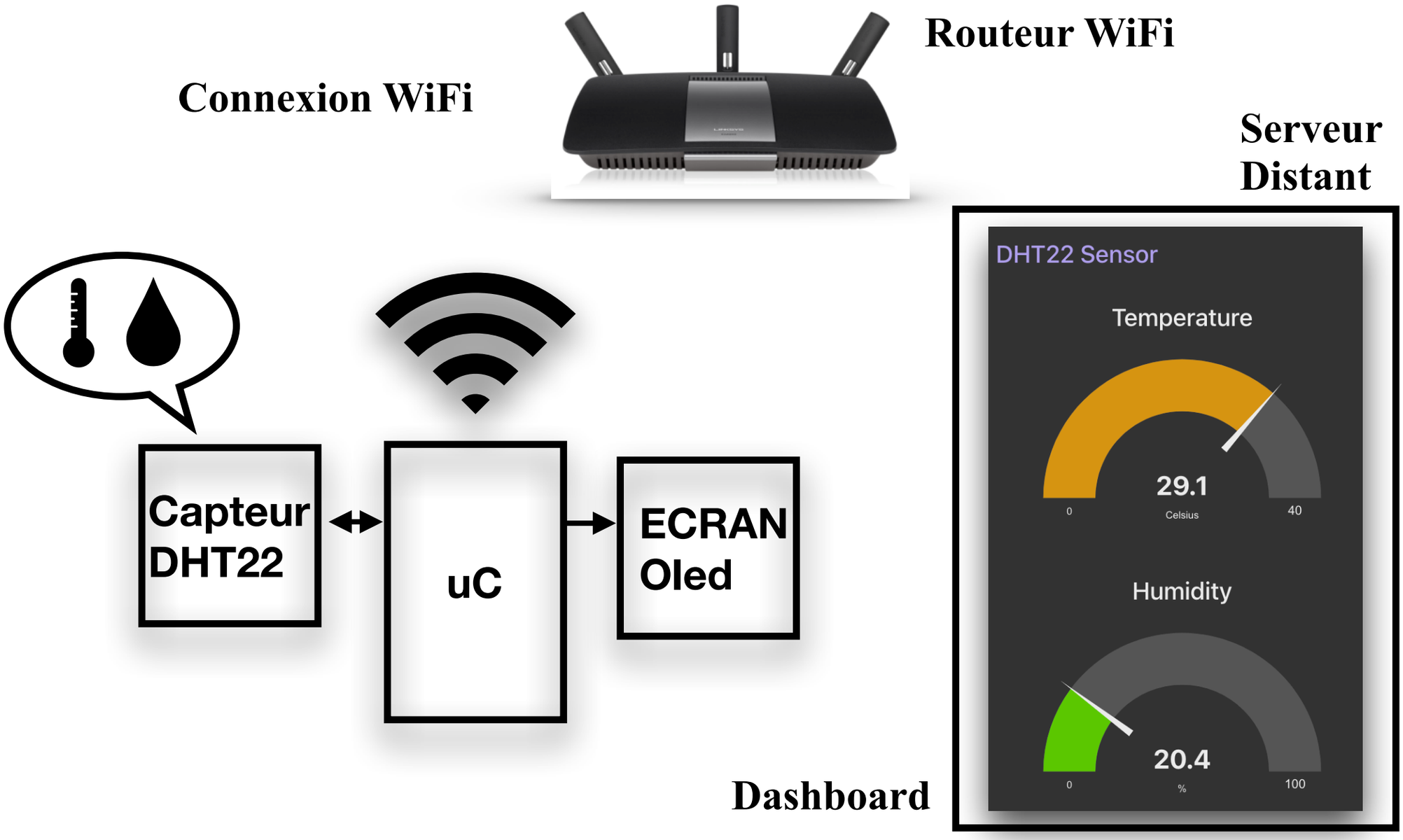
Santé, par exemple les montres connectées et ou les moniteurs dans les hôpitaux.

IUT CERGY-PONTOISE

But de ce projet

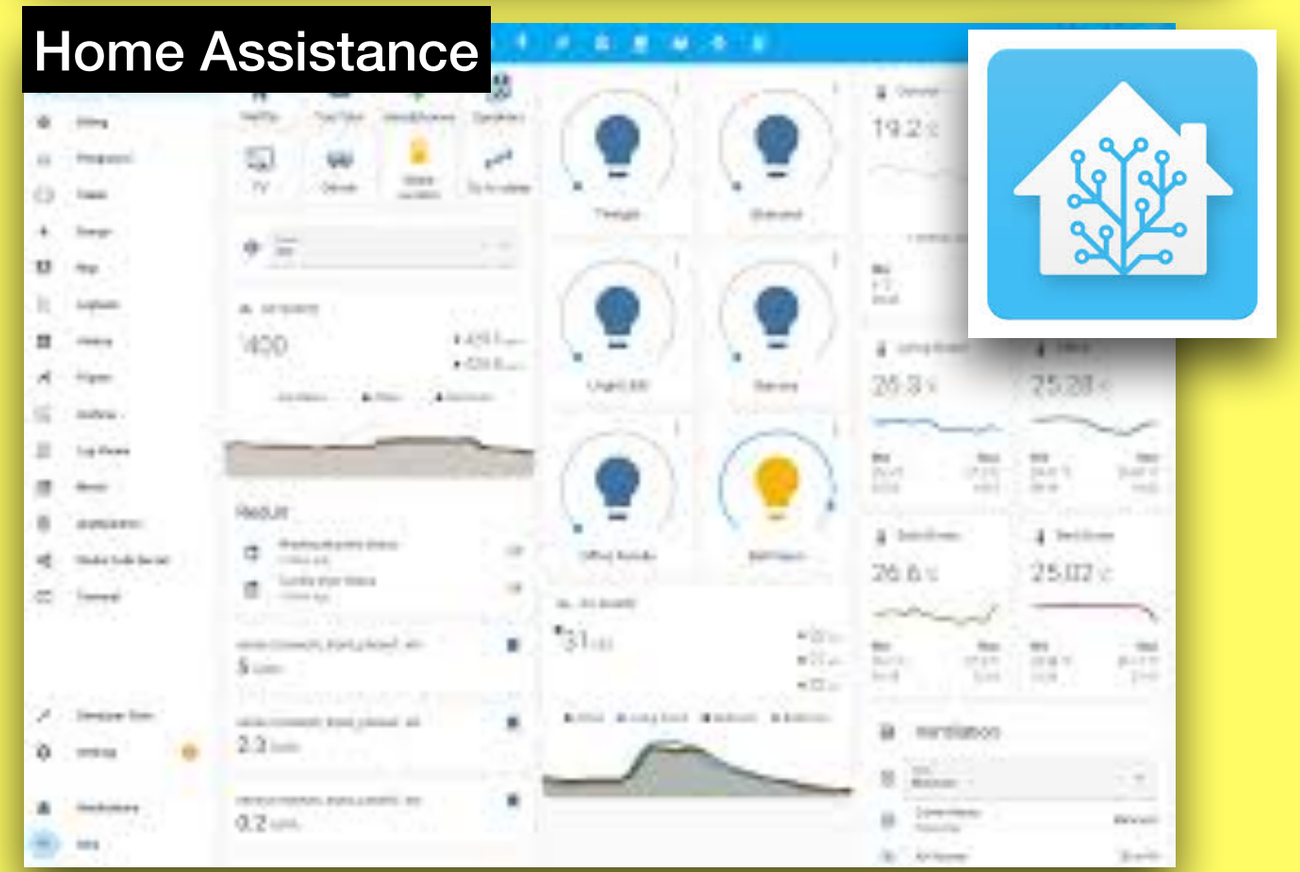
Réaliser un capteur intelligent, qui fournira à un serveur distant, la température et le taux d'hygrométrie à l'extérieur ou à l'intérieur d'une habitation.

Vous disposez de 6 séances de 4h pour mener à bien le projet.



Exemples Industriels

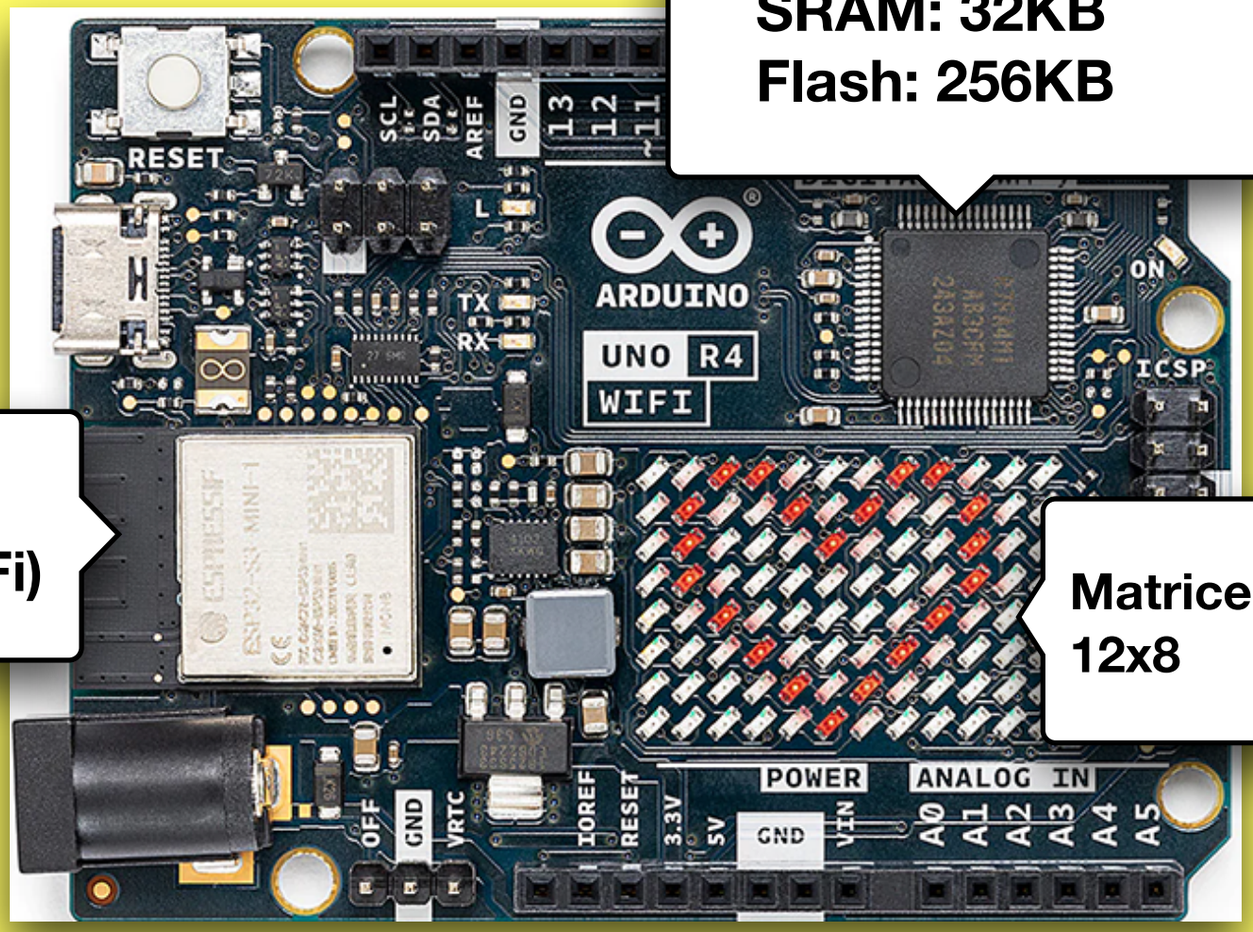
Apple HomeKit



La carte UNO R4 WIFI possède deux micro contrôleurs

RA4M1-Renasas
ARM Cortex M4

Horloge:48MHz
SRAM: 32KB
Flash: 256KB



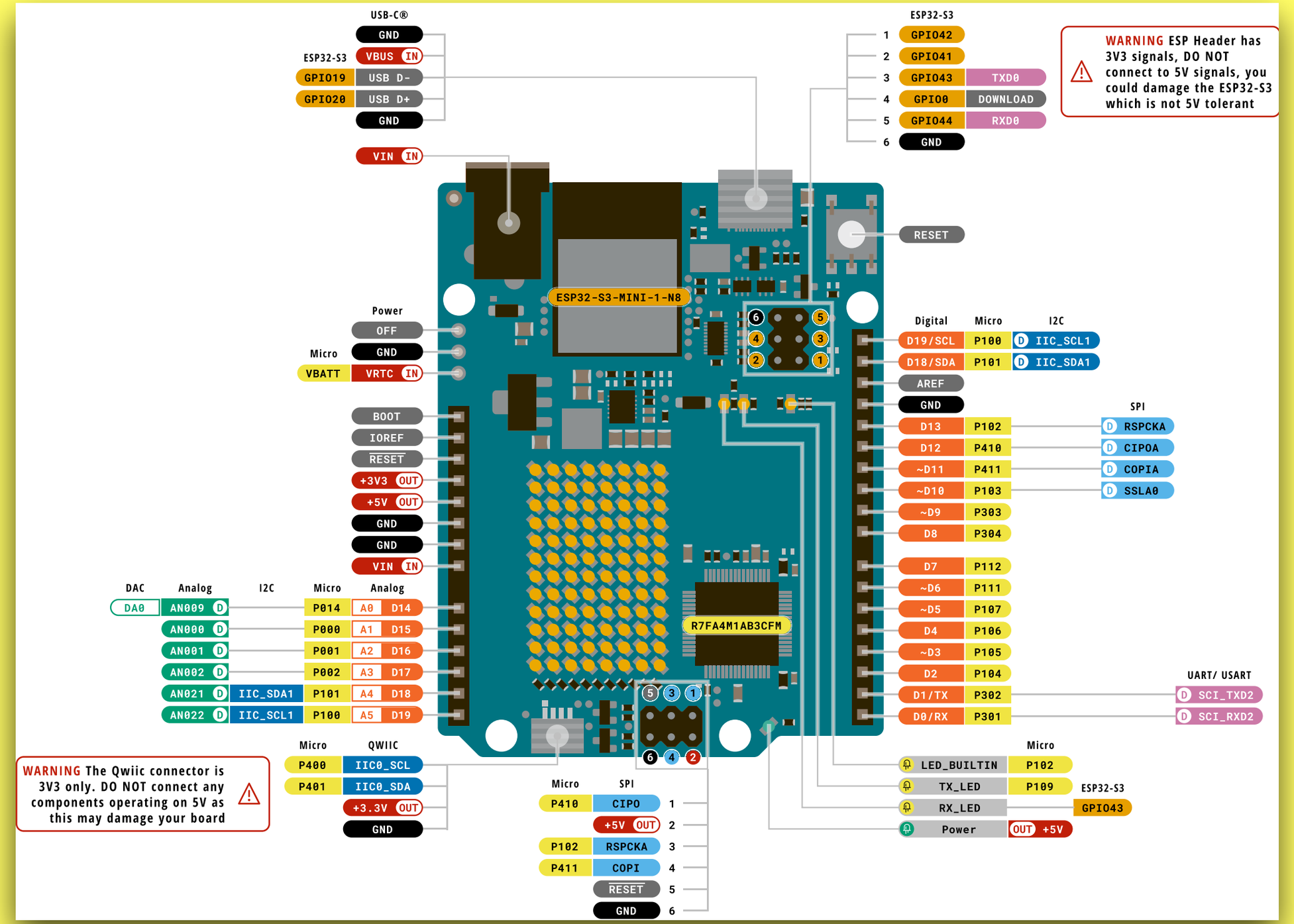
https://docs.arduino.cc/hardware/uno-r4-wifi/

IUT
CERGY-PONTOISE

Micro Controleur

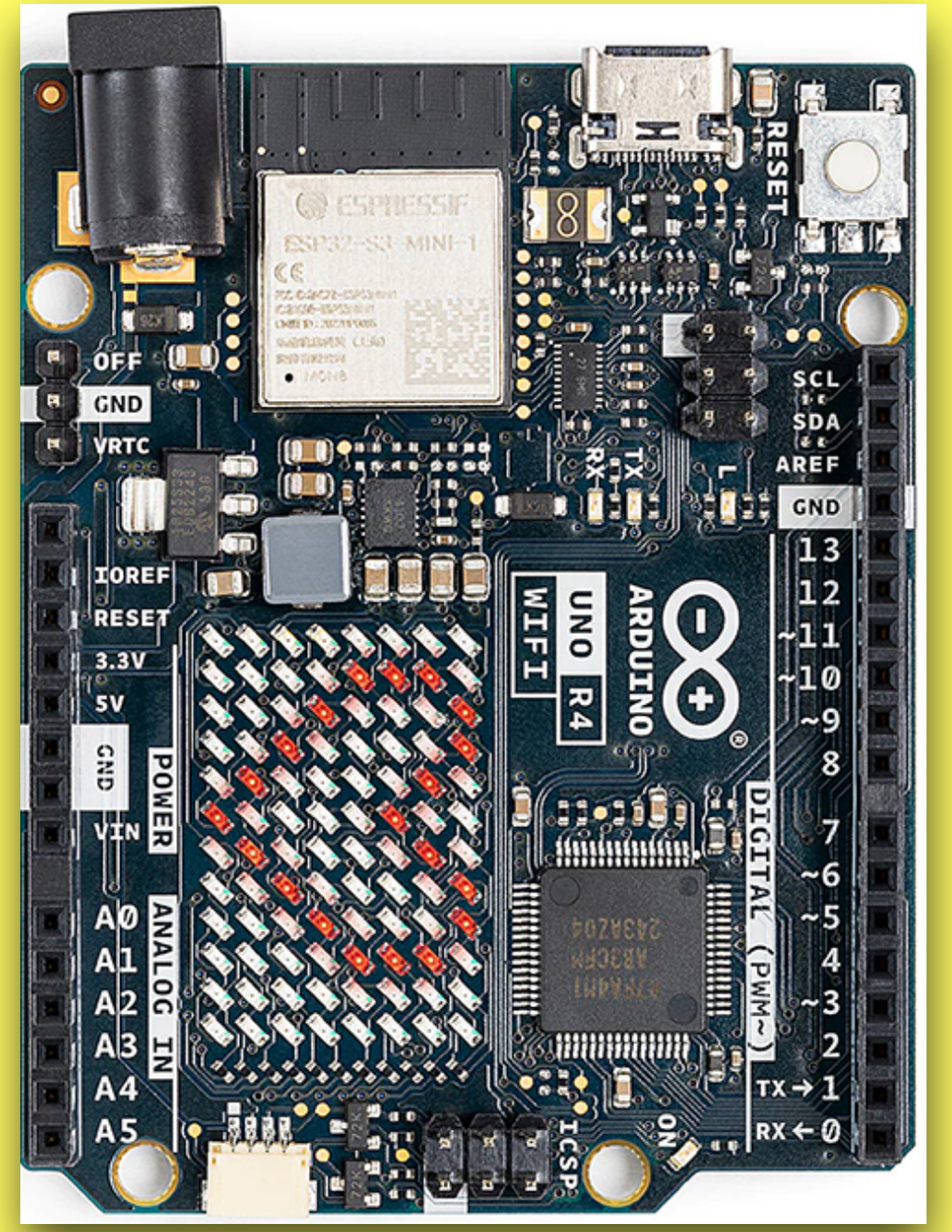
UNO R4 WIFI

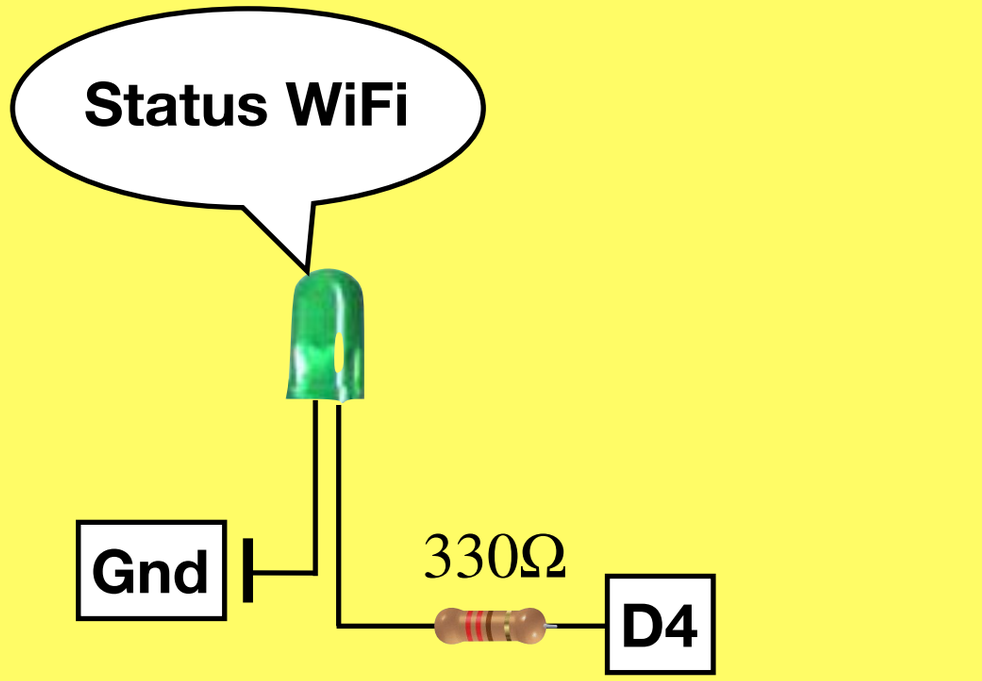
Internet Des Objets
R5-10



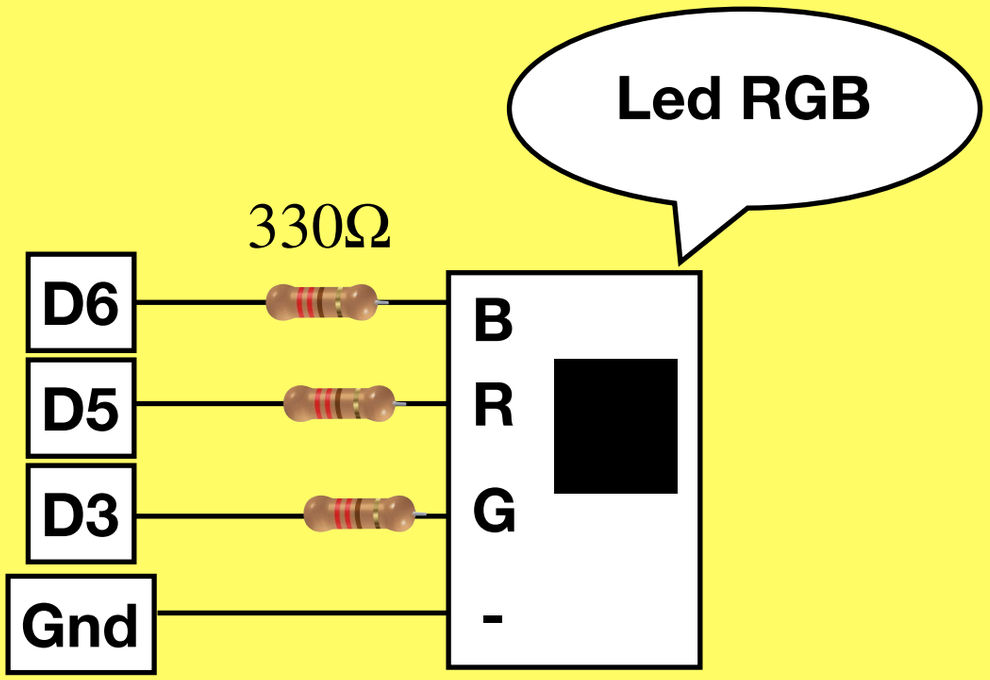
Micro Controleur

Montage Leds

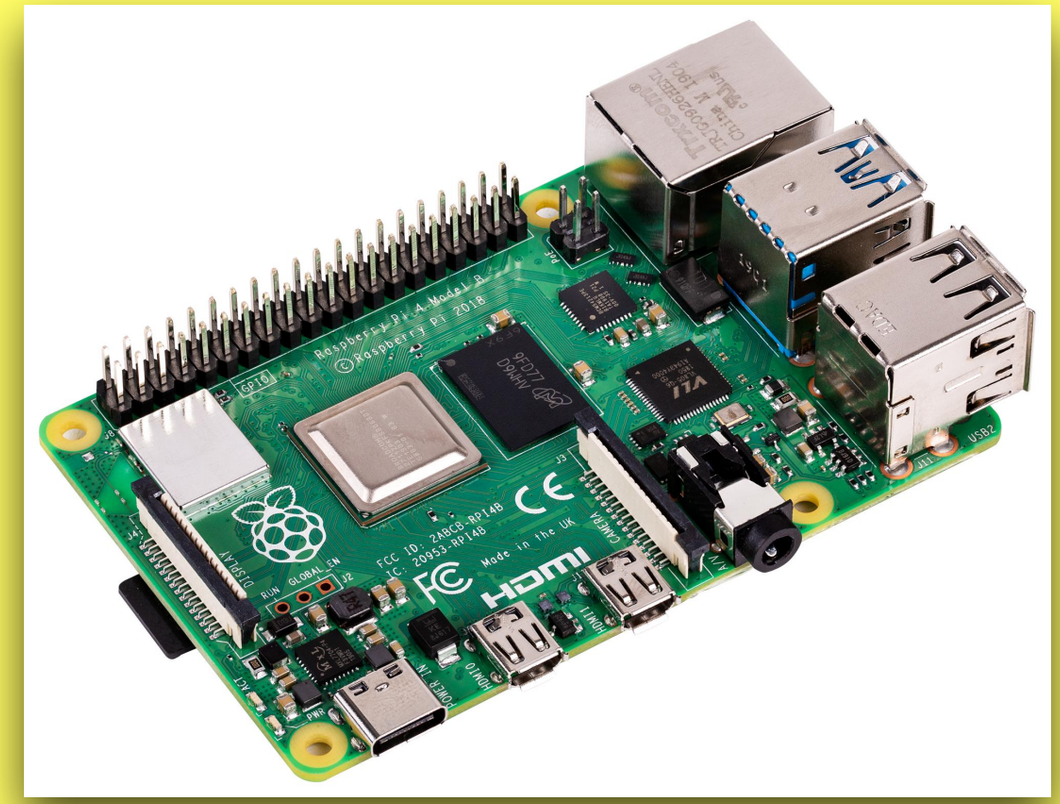




analogWrite( )



Serveur Domotique



Raspberry Pi 4

Le serveur domotique sera implémenté dans une carte Raspberry Pi 4 ou 400

Raspberry Pi 400

Commandes shell

pwd

Print Working Directory
Affiche le chemin absolu du dossier ou est l'utilisateur.

cd

Change Directory
Change de répertoire.

mkdir

Make Directory
Créer un répertoire , le nom doit suivre la commande.

ls

List
Permet d'afficher le contenu d'un répertoire.

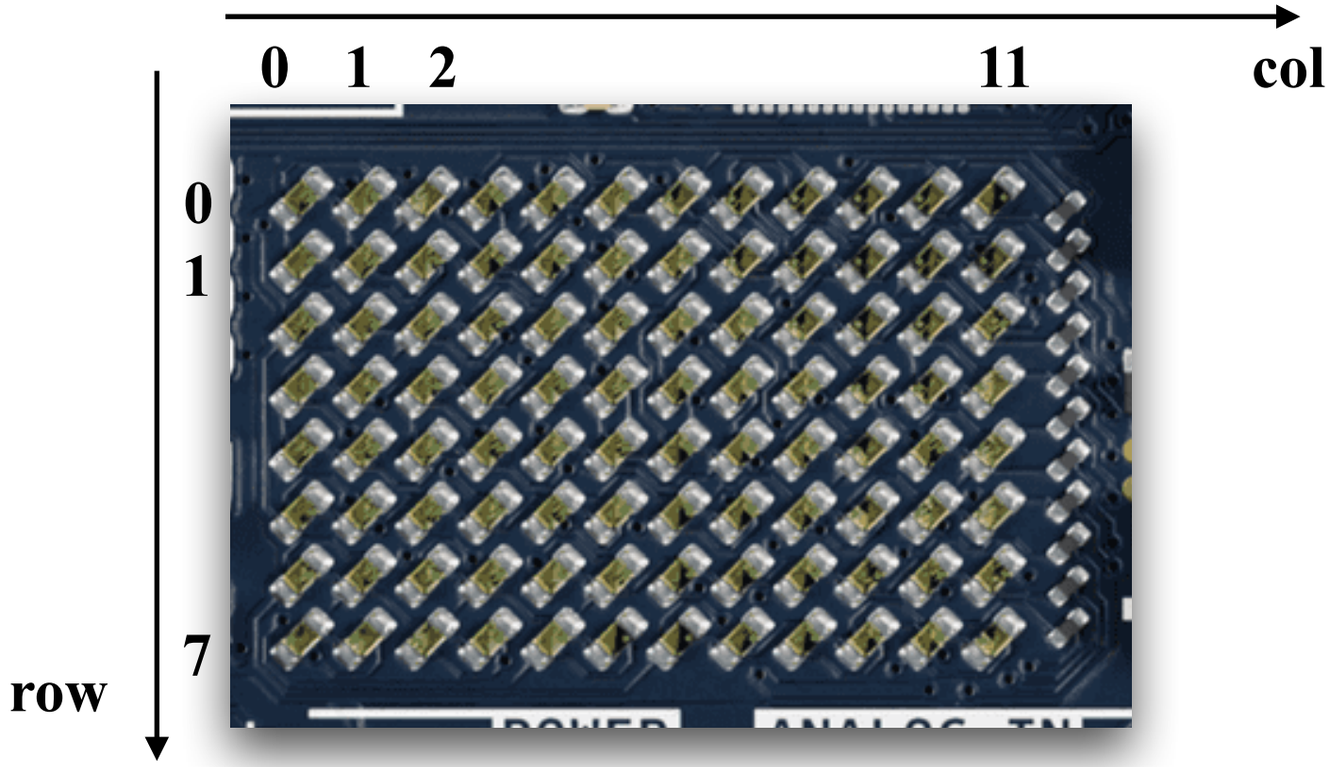
date

Date
Affiche ou change la date du jour.

sudo date -s "2023-06-30 9:15"

Exemple pour fixer la date du jour

UnoR4 matrix




```
#include "Arduino_LED_Matrix.h"

ArduinoLEDMatrix matrix;

void setup() {
  Serial.begin(115200);
  matrix.begin();
}
```

UnoR4 matrix


```
uint8_t frame[8][12] = {
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

void leftEye(){
  //Left eye
  frame[1][3] = 1;
  frame[1][4] = 1;
  frame[2][3] = 1;
  frame[2][4] = 1;
}

void wink(){
  //Wink with the left eye
  frame[1][3] = 0;
  frame[1][4] = 0;
  frame[2][3] = 1;
  frame[2][4] = 1;
}

void rightEye(){
  //Right eye
  frame[1][8] = 1;
  frame[1][9] = 1;
  frame[2][8] = 1;
  frame[2][9] = 1;
}
```

```
void mouth(){
  //Mouth
  frame[5][3] = 1;
  frame[5][9] = 1;
  frame[6][3] = 1;
  frame[6][4] = 1;
  frame[6][5] = 1;
  frame[6][6] = 1;
  frame[6][7] = 1;
  frame[6][8] = 1;
  frame[6][9] = 1;
}

//*****************************
void loop(){
leftEye();
rightEye();
mouth();

matrix.renderBitmap(frame, 8, 12);

delay(1000);
wink();

matrix.renderBitmap(frame, 8, 12);
delay(1000);
}
```

Scrolling text

```
#include "ArduinoGraphics.h"
#include "Arduino_LED_Matrix.h"

ArduinoLEDMatrix matrix;

void setup() {
  Serial.begin(115200);
  matrix.begin();

  matrix.beginDraw();
  matrix.stroke(0xFFFFFFFF);
  // add some static text
  const char text[] = "UNO r4";
  matrix.textFont(Font_4x6);
  matrix.beginText(0, 1, 0xFFFFFF);
  matrix.println(text);
  matrix.endText();

  matrix.endDraw();

  delay(2000);
}
```

```
//******************
void loop() {

  // Make it scroll!
  matrix.beginDraw();

  matrix.stroke(0xFFFFFFFF);
  matrix.textScrollSpeed(50);

  // add the text
  const char text[] = "    Hello World!    ";
  matrix.textFont(Font_5x7);
  matrix.beginText(0, 1, 0xFFFFFF);
  matrix.println(text);
  matrix.endText(SCROLL_LEFT);

  matrix.endDraw();
}
```

DHT22 Sensor

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

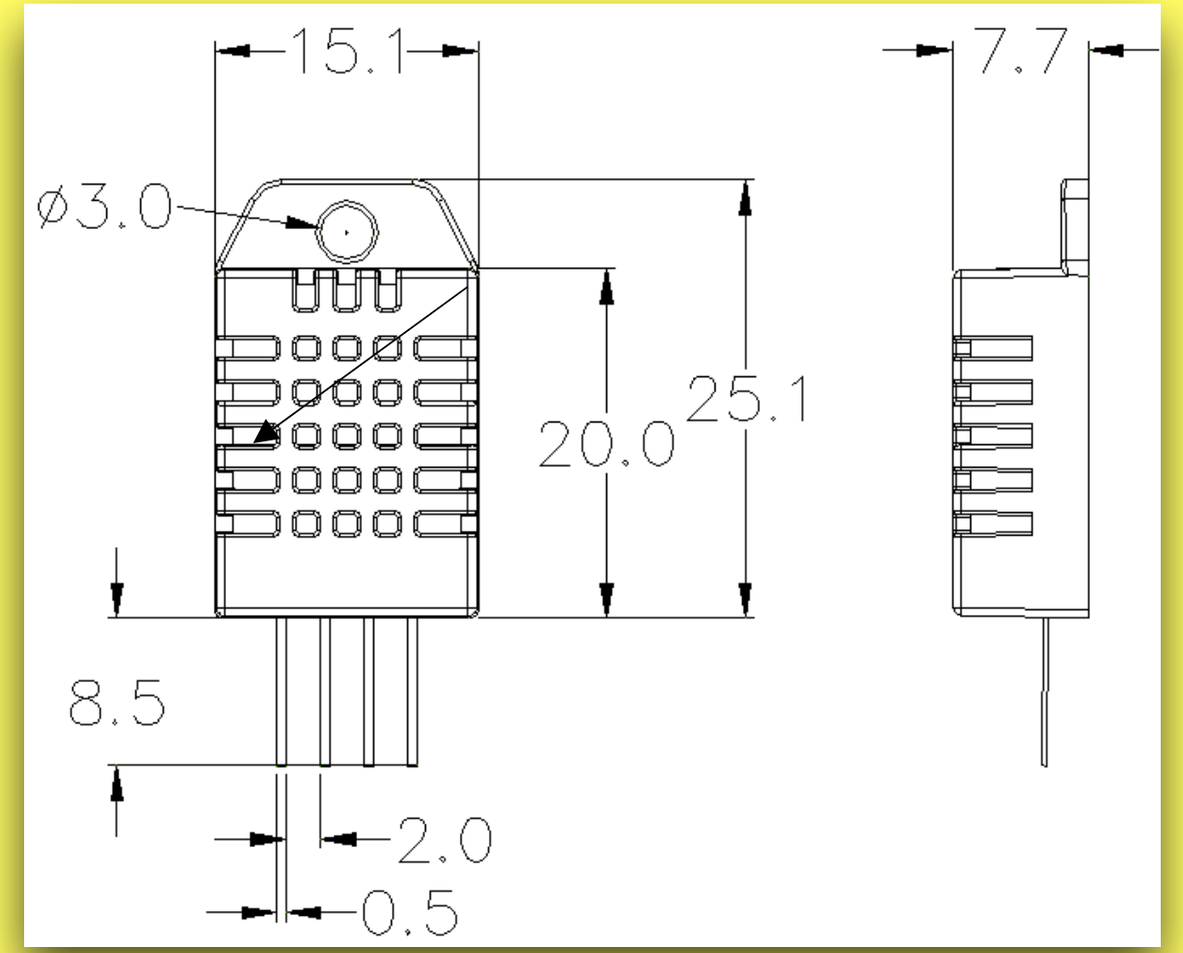
Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

Technical Specification

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

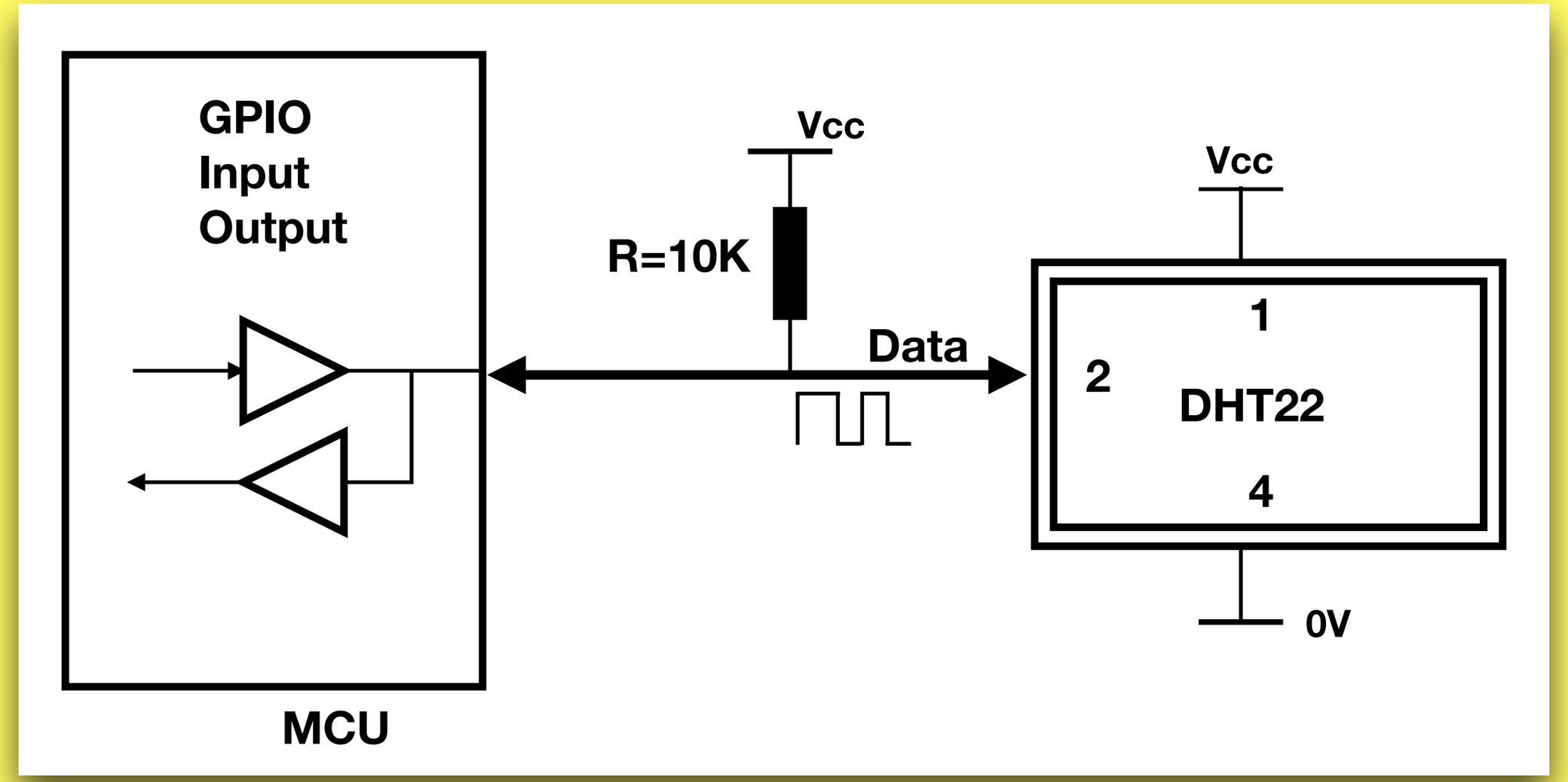
Technical Specification



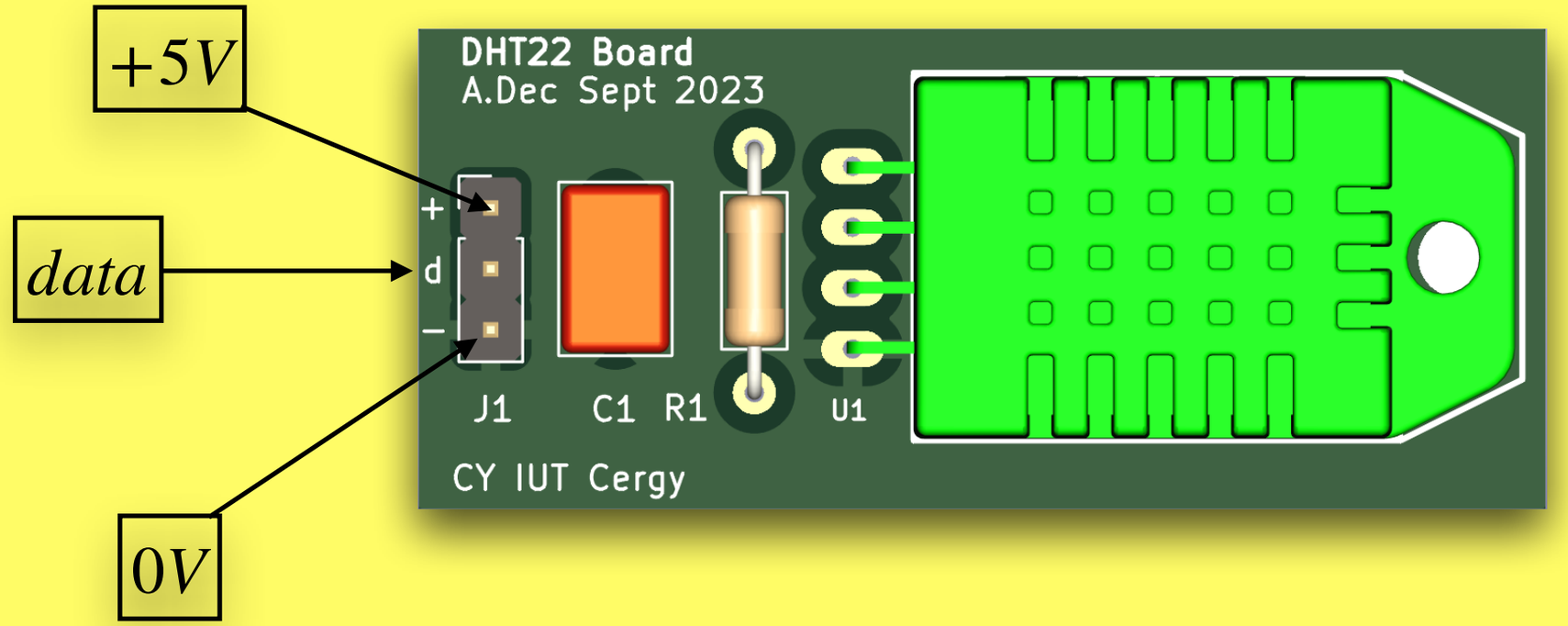
Pin sequence number: 1 2 3 4 (from left to right direction).

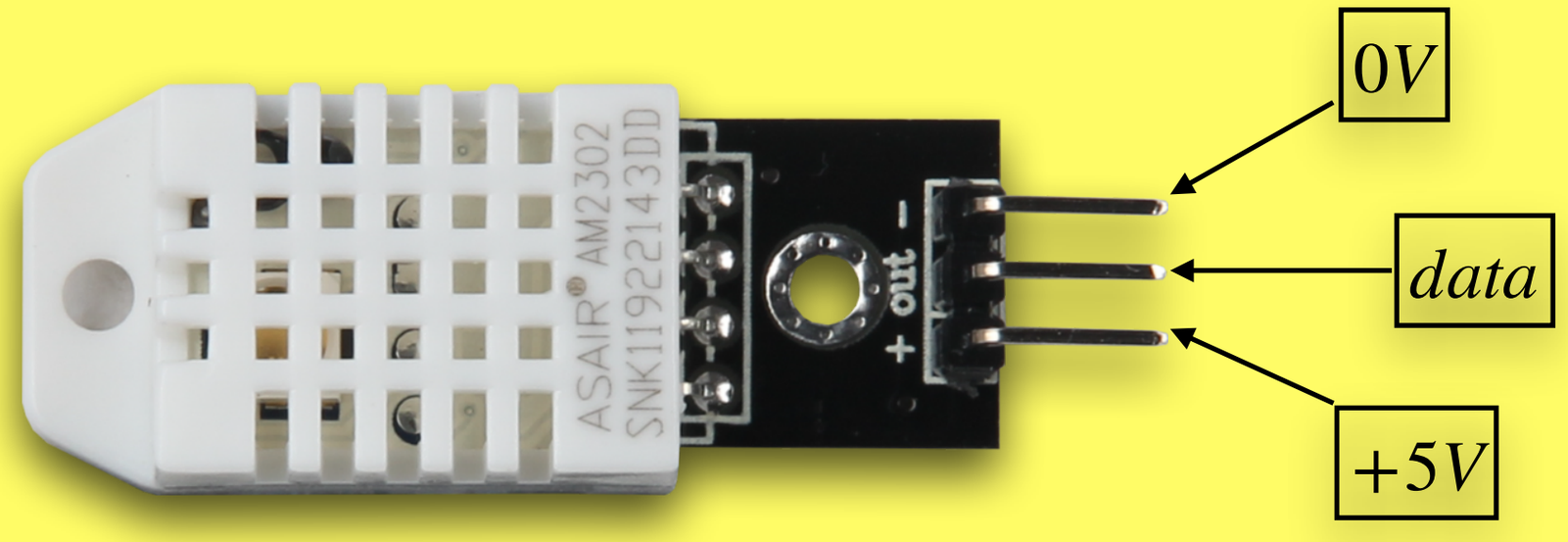
Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

Electrical Connection



Mini Plaquette DHT22





Operating Specifications

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

Single-bus data is used for communication between MCU and DHT22, it costs 5mS for single time communication.

Data is comprised of integral and decimal part, the following is the formula for data.

DHT22 send out higher data bit firstly!

DATA=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data+8 bit check-sum

If the data transmission is right, check-sum should be the last 8 bit of "8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data".

Operating Specifications

Exemple: MCU has received 40 bits data from DHT22

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16 bits RH data 16 bits Temperature data Check sum

0000 0010 1000 1100 ⟶ 652 (dec)

RH=652/10=65.2%

0000 0001 0101 1111 ⟶ 351 (dec)

Temperature=351/10=35.1 deg Celsius

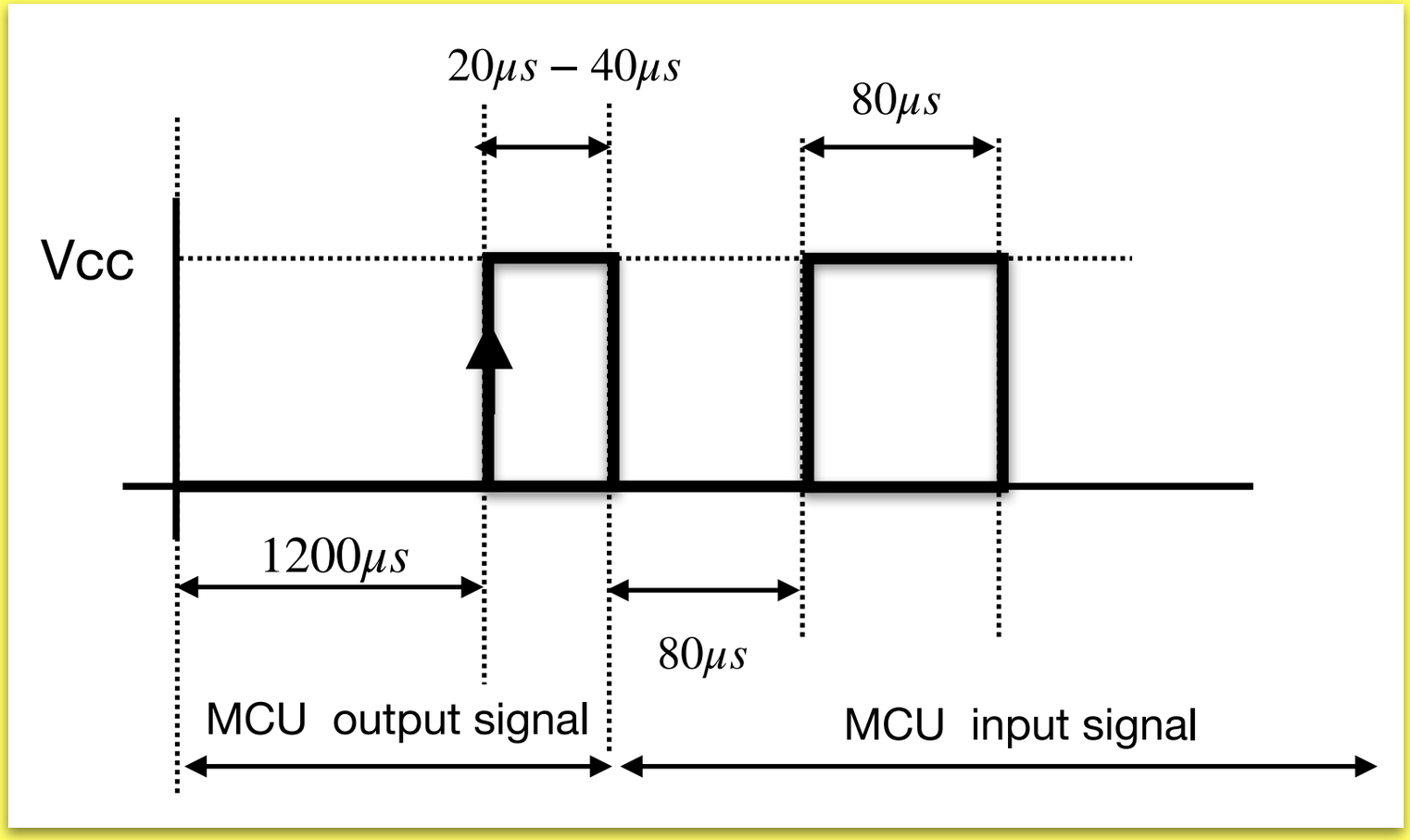
sum= 0000 0010 + 1000 1100 + 0000 0001+ 0101 1111

sum=1110 1110

Check sum= 1110 1110

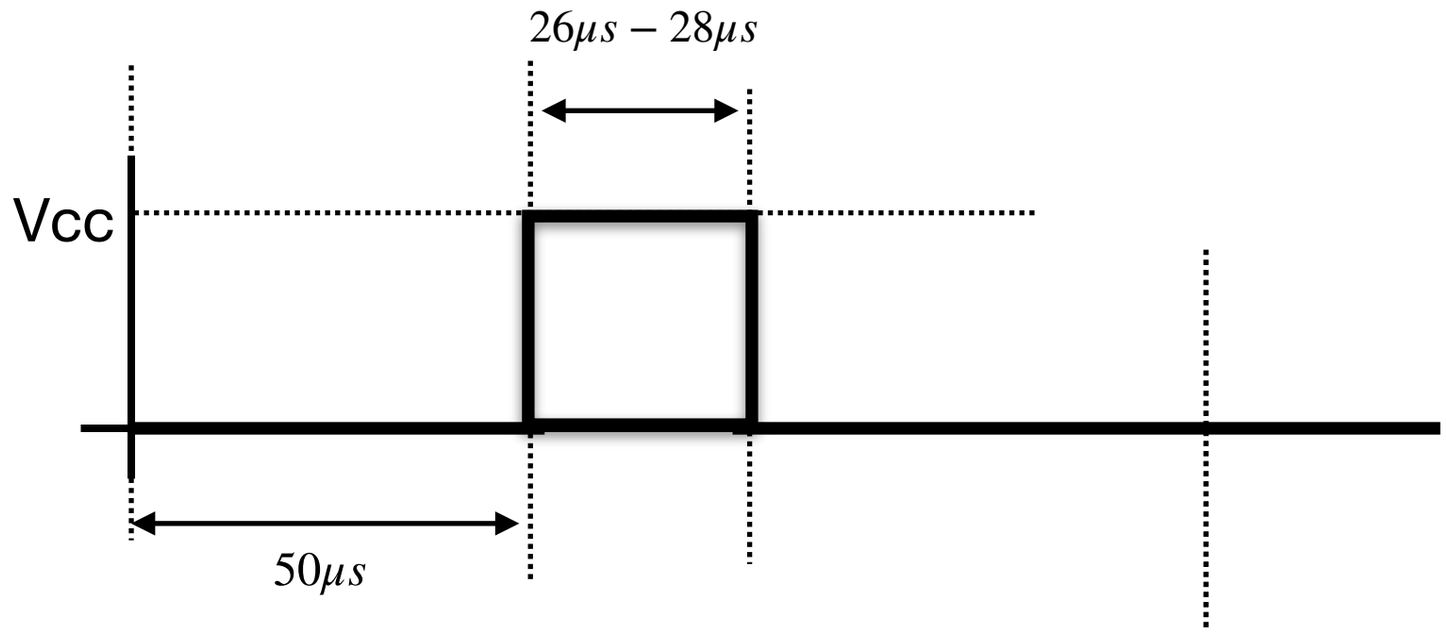
Start Communication

Data-bus's free status is high voltage level. When communication between MCU and DHT22 begin, program of MCU will transform data-bus's voltage level from high to low level and this process must beyond at least 1ms to ensure DHT22 could detect MCU's signal, then MCU will wait 20-40us for DHT22's response.

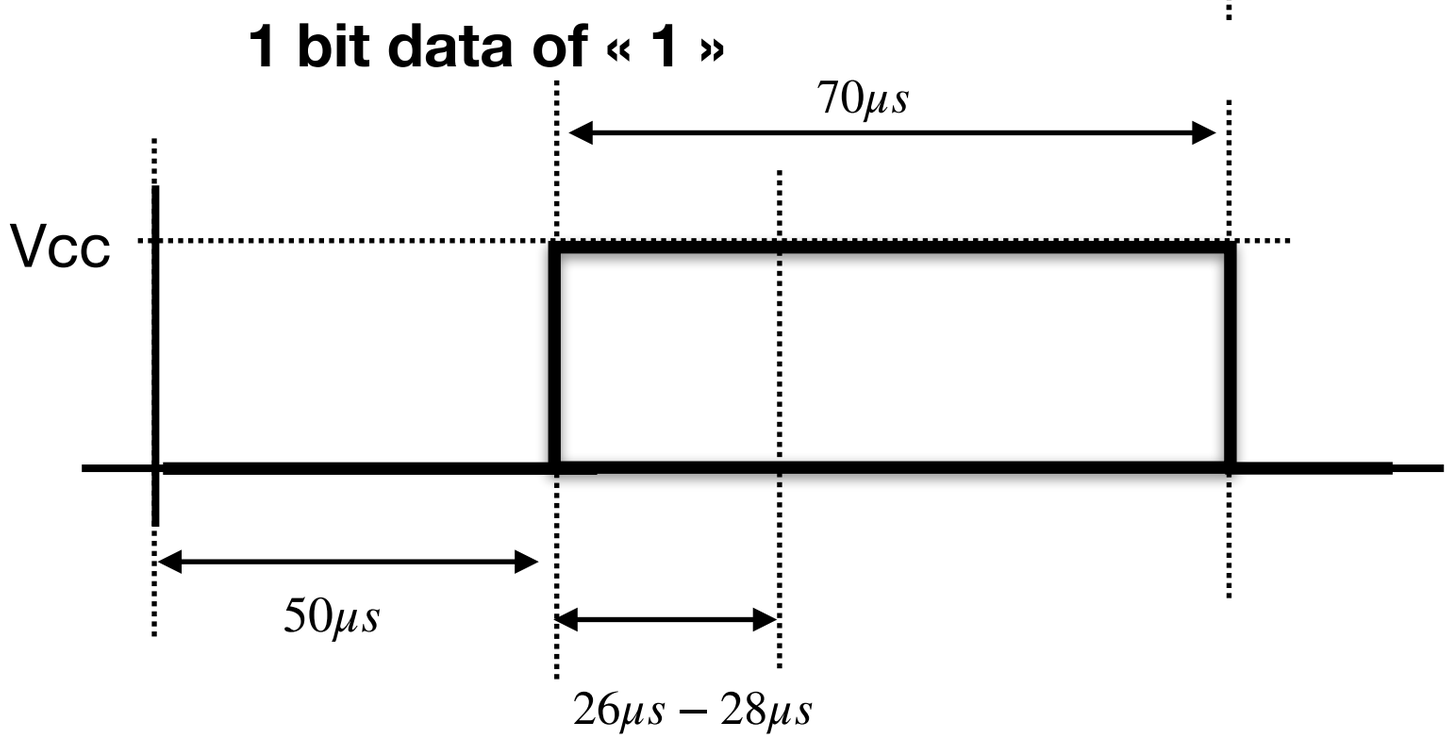


Technical Specification

1 bit data of « 0 »



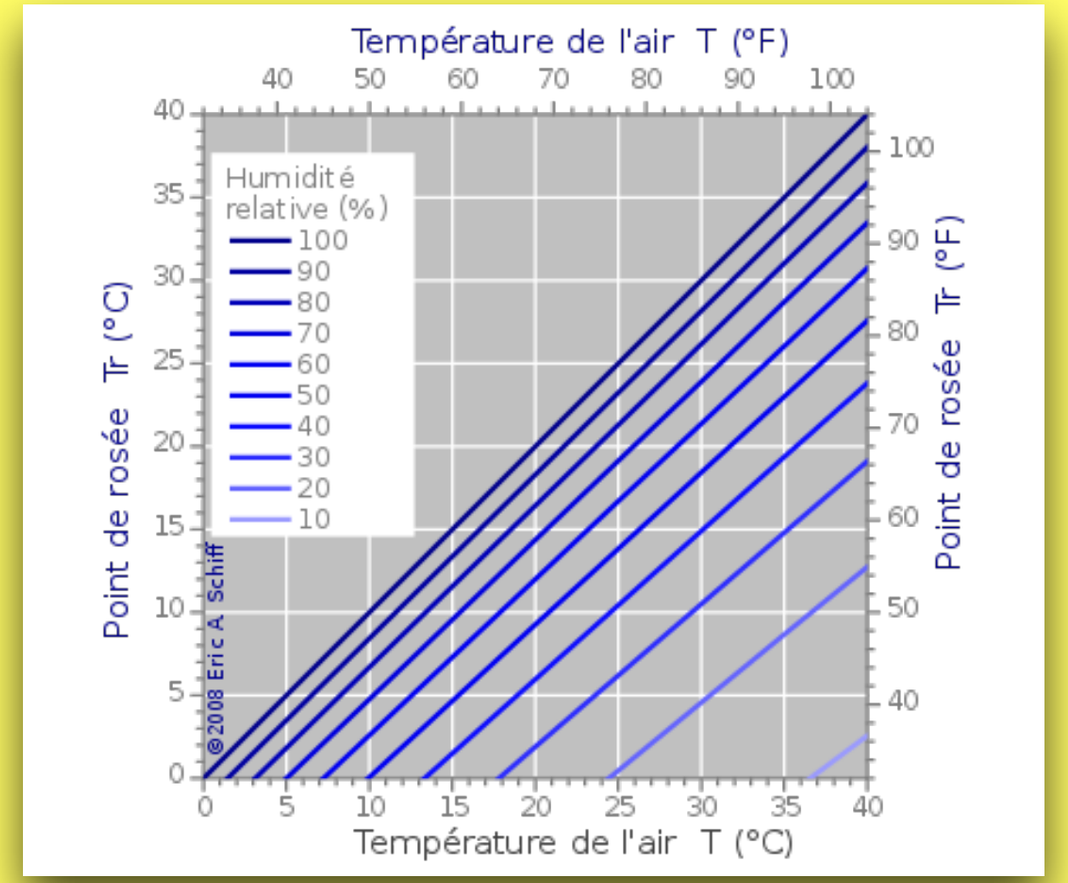
1 bit data of « 1 »



Point de Rosée

WIKIPEDIA

Le point de rosée ou température de rosée est la température sous laquelle de la rosée de dépose naturellement. En dessous de cette température qui dépend de la pression et de l'humidité constante, la vapeur d'eau contenue dans l'air se condense sur les surfaces, par effet de saturation.



Courbe de dépendance du point de rosée, par rapport à la température de l'air, pour différents niveau d'humidité.

Point de Rosée

Formule de Heinrich Gustav Magnus-Tetens

$$T_R = \frac{b \times \Psi(T, Rh)}{a - \Psi(T, Rh)}$$

$$\Psi = \frac{a \times T}{b + T} + ln(\frac{Rh}{100})$$

$$a = 17{,}27$$

$$b = 237{,}7$$

T_R Température de rosée [°C]

T Température ambiante [°C]

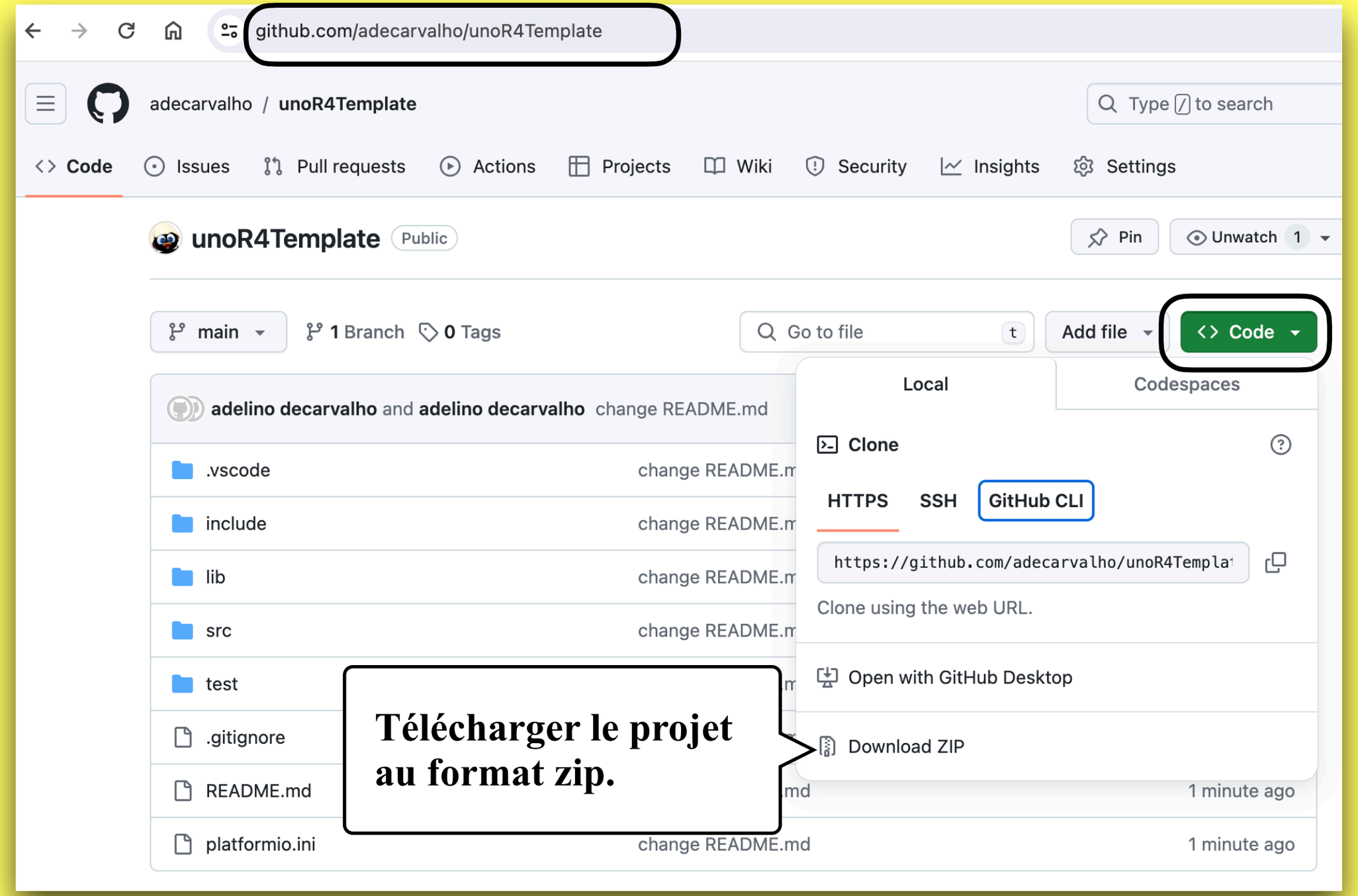
Rh Humidité relative [0,100%]

Domaine de validité de la relation

$0 < T < 60$ $\quad$ $0 < Rh < 100\,\%$

Visual Studio Code

Le projet sera réalisé sous l'IDE Visual Studio Code, avec l'extension PlatformIO.
Télécharger à cette adresse un projet unoR4Template.

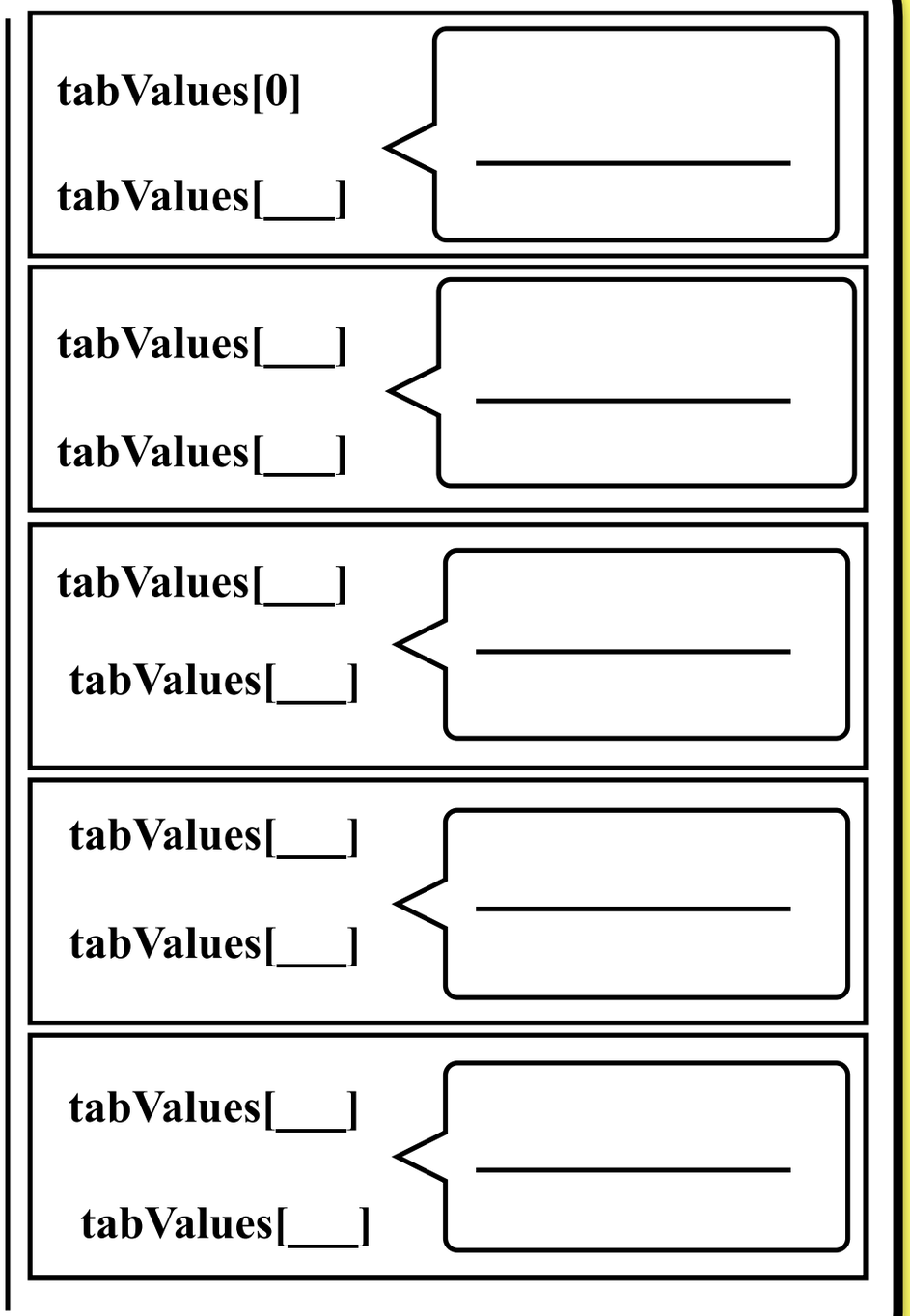


Programme en C

Pour le DHT22, quelques constantes et variables nécessaires:

```
enum DHT_STATUS
{
    DHT_OK = 0,
    DHT_ERROR_RESPONSE,
    DHT_ERROR_CHECKSUM,
    DHT_ERROR_TIMEEOUT
};


// const
const uint8_t DHT_PIN = ????;

const uint8_t NB = 40;
```

```
// var
uint8_t tabValues[NB] = {0};

float theTemperature = 0;

float theHumidity = 0;

float theDewPoint = 0;
```



Programme en C

Réaliser une fonction en C/C++ afin de remplir le tableau “tabValues” avec des zéros.

```
void DHT_tabReset( )
{
    //todo

}
```

Programme en C

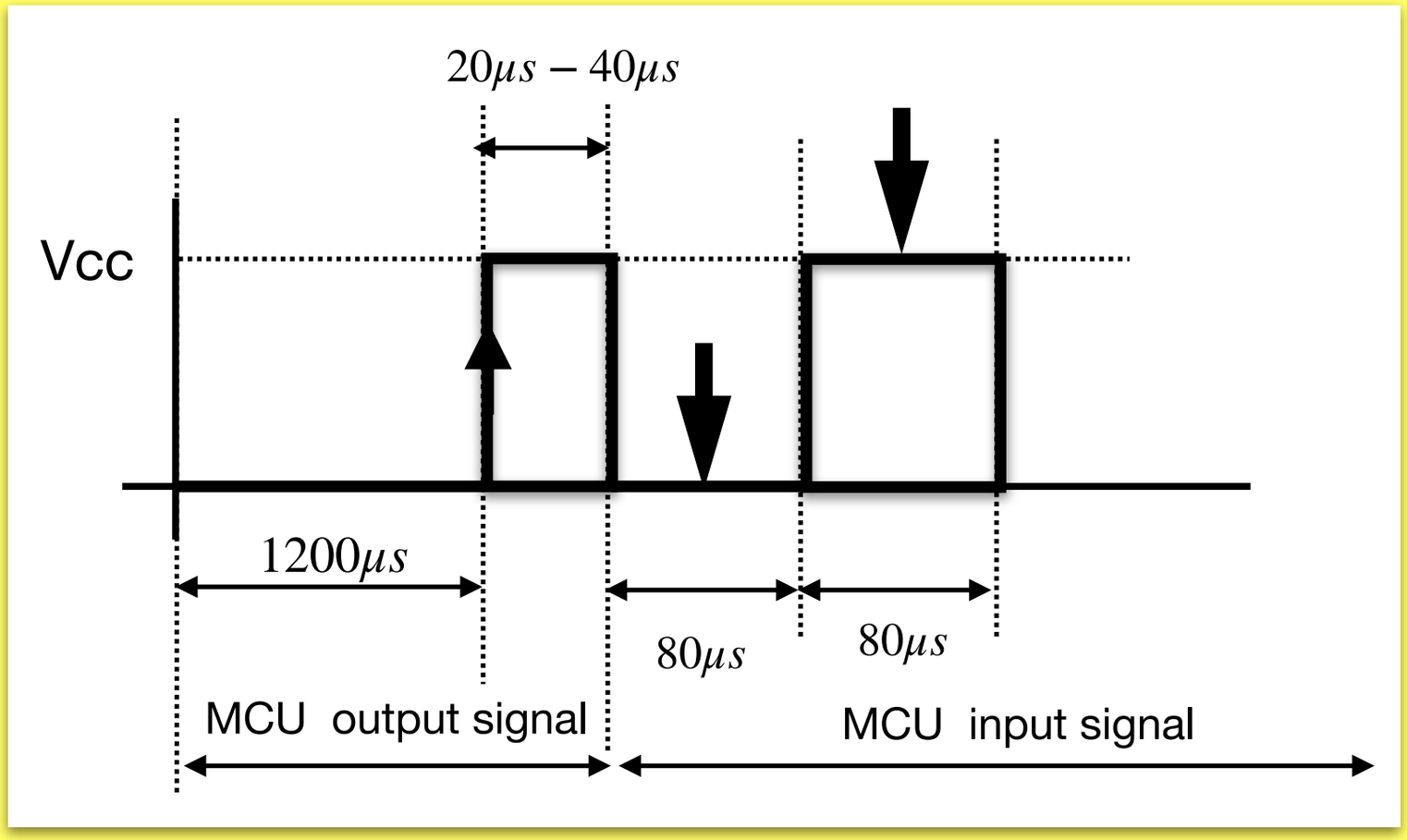
Réaliser une fonction en C/C++ afin d'initialiser le capteur dans le "setup":

```
void DHT_begin( )
{
      // Broche du capteur en entrée

      //Initialiser le tableau tabValues

}
```

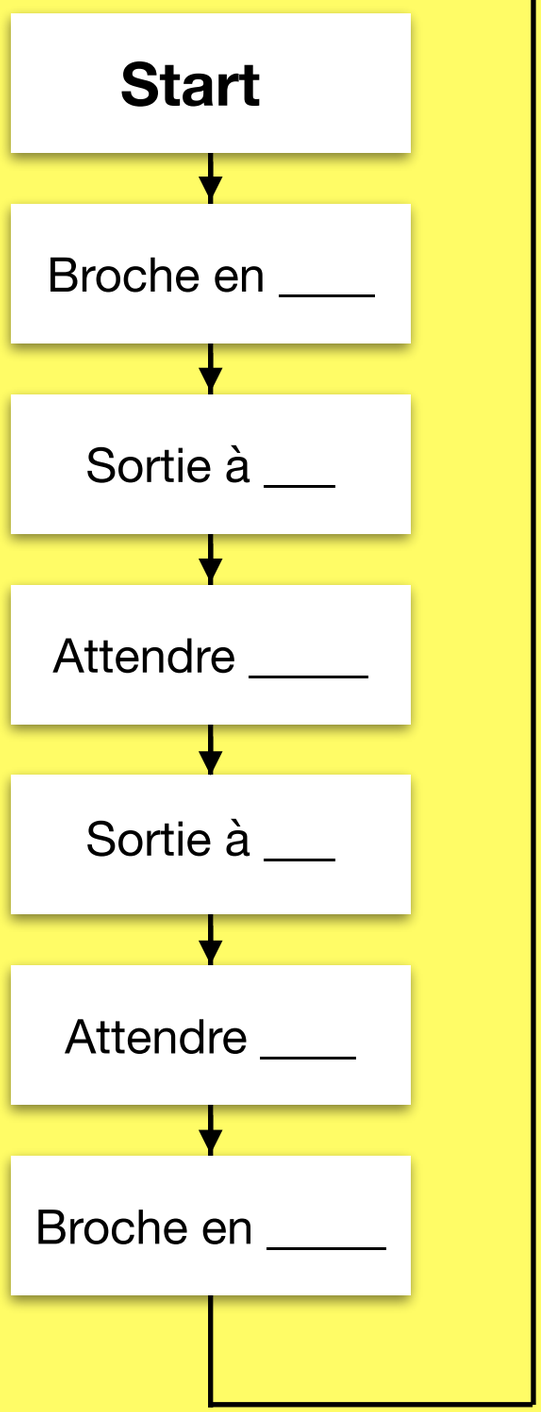
Programme en C

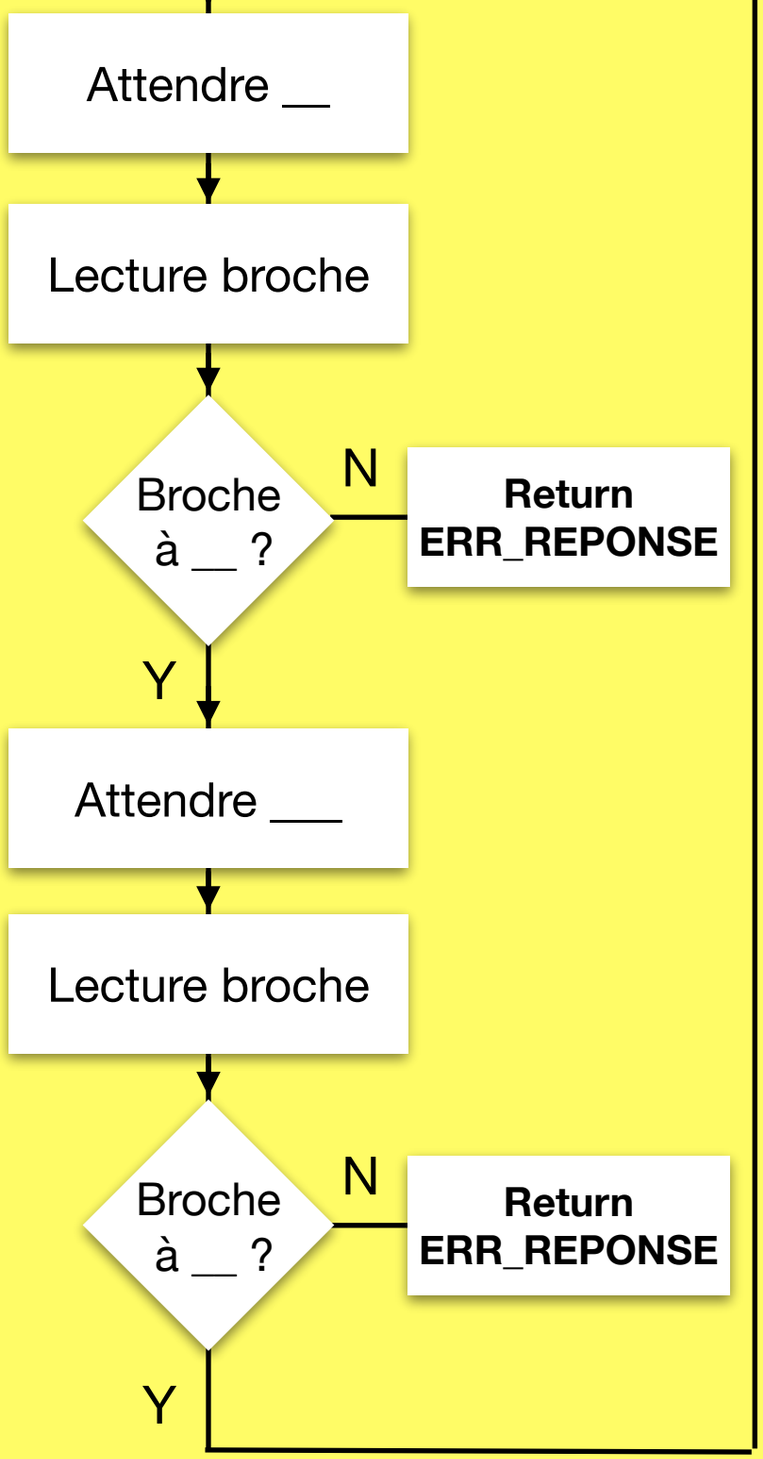
Réaliser une fonction en C/C++ afin de lancer une nouvelle mesure.

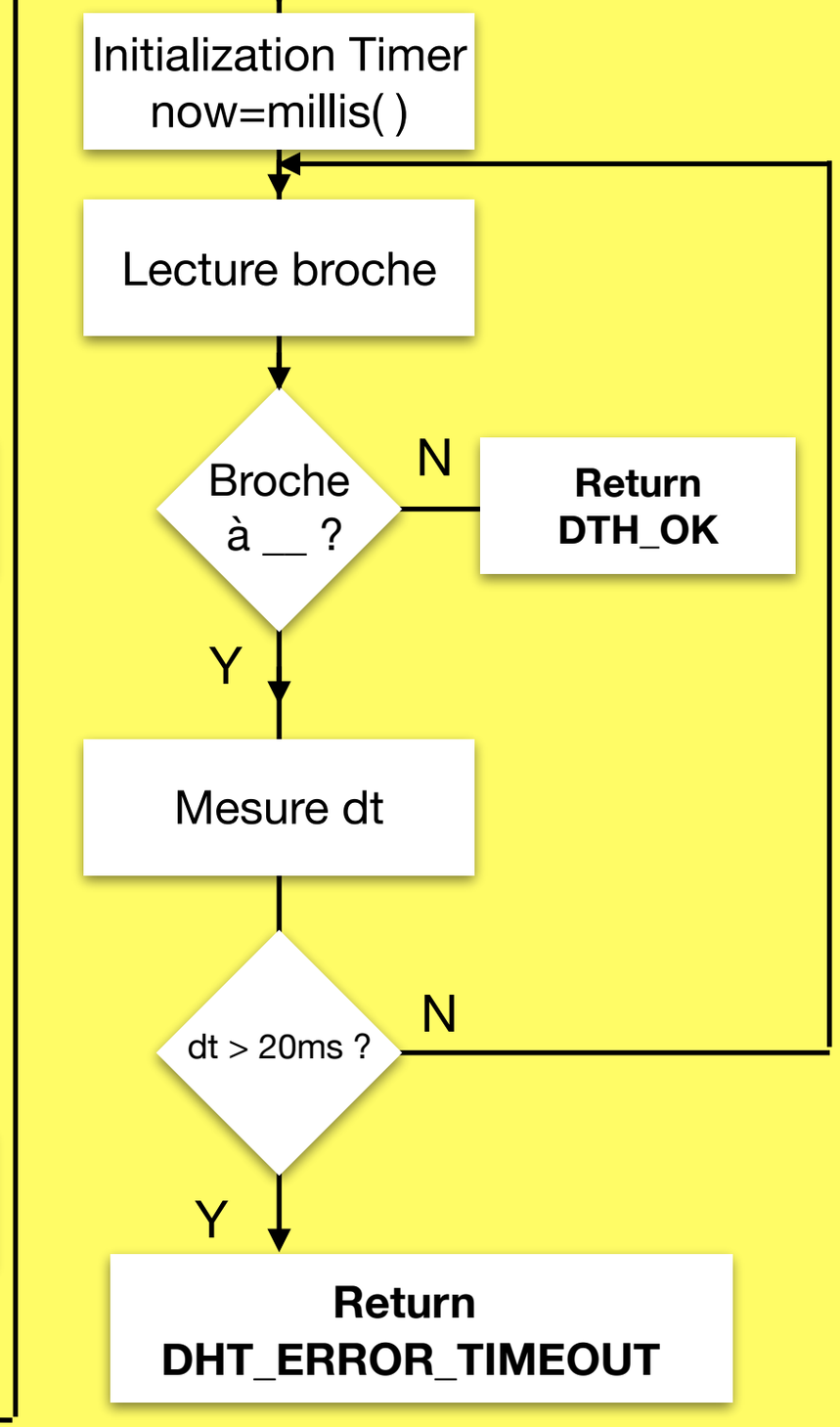
```
DHT_STATUS   DHT_start( )
{
    //todo
}
```



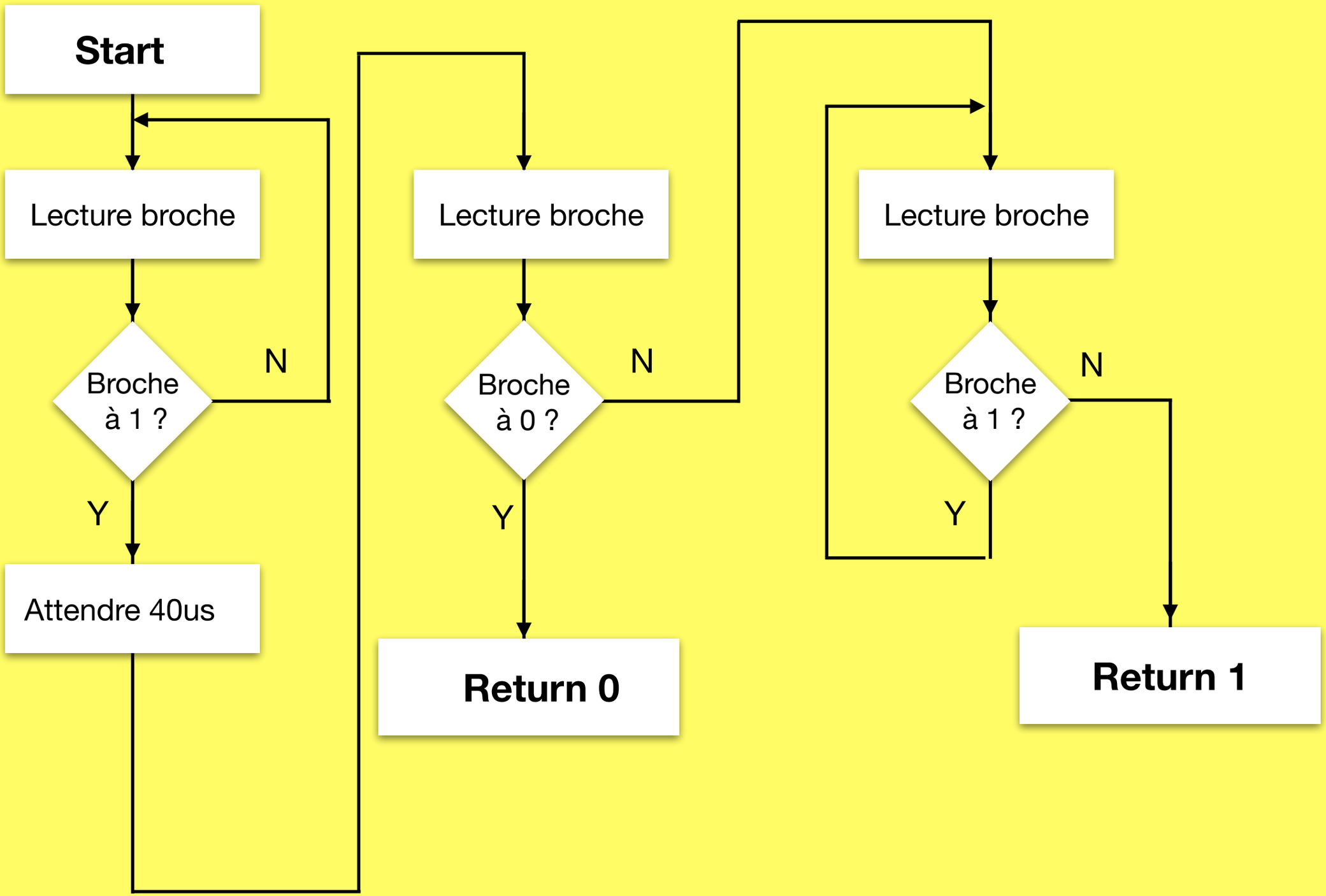
Programme en C







Lecture un bit reçu



Programme en C

Réaliser une fonction en C/C++ afin d'effectuer une lecture des données du capteur:

```
DHT_STATUS DHT_readValues()
{
    uint8_t humMSB = 0x00;
    uint8_t humLSB = 0x00;
    uint8_t tempMSB = 0x00;
    uint8_t tempLSB = 0x00;
    uint8_t check = 0x00;
    uint8_t sum = 0x00;

    DHT_STATUS status = DHT_start();

    if (st == DHT_OK)
    {
        for (uint8_t i = 0; i < NB; i++)
        {
         //todo
         }

    }

    else
     {

     //todo

      }


 return status;
}
```

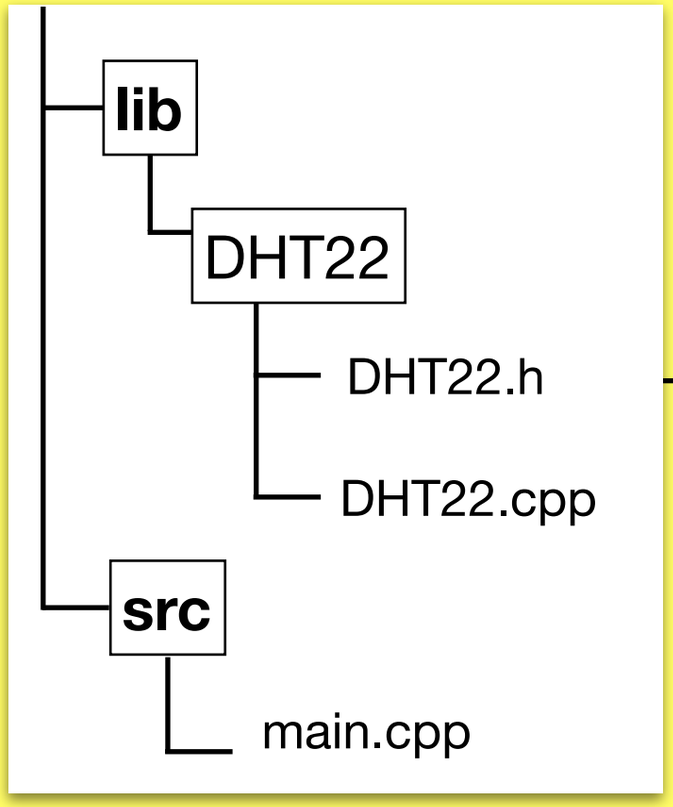
Programme en C

```
//****************************************
void action()
{
  DHT_STATUS res = DHT_readValues();

  if (res == DHT_OK)
  {
    theDewPoint = DHT_getDewPoint(theTemperature, theHumidity);

    Serial.println(“**********”);
    Serial.println(theTemperature);
    Serial.println(theHumidity);
    Serial.println(theDewPoint);

  }
  else
  {
    Serial.print("DHT Read ERROR : ");
    Serial.println(res);
  }
}
//****************************************
void setup()
{
  Serial.begin(115200);
  //
  DHT_begin();
  //
  ticker.start();
}
//****************************************
void loop()
{
  ticker.update();
}
```

Faire valider votre programme par l’enseignant.

Programme en C++

Réaliser en langage C++ la classe suivante:

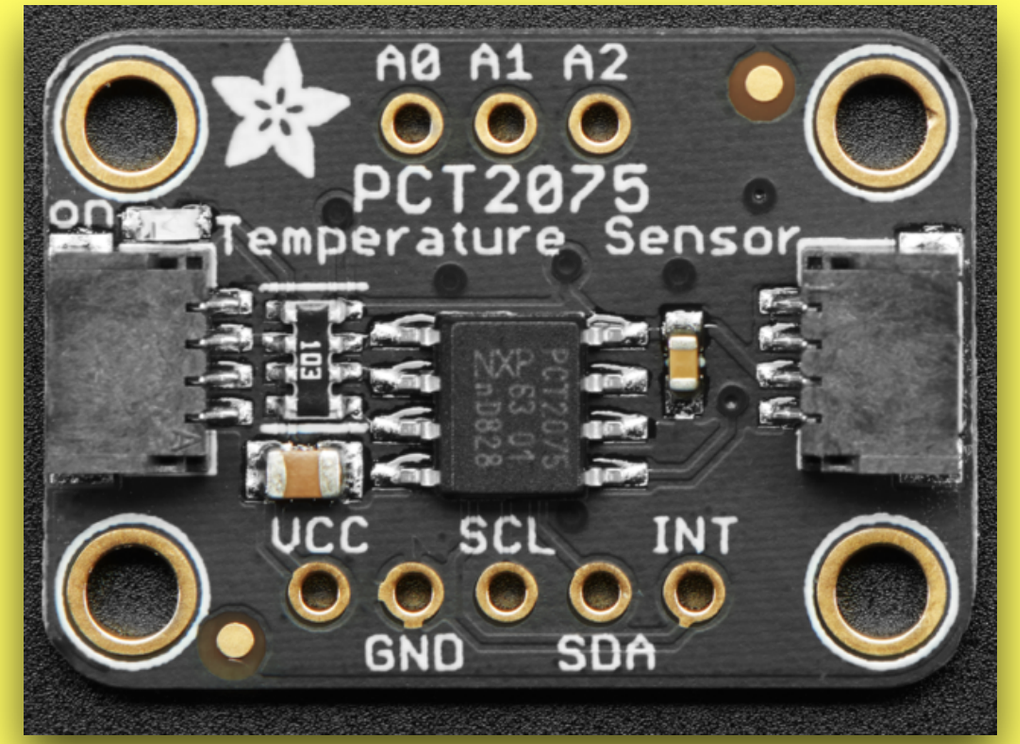



```
#ifndef DHT22_H
#define DHT22_H

#include <Arduino.h>

enum DHT_STATUS
{
    DHT_OK = 0,
    DHT_ERROR_RESPONSE,
    DHT_ERROR_CHECK_SUM,
    DHT_ERROR_TIMEOUT
};

class DHT22
{
public:
    DHT22();

    void begin(uint8_t dhtPin);

    DHT_STATUS readValues();

    float getTemperature();
    float getHumidity();
    float getDewPoint();

private:
     uint8_t _pin;
    float _temperature;
    float _humidity;
    float _dewpoint;
    uint8_t _tab[40];

    DHT_STATUS _startConv();
    DHT_STATUS _read();
    uint8_t _getByteFromTab(uint8_t startIndex);
    float _processDewPoint(float tmp, float hum);
};
#endif
```

Faire valider votre programme par l'enseignant.

Capteur PCT2075

Afin d'améliorer la précision pour la mesure de la température, nous allons utiliser en plus du DHT22, une mini plaquette d'Adafruit PCT2075.



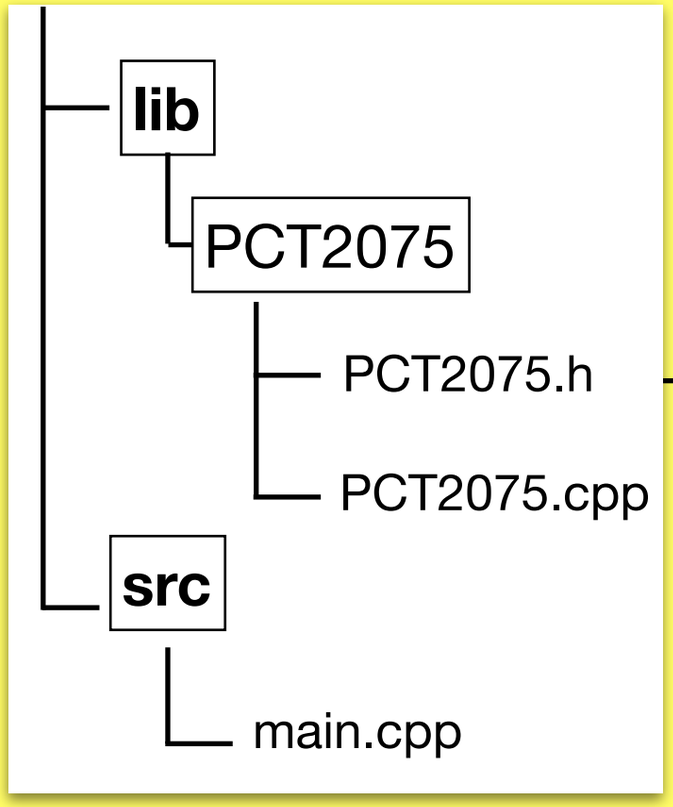
Mini plaquette PCT2075: https://www.adafruit.com/product/4369

Datasheet PCT2075: https://www.nxp.com/docs/en/data-sheet/PCT2075.pdf

IUT CERGY-PONTOISE

Programme en C++

Réaliser en langage C++ la class suivante:



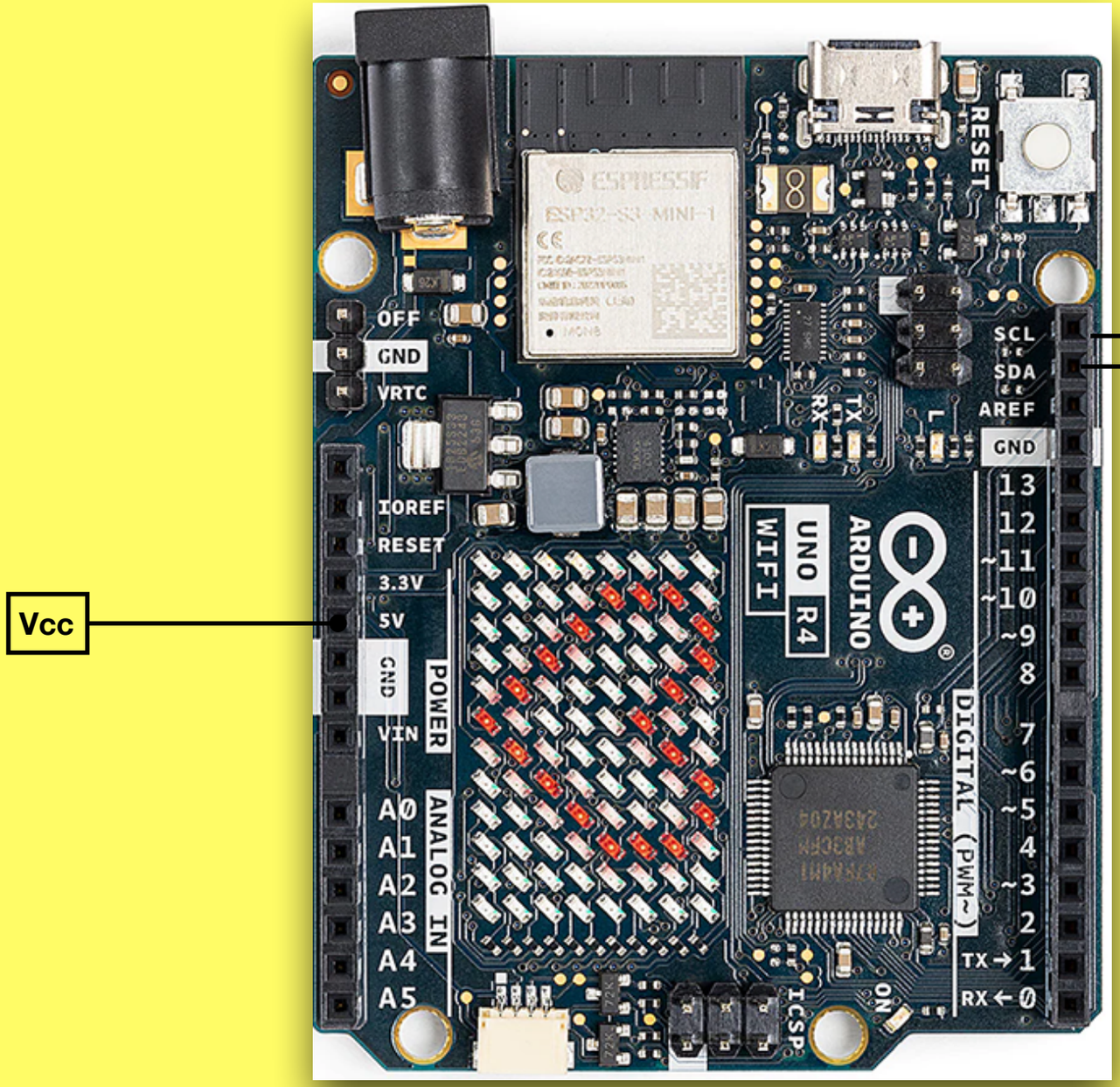

```
#ifndef PCT2075_H
#define PCT2075_H

#include <Arduino.h>

class PCT2075
{
public:
    PCT2075();

    void begin(uint8_t adress);

    float getTemperature();


private:
    uint8_t _adress;
    float _temperature;


};
#endif
```

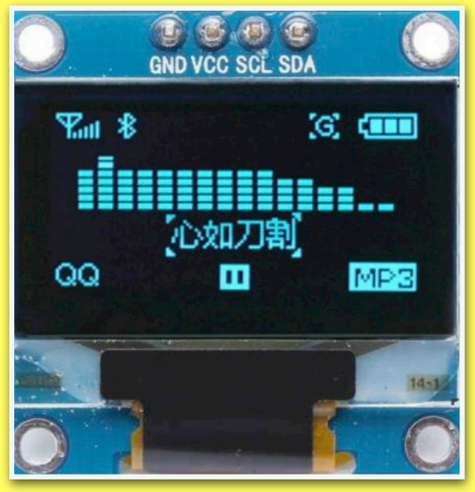
Faire valider votre programme par l'enseignant.

IUT CERGY-PONTOISE

Afficheur OLED

Réaliser le montage suivant



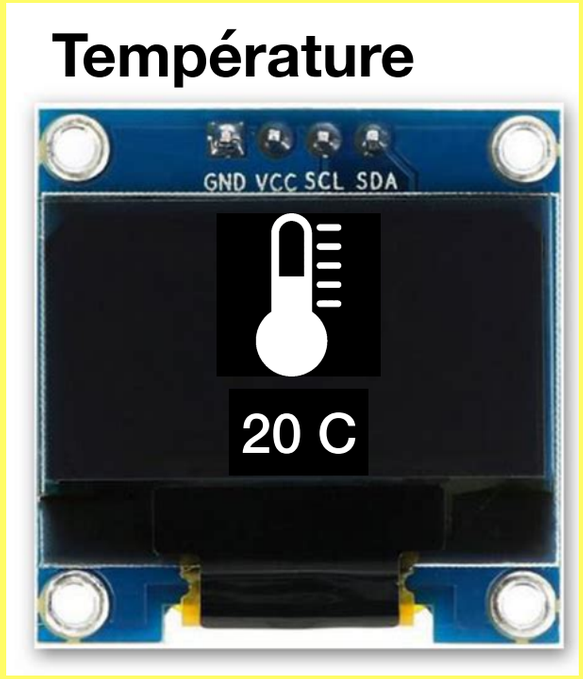


https://github.com/adafruit/Adafruit_SSD1306

https://github.com/adafruit/Adafruit-GFX-Library

Afficheur OLED

Réaliser les différents écrans graphiques

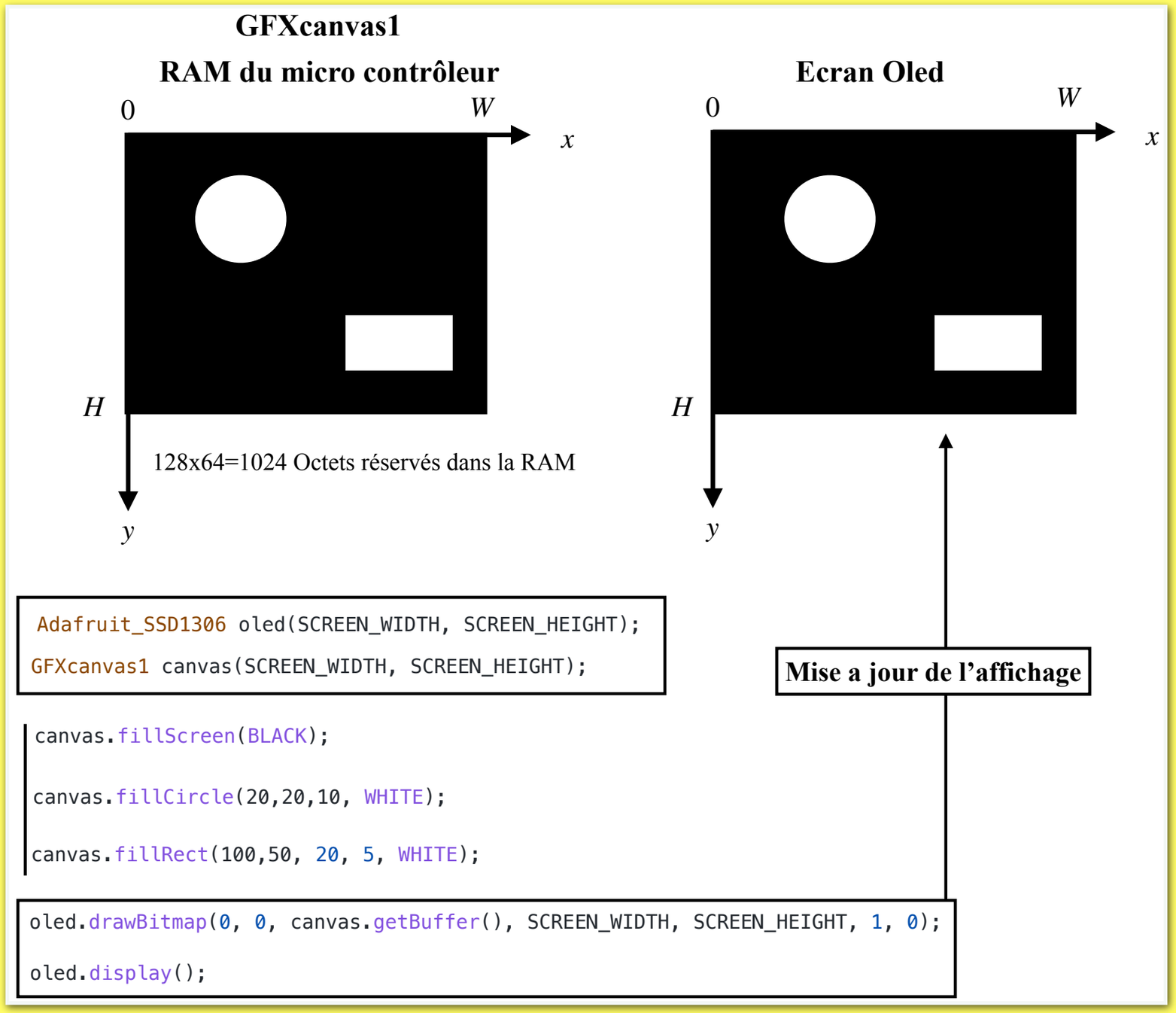


Application Web pour transformer une image en tableau de char

https://javl.github.io/image2cpp/

Afficheur OLED

Utilisation de l'objet GFXCanvas



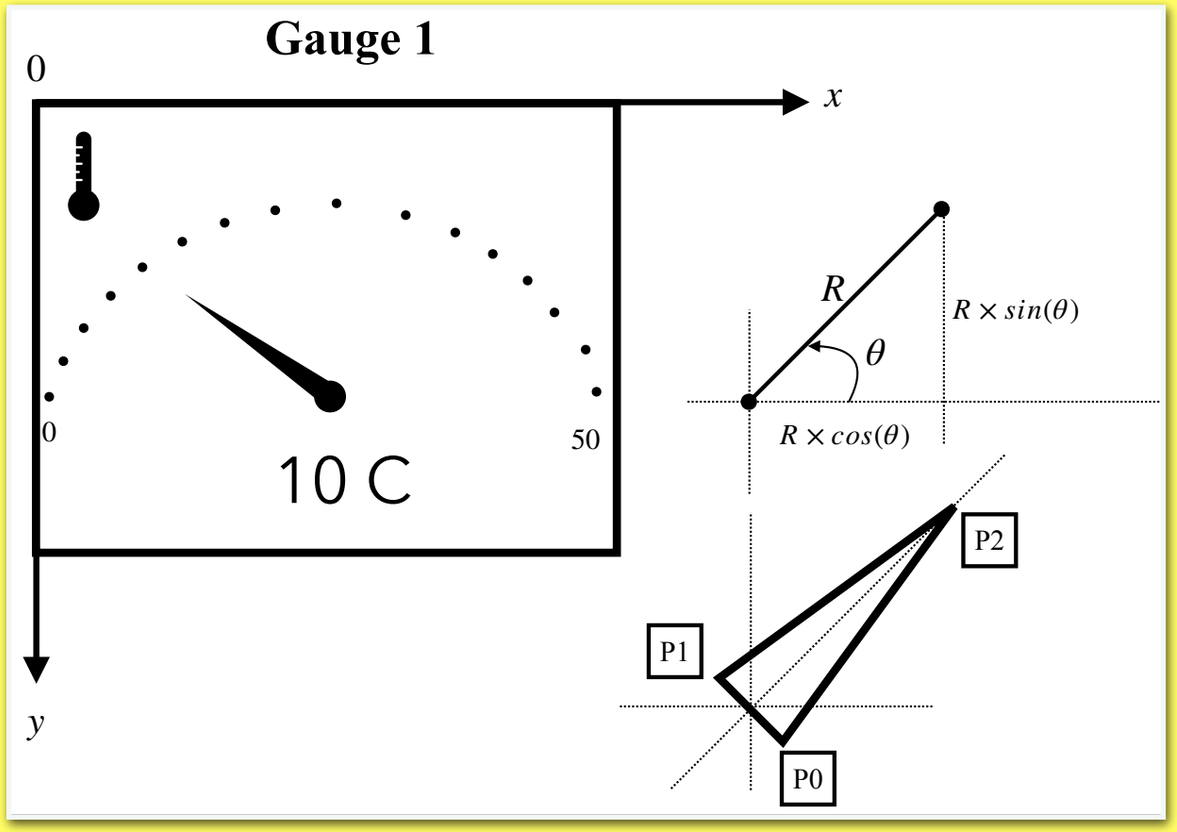

```
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT);
GFXcanvas1 canvas(SCREEN_WIDTH, SCREEN_HEIGHT);
```

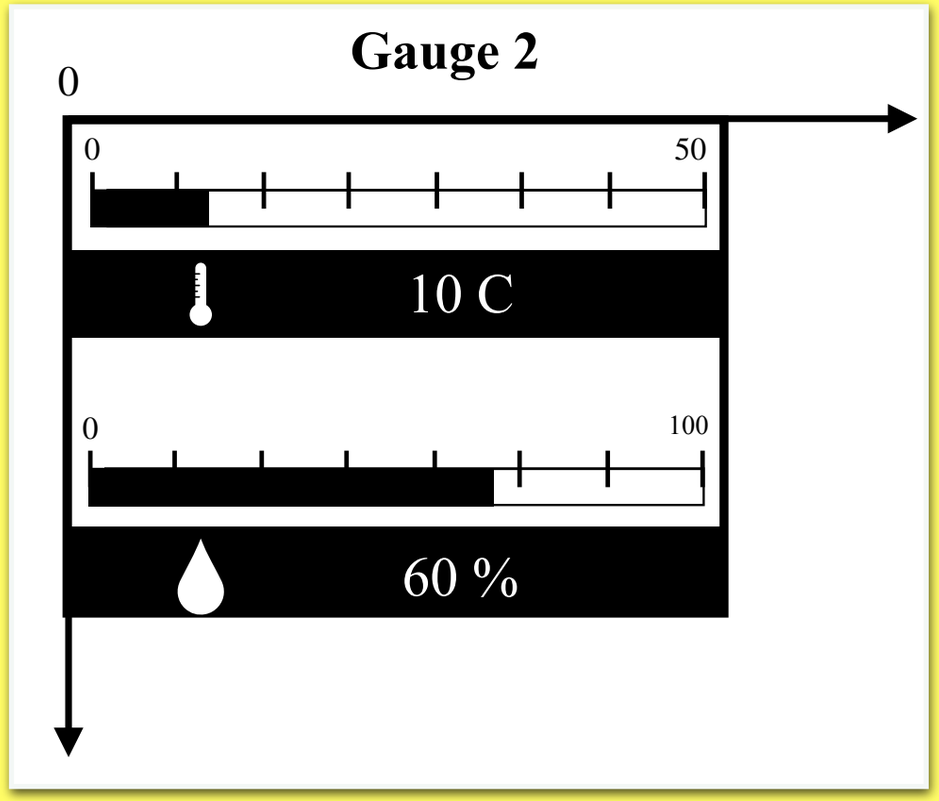
```
canvas.fillScreen(BLACK);

canvas.fillCircle(20,20,10, WHITE);

canvas.fillRect(100,50, 20, 5, WHITE);
```

```
oled.drawBitmap(0, 0, canvas.getBuffer(), SCREEN_WIDTH, SCREEN_HEIGHT, 1, 0);
oled.display();
```

Afficheur OLED

Utilisation de l'objet GFXCanvas





Format JSON

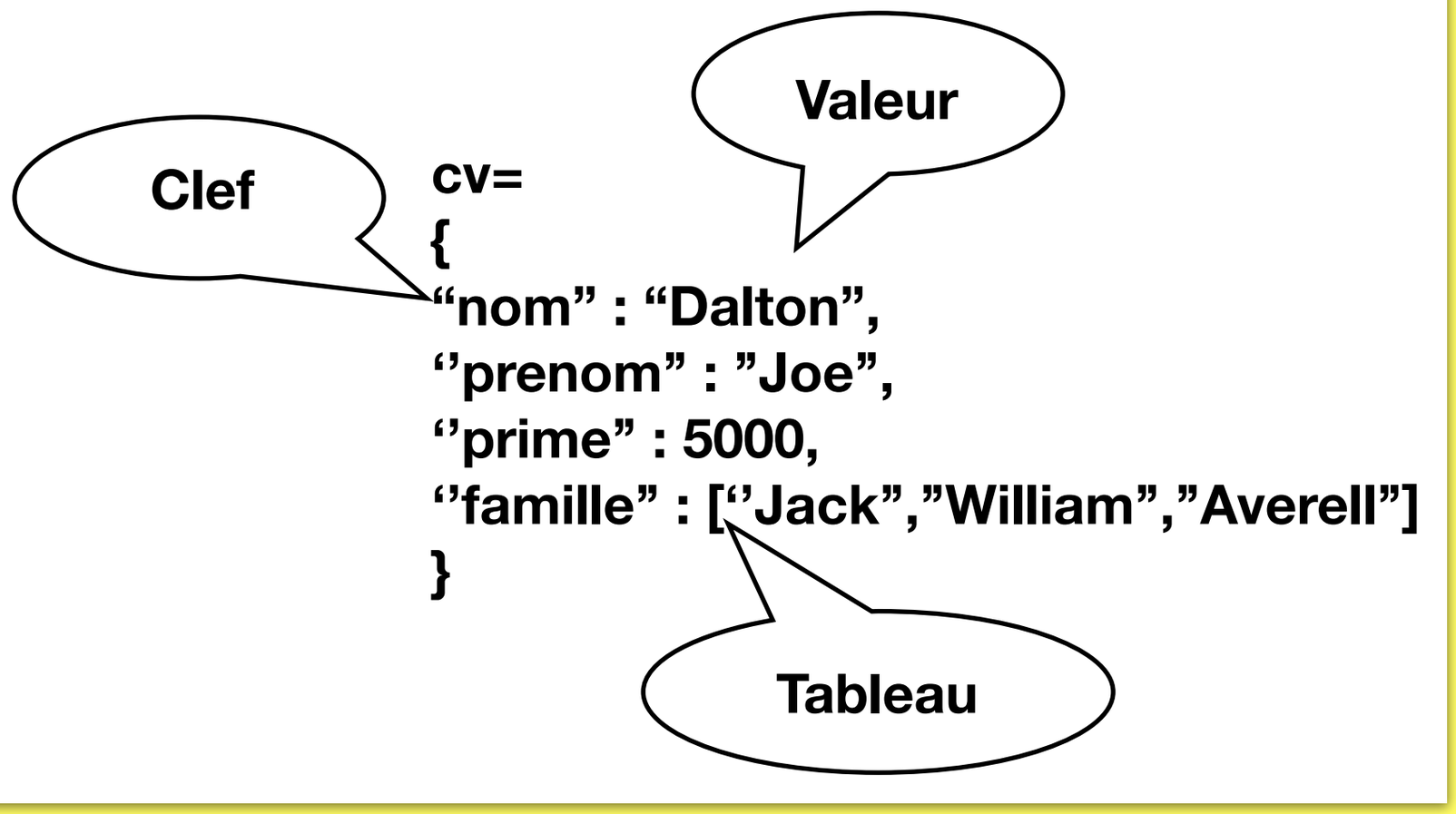
Le Javascript Object Notation (JSON) est un format de données textuel dérivé de la notation des objets littéraux en javascript.

Un text en JSON comprend:

- **Des objets javascript, ou ensemble «clef/valeur »**
- **Des listes ordonnées ou tableau**
- **Des booléens**
- **Des nombres**
- **Des chaines de caractères (String)**
- **Valeur null**

Format JSON

Exemple de donnée en JSON



Pour accéder aux données:

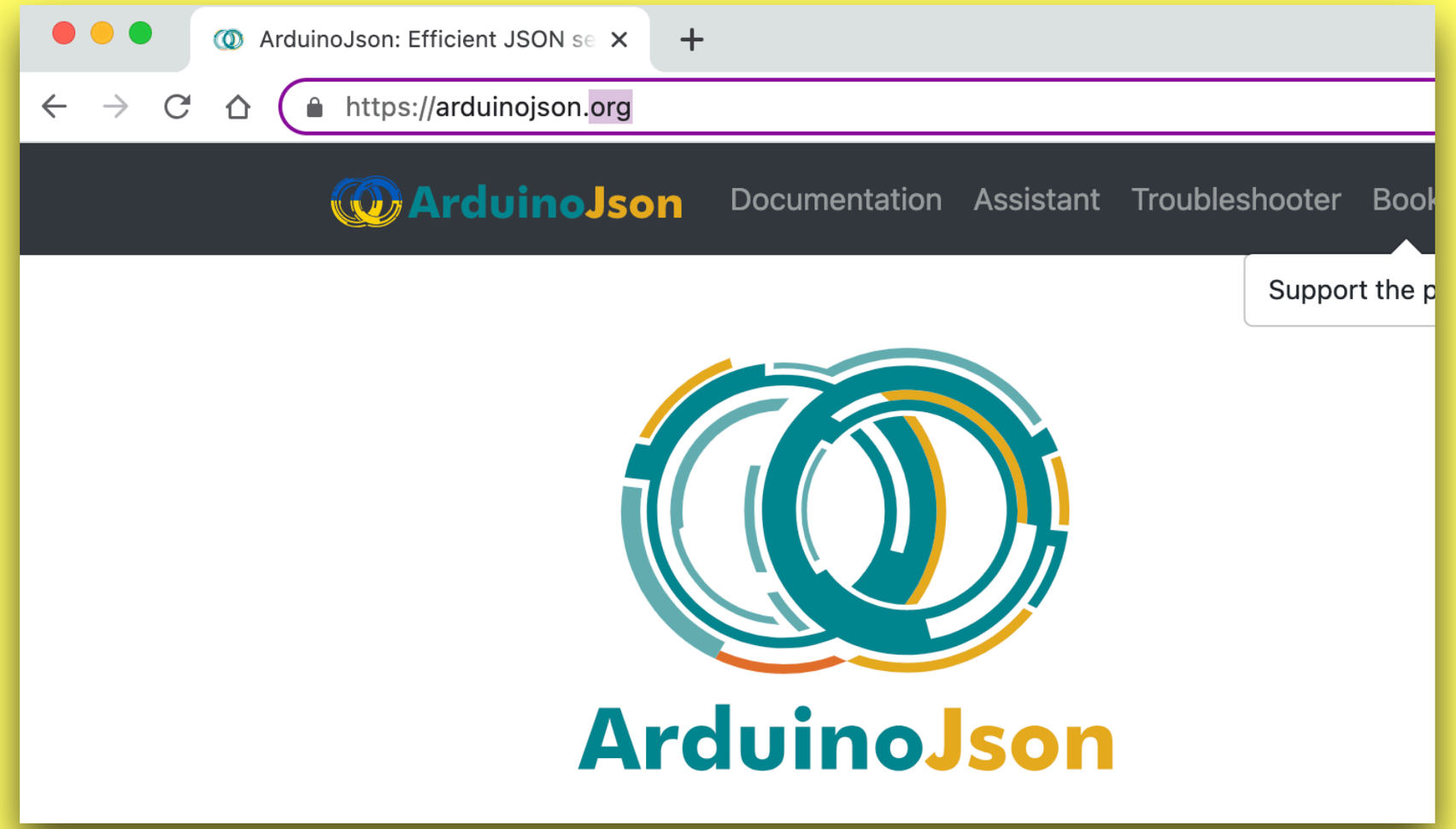
String name=cv.nom **ou String** name=cv[‘’nom”]

String first=cv.famille[0] //first=Jack

String last=cv.famille[2] // last=Averell

Bibliothèque ArduinoJson

Pour facilité l'utilisation des données JSON.
Nous utiliserons la bibliothèque « ArduinoJson »



Bibliothèque ArduinoJson

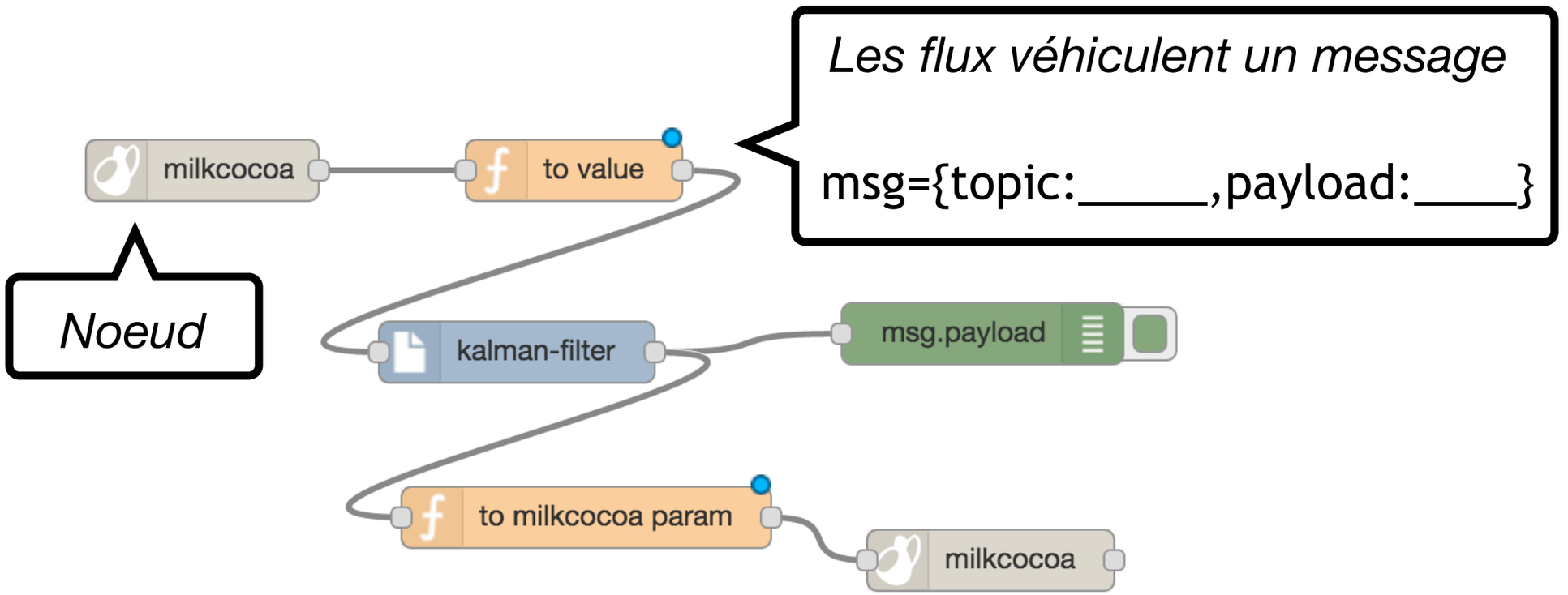
Utiliser une telle bibliothèque afin de créer et d'afficher dans la console série, le JSON suivant:

```
{
''board":"uno4xx",
"dhtStatus":"DHT_OK",
"temperature":23,
"humidity":30,
"dewPoint":18,
"ledstate":0,
}
```

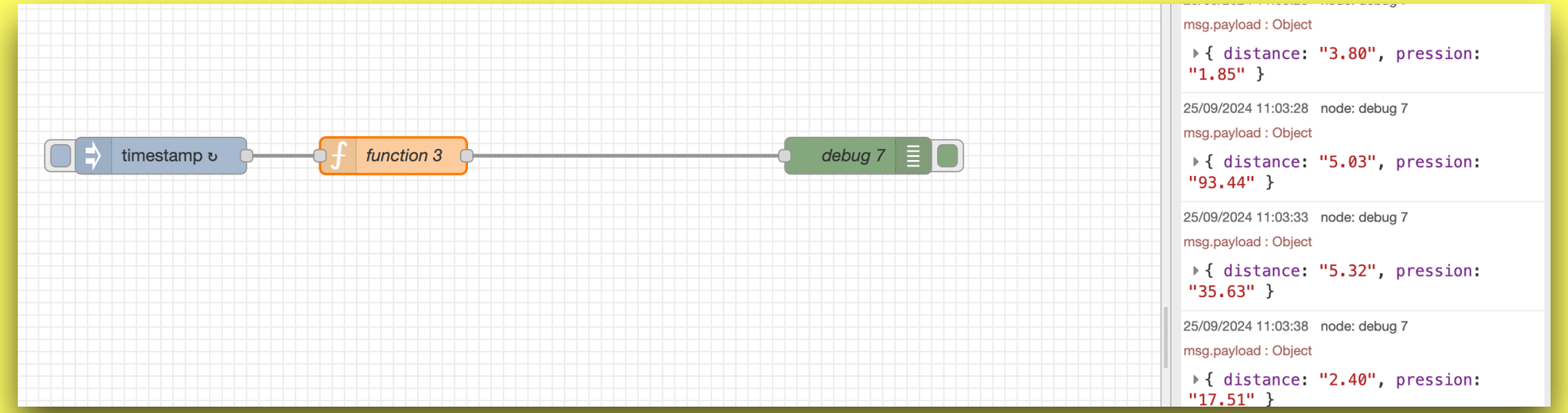
Faire valider votre programme par l'enseignant.

Node-RED

Environnement de programmation graphique pour développer rapidement des applications, dédiées aux objets connectés.



Soit une fonction qui délivre périodiquement un objet json.

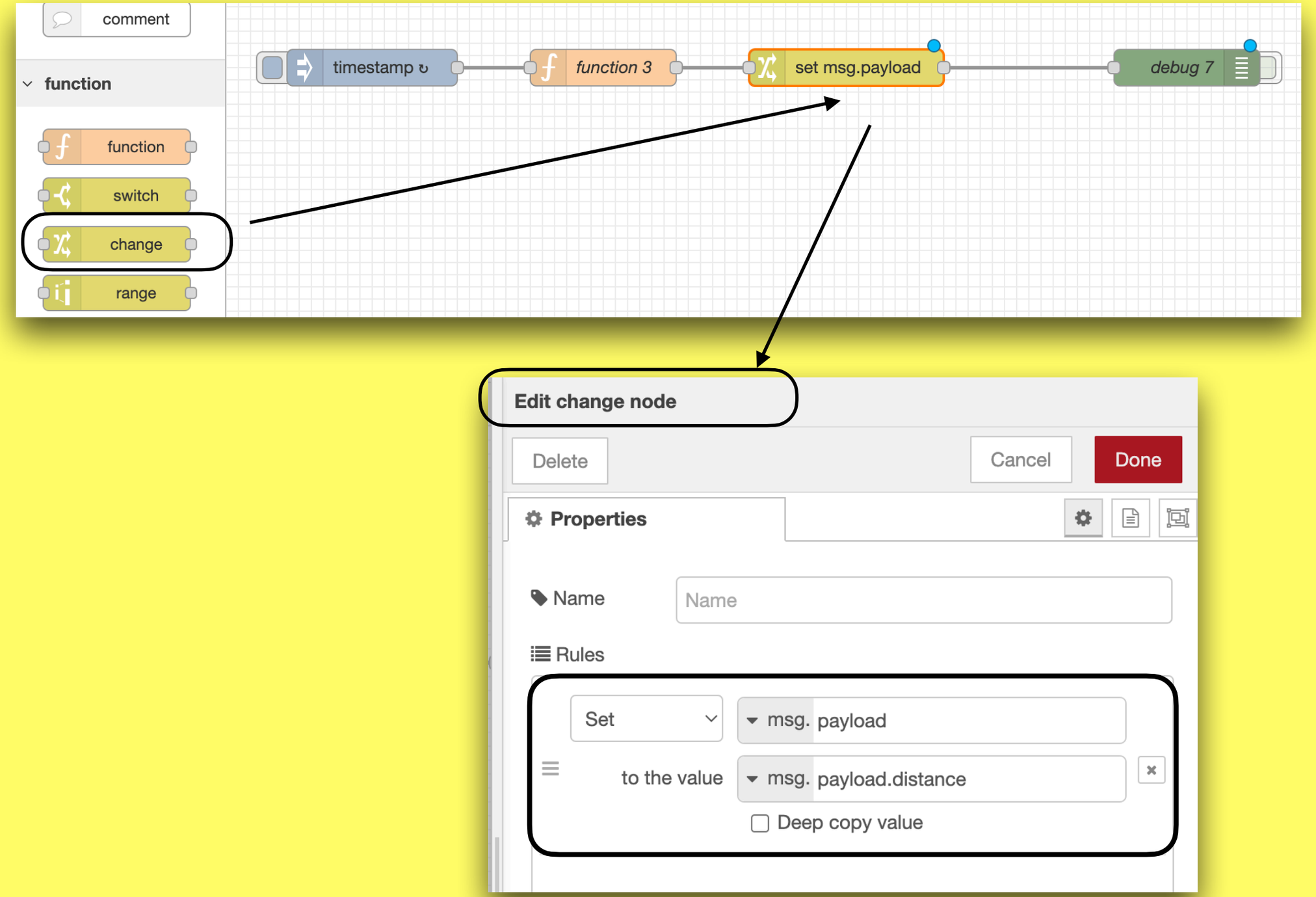


On souhaite extraire la valeur de la distance, afin de l'afficher via une gauge graphique.

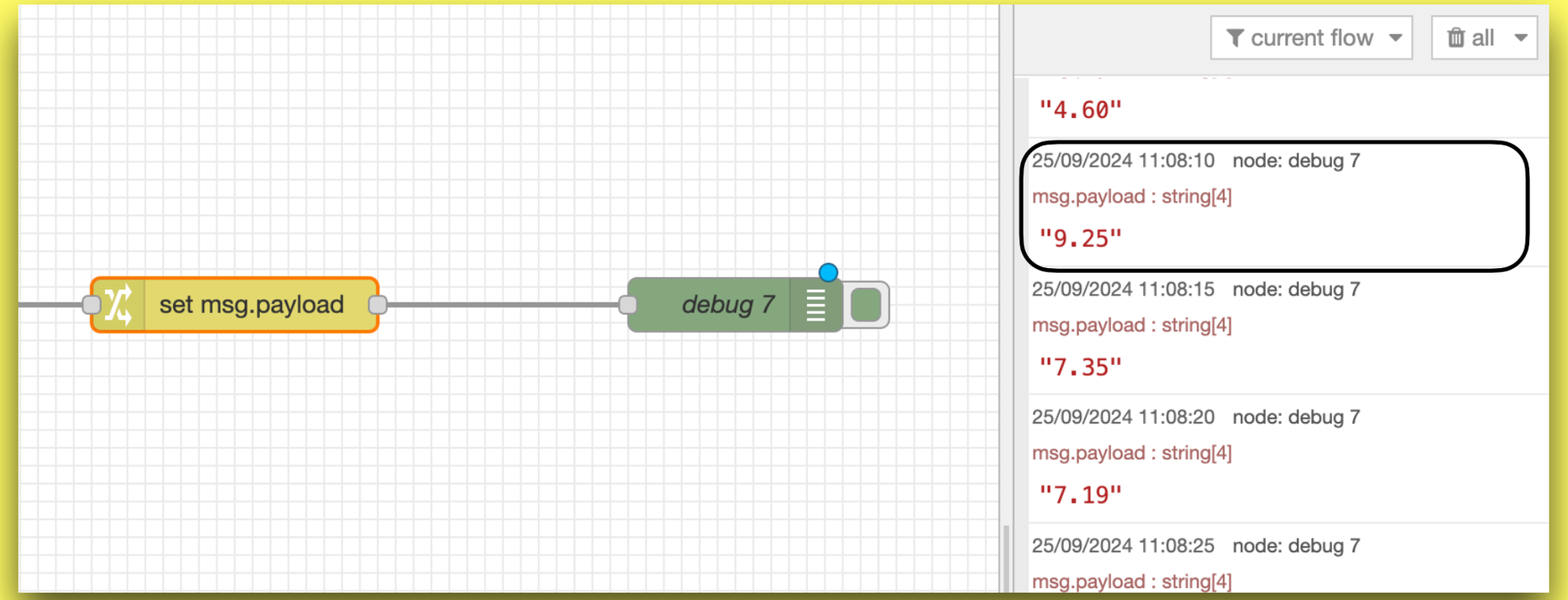
Node-RED

Tutorial simple

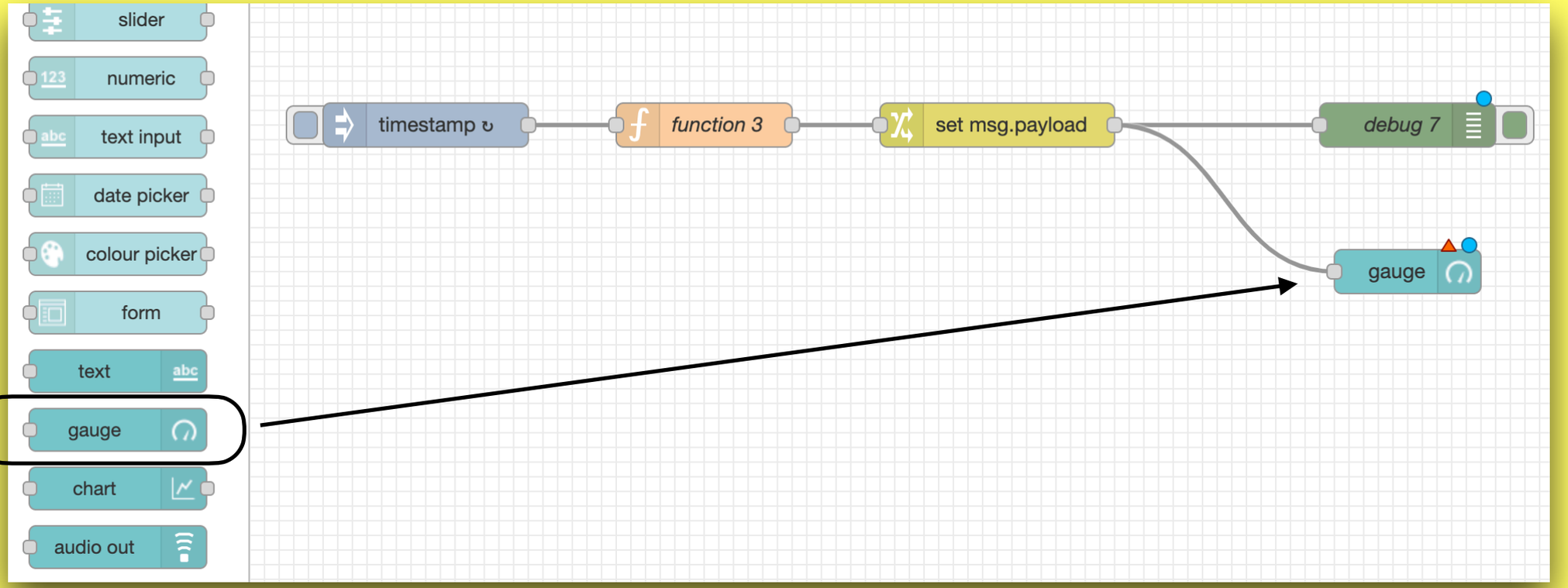
Placer un noeud “Change” en sortie de la fonction.



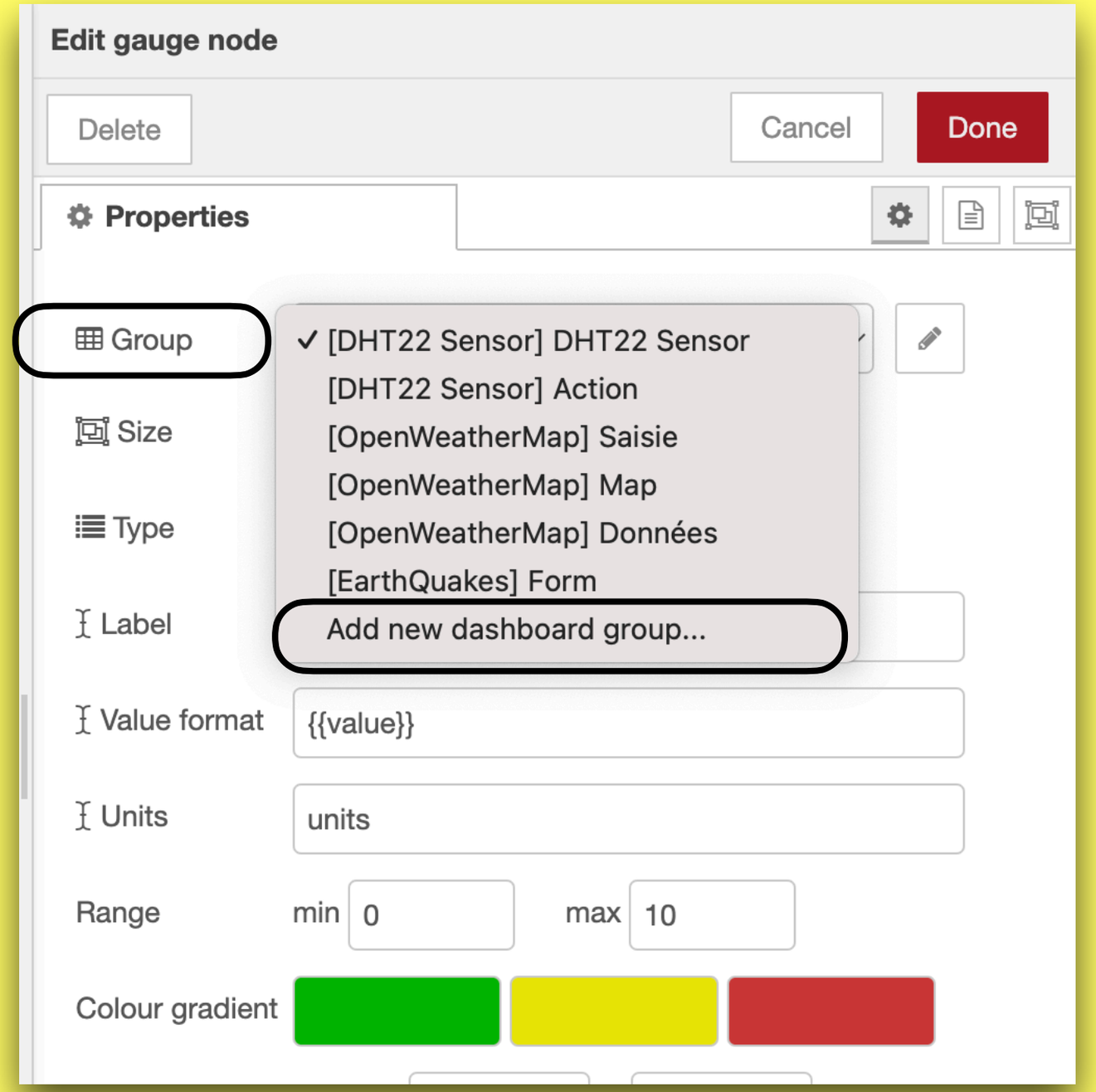
Après avoid déployé le projet, on obtient dans la console la valeur de la distance.



Placer un noeud “Gauge” comme indiquer ci dessous.

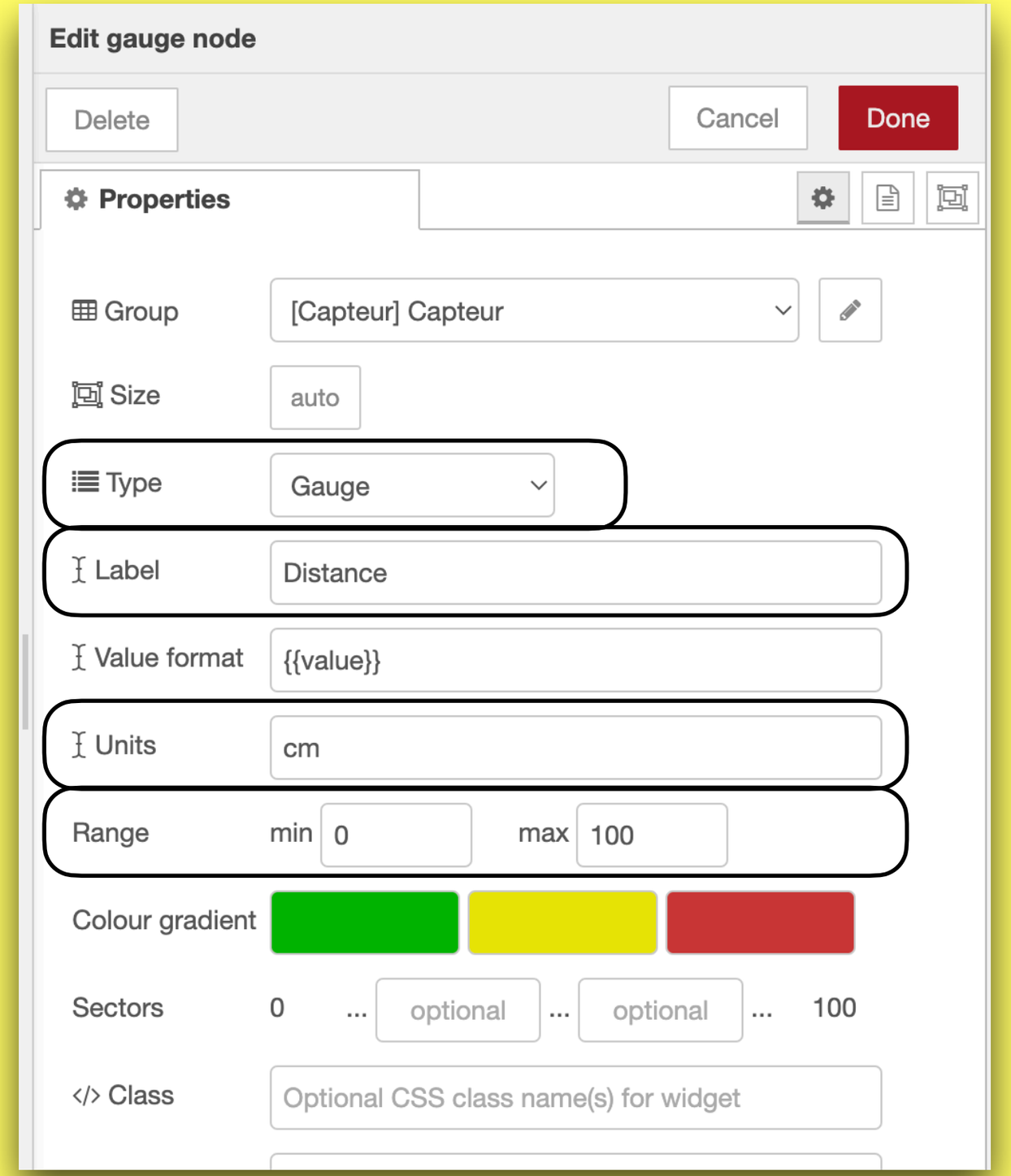


Editer le noeud “Gauge”, et créer un nouveau groupe de composants.

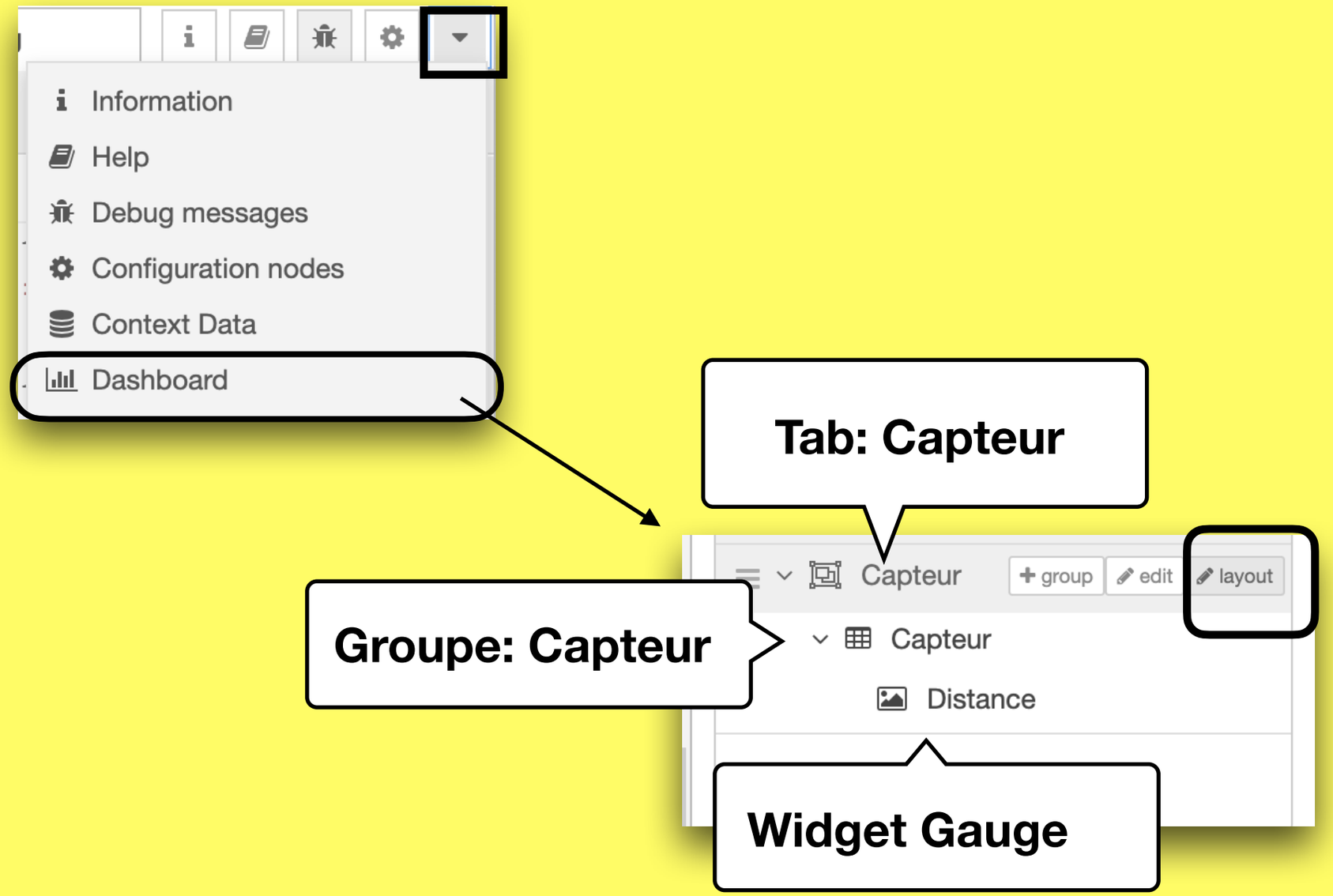



Edit gauge node > Add new dashboard group config node
Add
Propertie
Nom du groupe
Name
Capteur
Tab
Capteur
Class
Créer une nouvelle Tab
(nouvelle page HTML)
Width
Allow group to be collapsed

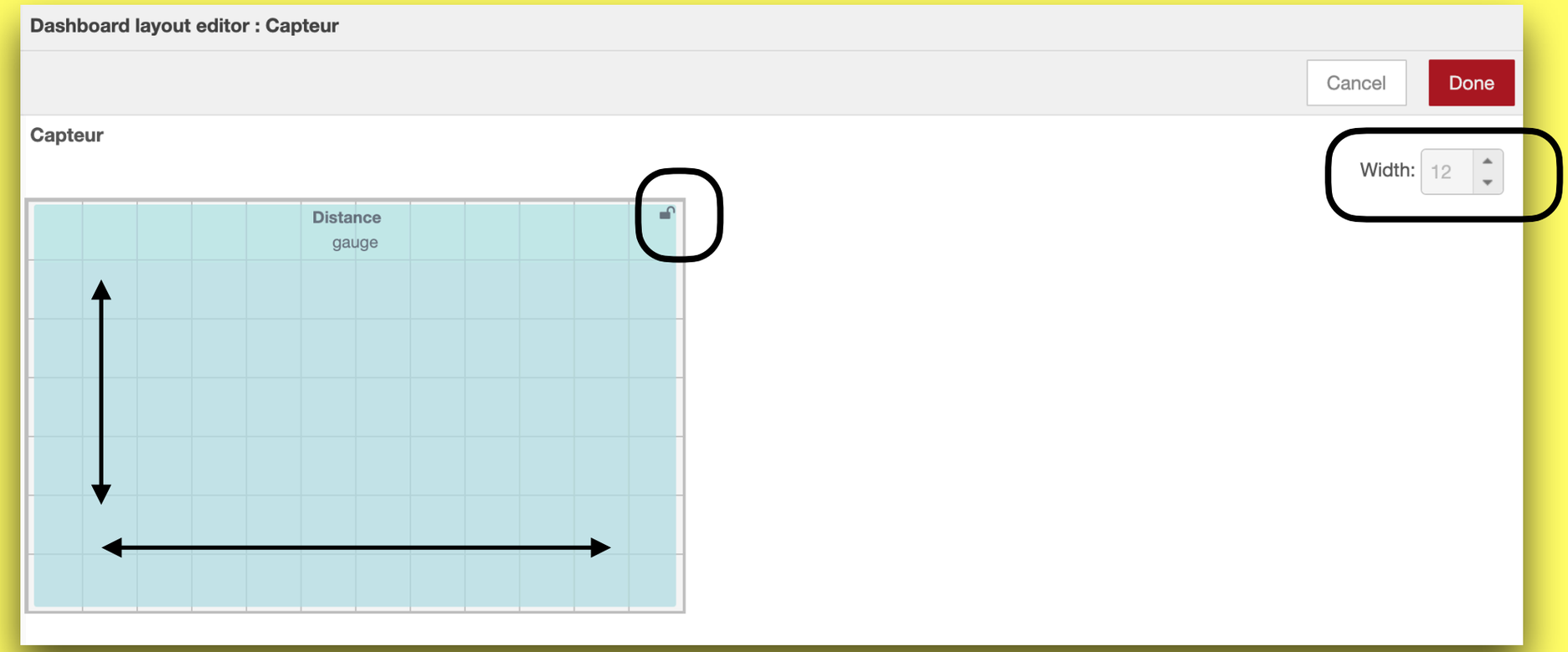
Compléter les autres paramètres comme indiquer ci dessous.



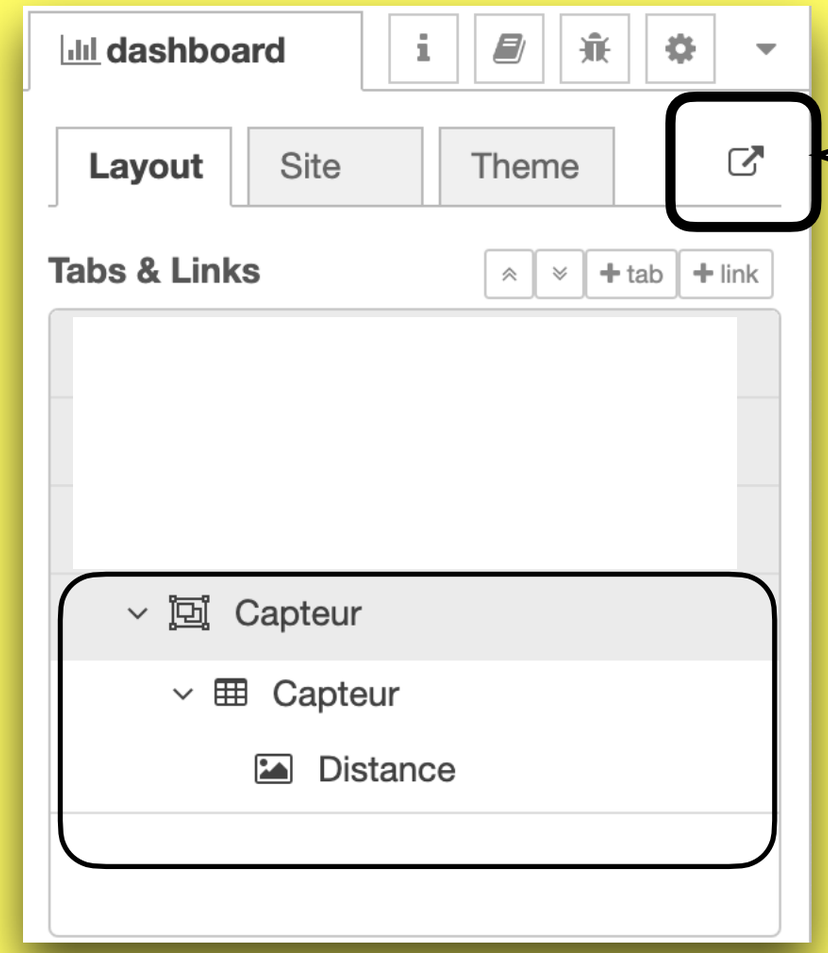
On peut gérer la disposition des widgets du groupe via le menu “layout” .



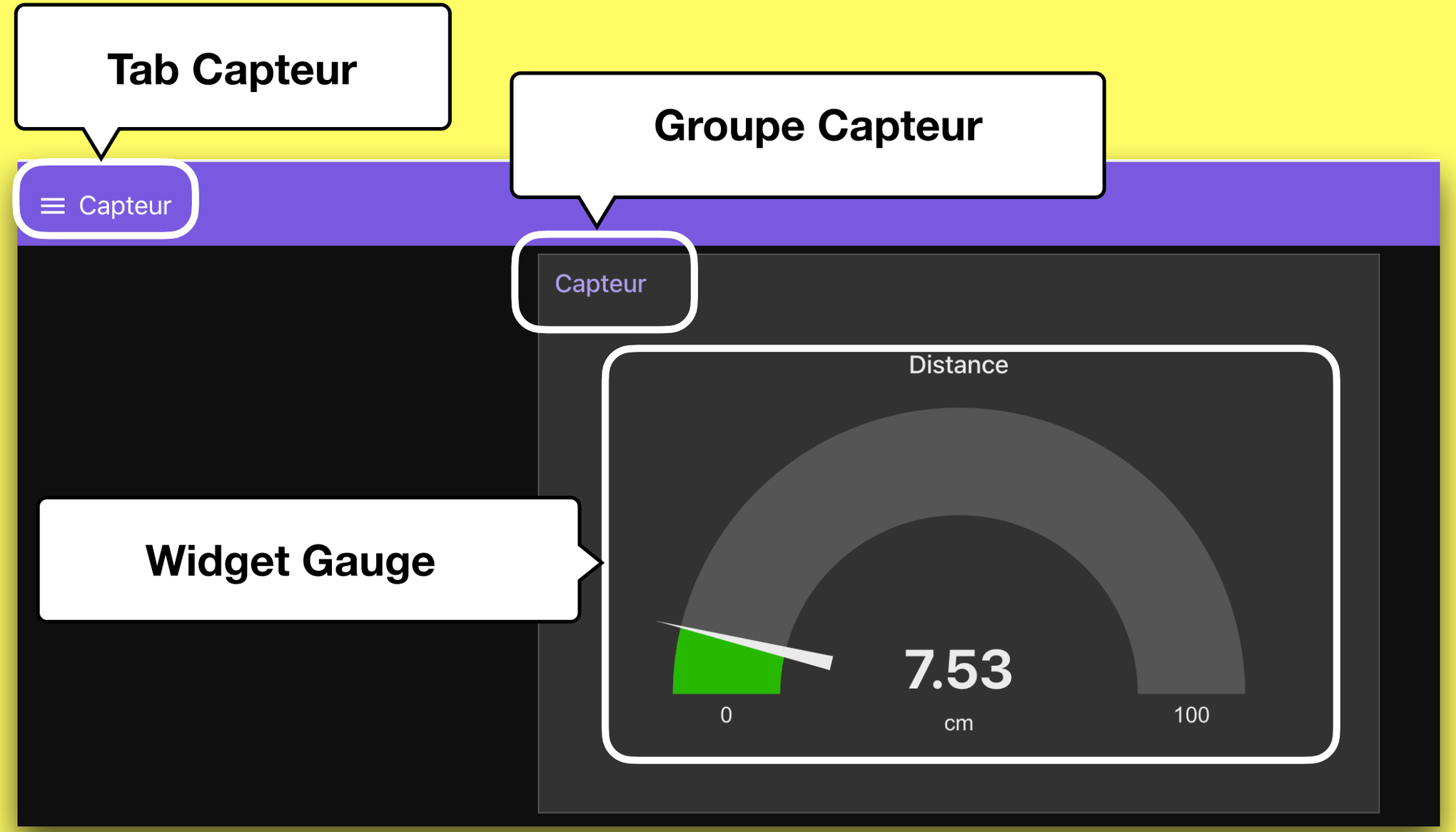
Via la souris on peut changer la disposition des widgets.



Après avoir déployé votre projet .
La page contenant les widgets est donnée via ce lien

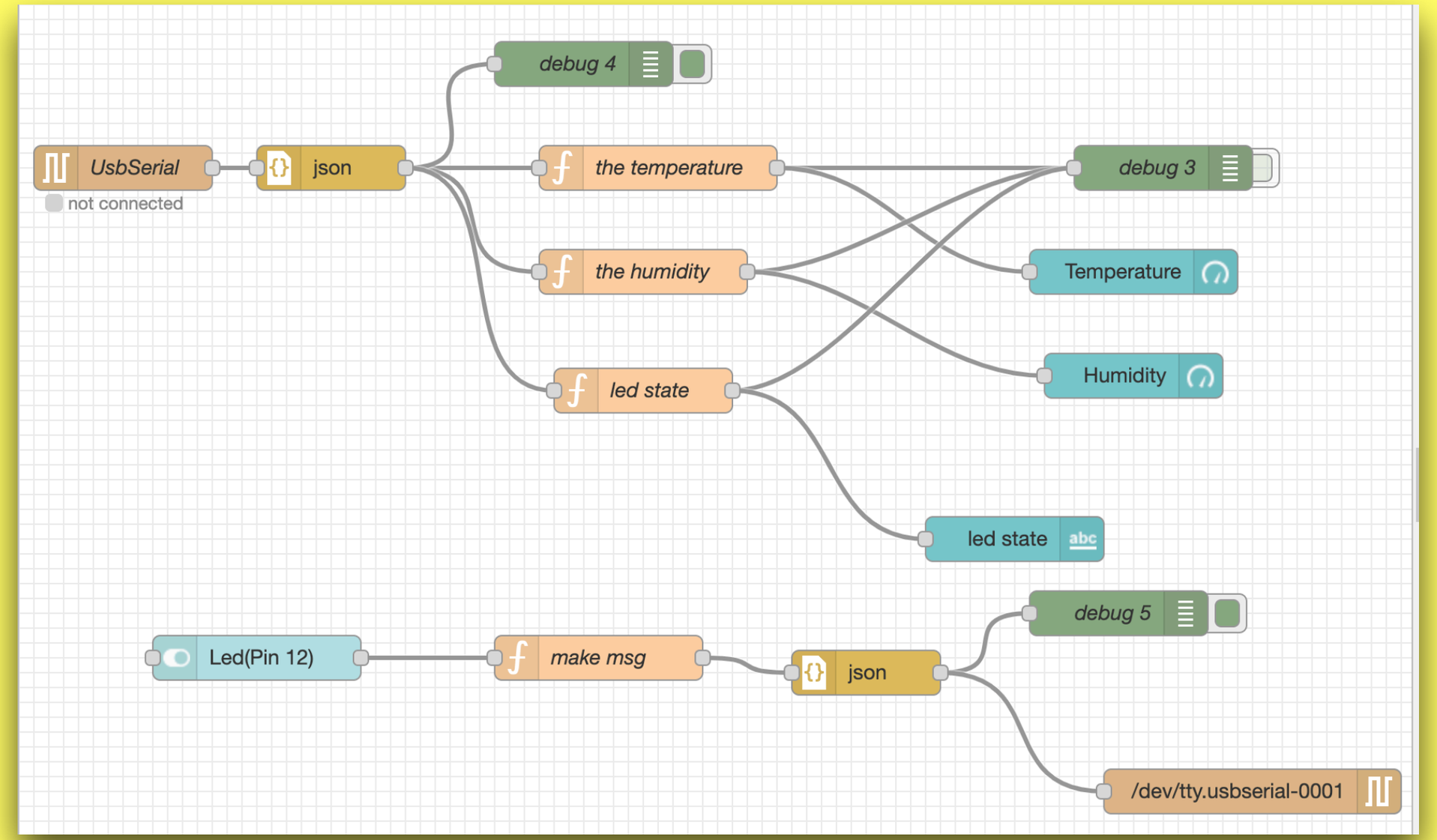


Le dashboard doit ressembler à la fenêtre ci dessous (en fonction du thème choisi) .



Node-RED

Utiliser l'environnement Node-RED afin de créer un dashboard, pour visualiser les données via la liaison série (USB).



MQTT

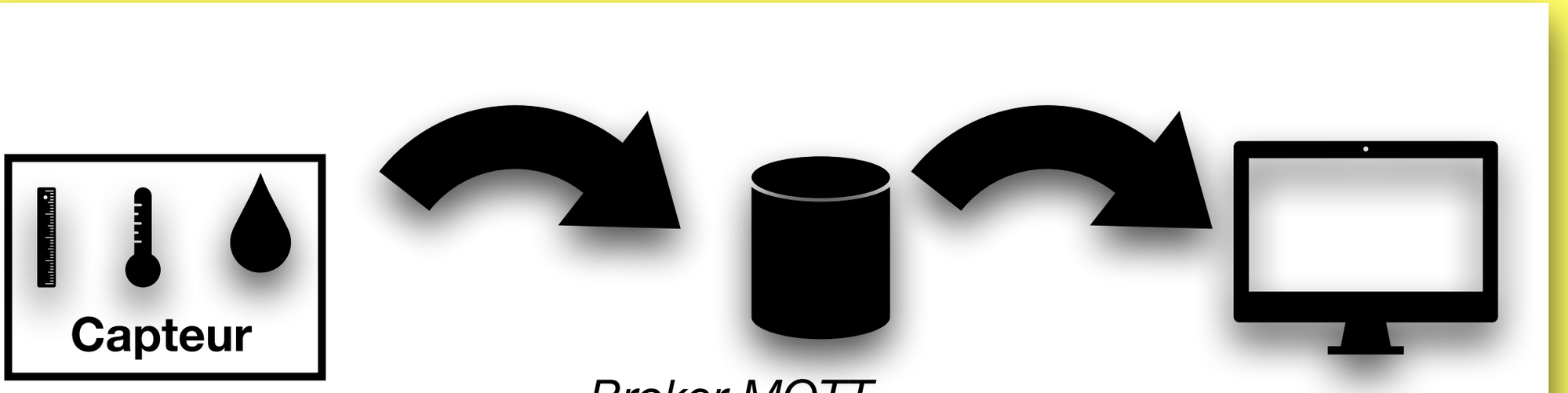
- MQTT: Message Queing Telemetry Transport
  Protocole open source de transfert de données basé sur l'Ethernet TCP/IP.

- MQTT est très utilisé pour le transfert « Machine to Machine ».
  Il est considéré comme l'un des principaux protocoles de message de l'internet des objets.

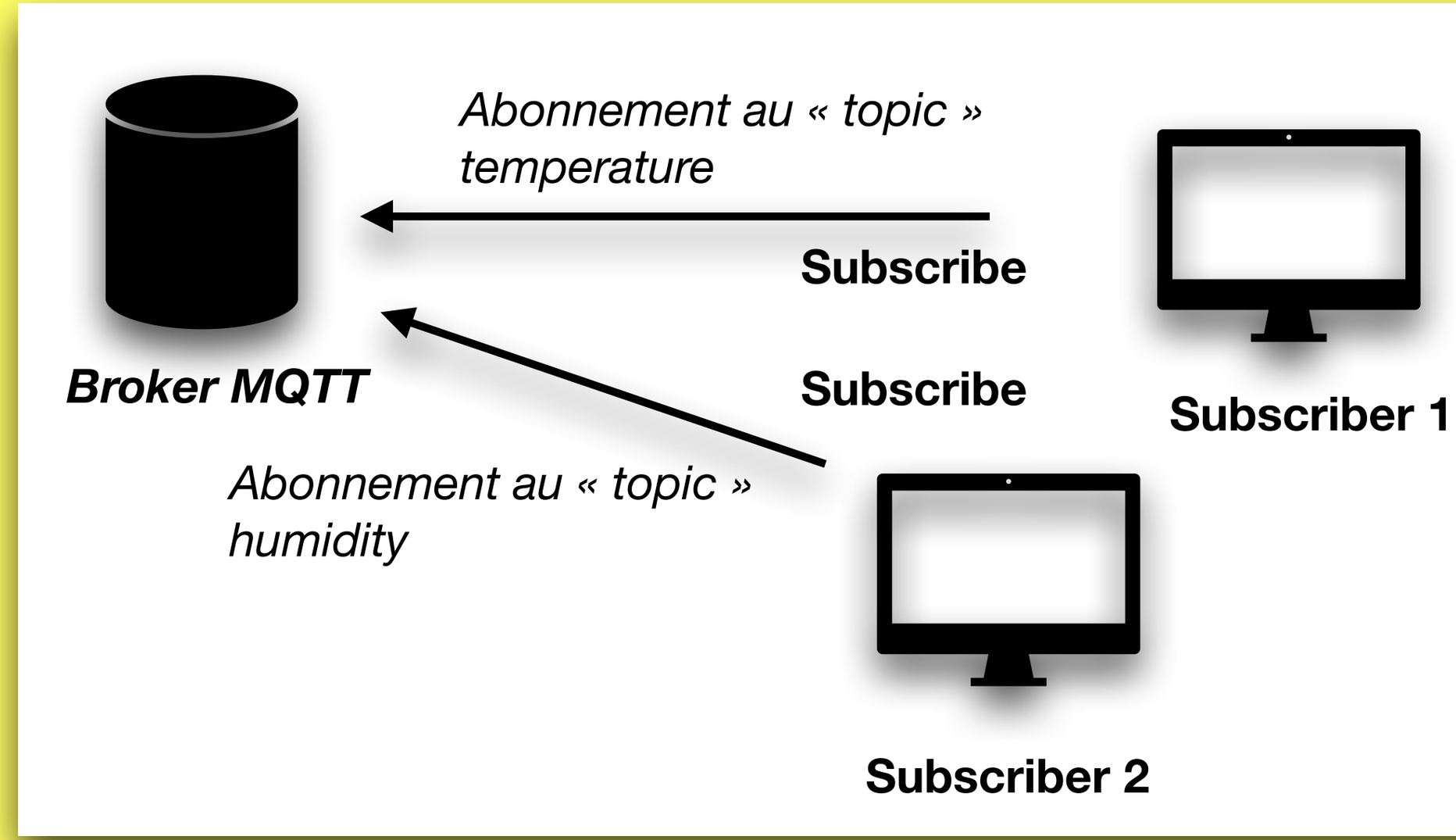
- Apparu en 1999, standardisé ISO depuis 2014.

MQTT

- Protocole léger avec une faible utilisation de la bande passante. Fiable et rapide (93 fois plus rapide que HTTP).
- MQTT fonctionne sur le principe de “publication/abonnement” ou en anglais publish/subscribe.

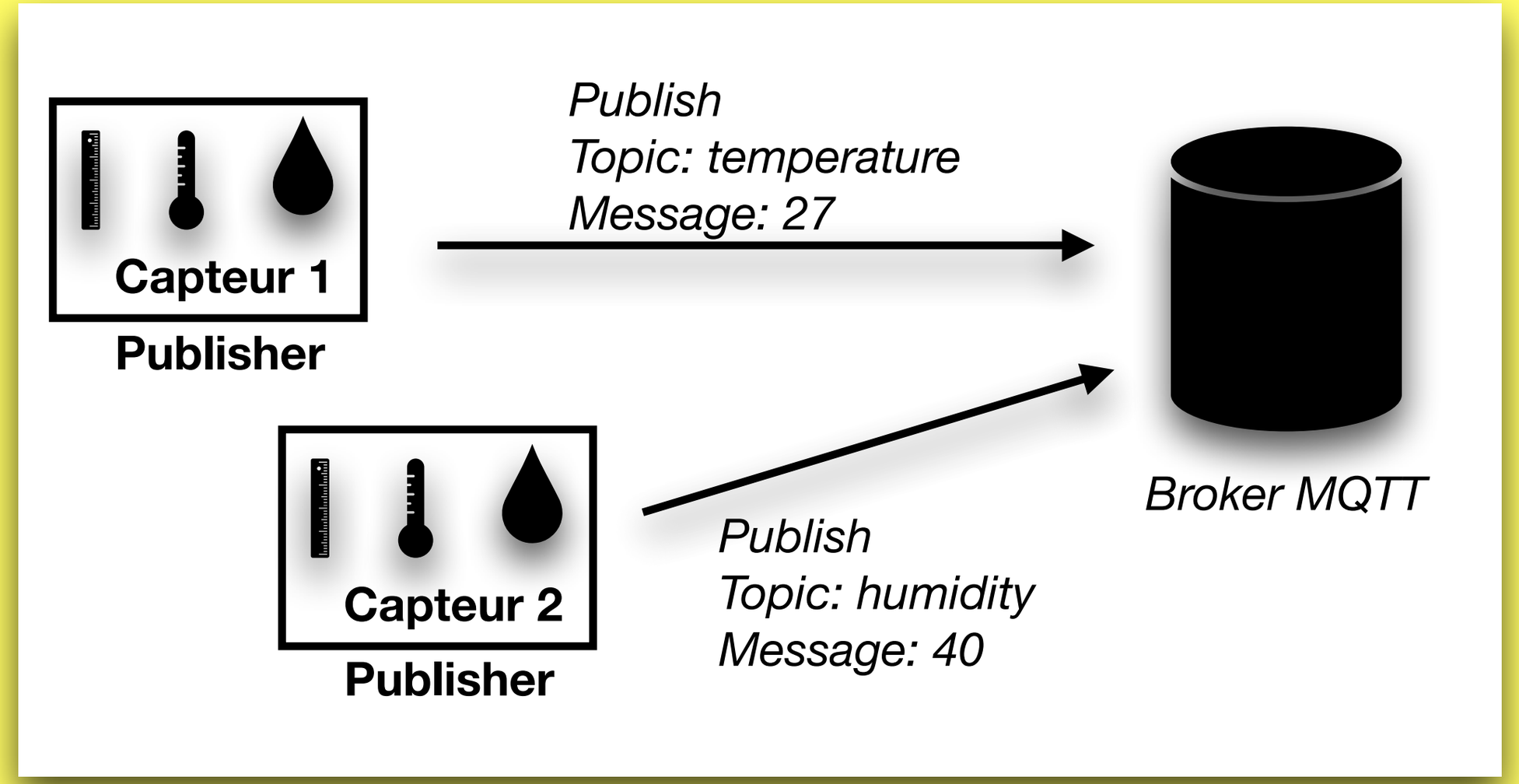


Etape 1
Les différents “ subscriber ” s’abonnent au près du broker.

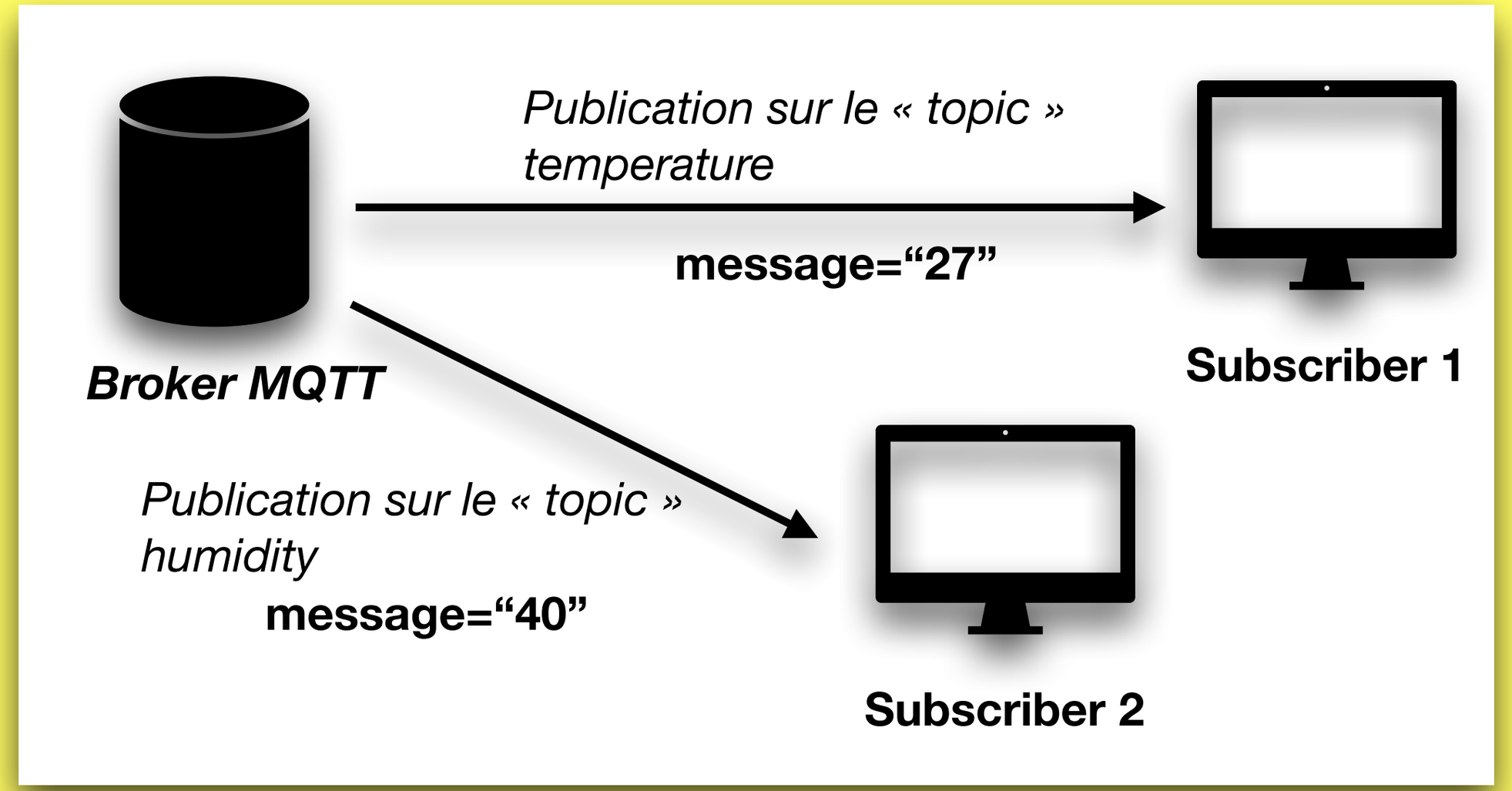


Etape 2:

- Le capteur 1 publie sur le topic “temperature” le message “27”
- Le capteur 2 publie sur le topic “humidity” le message « 40»



Etape 3
Le broker informe ces abonnés au topic “temperature” et “humidity”des messages reçus du publisher.



MQTT

Avec MQTT la QoS (qualité de service) est réglable.
Il existe trois niveaux de qualité de service:

QoS niveau 0: (mode par défaut)
L'information n'est transmise qu'une seule fois.
Le message n'est pas stocké par le broker.
Pas d'accusé de réception du souscripteur.
Le message est perdu en cas d'arrêt du broker ou du client.

QoS niveau 1
L'information est transmise au moins une fois.
Utilisation de séquences avec accusés de réception.

QoS niveau 2
Les informations seront reçues qu'une et unique fois.
Echange de deux paires de paquets, avec accusé
de réception par message.

Broker Mosquitto

Dans la suite, nous utiliserons le broker open source Mosquitto.

Broker Mosquitto

Publisher

mosquitto_pub
-h localhost
-d
-t temperature
-m 27

Fenêtre terminal

Fenêtre terminal

-h: adresse du broker
-d: mode debug
-t: topic
-m: message

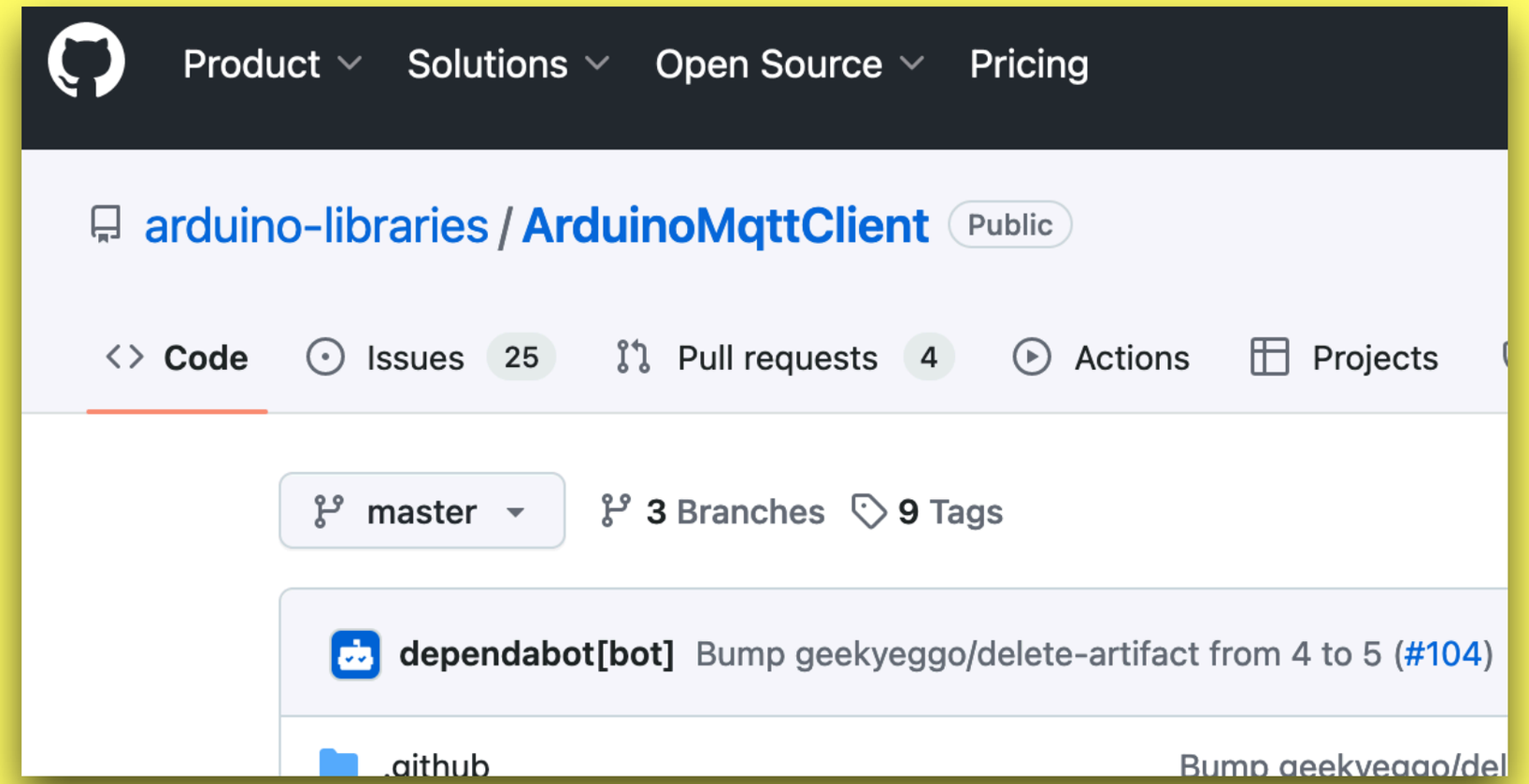
Subscriber

mosquitto_sub
-h localhost
-d
-t temperature

Fenêtre terminal

Bibliothèque Arduino

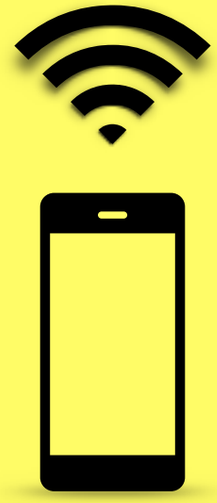
Pour utiliser le protocole MQTT en C/C++ sous Arduino, on peut utiliser la bibliothèque: **ArduinoMqttClient**



https://github.com/arduino-libraries/ArduinoMqttClient

Esp32 WiFi

Point Access



ssid: Nom du réseau

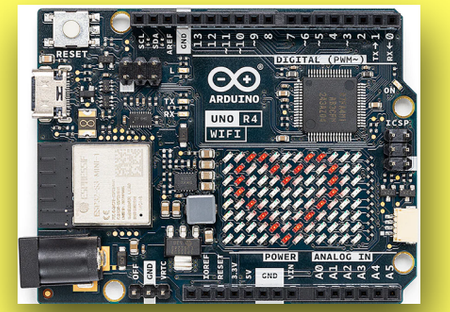
password: ******

Service DHCP*

Demande Connexion

Adresse IP Locale

Réseau WiFi



Mode Station

* Dynamic Host Configuration Protocol

Esp32 WiFi

Programme exemple pour connecter la carte en Wifi

https://docs.arduino.cc/tutorials/uno-r4-wifi/wifi-examples/

Home / Hardware / UNO R4 WiFi / **UNO R4 WiFi Network Examples**

UNO R4 WiFi Network Examples

Discover examples compatible with the WiFi library included in the UNO R4 Board Package.

Author • Jacob Hylén Last revision • 05/01/2024

The **Arduino UNO R4 WiFi** has a built in ESP32-S3 module that enables you to connect to Wi-Fi® networks, and perform network operations. Protocols including HTTPS, MQTT, UDP are tested and supported, and in this article, you will find a number of examples that will get you started.

Wi-Fi® support is enabled via the built-in `WiFiS3` library that is shipped with the **Arduino UNO R4 Board Package**. Installing the Board Package automatically installs the `WiFiS3` library.

> **Note:** The easiest way to connect your board to the Internet is via the **Arduino Cloud** platform. Here you can configure, program, monitor and synchronize your devices without having to write any networking code.

Esp32 WiFi


```
#include <WiFiS3.h>
```

Import bibliothèque UnoR4 WiFi (déjà présente dans la bibliothèque Arduino)

```
// const
const uint8_t LED_WIFI_PIN = 3;
```

Led pour visualizer l' état du WiFi

```
const char *ssid = “Linksys01370";
const char *pwd = "3fanq5w4pb";
```

Paramètres WiFi

```
// var global
bool isWifiConnection = false;
```

WiFi connecté ?

Esp32 WiFi

```
void connectWifi()
{
  uint8_t cmp = 0;
  int status = WL_IDLE_STATUS;
  //
  digitalWrite(LED_WIFI_PIN, 0);
  isWifiConnection = false;

  // check wifi module
  if (WiFi.status() == WL_NO_MODULE)
  {
    Serial.println("No Wifi module, impossible connection");
  }
  //

  Serial.println("Try wifi connection ");
  status = WiFi.begin(ssid, pwd);

  while (status != WL_CONNECTED)
  {
    Serial.print(".");

    delay(1000);
    cmp++;
    if (cmp > 15)
    {
      Serial.println("Failed Wifi connection !");
      isWifiConnection = false;
      return;
    }
  }
  //
  IPAddress ip = WiFi.localIP();
  Serial.println("IP adress:");
  Serial.println(ip);
  digitalWrite(LED_WIFI_PIN, 1);
  isWifiConnection = true;
  //
}
```

Pas de module WiFi !

Tentative de connexion

Adresse Local IP

Arduino MQTT


```
#include <ArduinoJson.h>

#include <ArduinoMqttClient.h>


// mqtt params
const char *MQTT_BROKER = “192.168.??.??”;
const int MQTT_PORT = 1883;
const char *mqtt_topic_led = “Unoxxx/led”;
const char *mqtt_topic_datas = “Unoxxx/datas”;

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

//
void mqttConnection()
{
  if (isWifiConnection == true)
  {
    mqttClient.setId(“uno4xxx");

    Serial.println("try to connect to the MQTT Broker”);

    //
    if (mqttClient.connect(MQTT_BROKER, MQTT_PORT) == false)
    {
      Serial.println("Failed MQTT Connection:");
      Serial.println(mqttClient.connectError());
      return;
    }
    //
    Serial.println("Connection MQTT OK");
    mqttClient.onMessage(mqttMessageHandler);
    mqttClient.subscribe(topic_led);
  }
  else
  {
    Serial.println("Error MQTT no wifi connection");
  }
}
```

Var global

Dans le setup !

Subscribe

Fonction de callback

Arduino MQTT exemple

Fonction de callback

```
void mqttMessageHandler(int messageSize)
{
  Serial.println("Received message");

  Serial.println(mqttClient.messageTopic());

   String msg = mqttClient.readString();

     //todo with msg
 }
```

Fonction pour transmettre message au broker

```
void mqttSendMessage()
{
  char thebuffer[200] = {'\0'};

   //todo create the json with ArduinoJson
   // place the result in thebuffer

   //send the msg
  mqttClient.beginMessage(topic_datas);
  mqttClient.print(thebuffer);
  mqttClient.endMessage();
}
```

Arduino MQTT exemple

setup

```
void setup()
{
  //
  //
  connectWifi();
  //
  if (isWifiConnection)
  {
    printMatrixText(WiFi.localIP().toString());

    mqttConnection();
  }
  else
  {
    printMatrixText("!!!!  No Wifi Connection");
  }
```

Affiche IP dans la matrice

loop

```
//loop

void loop()
{
  //
  if (isWifiConnection)
  {
    mqttClient.poll();
  }
  //
  ticker.update();
}
```

En attente de message du broker

InfluxDB

La collecte de données de mesure scientifique ou technique au moyen de capteur, génère d'énorme quantité de données dans un court laps de temps.

Ces données doivent faire l'objet d'un horodatage et être traités.
Ces données chronologiques exigent des bases de données spéciales.

Dans la suite nous présentons influxDB, un système de gestion de base de données spécialement conçu pour cette tache.

InfluxDB

- InfluxDB est un système de gestion de base de donnée développée par la société influxData Inc. C'est un logiciel open source et peut être utilisé gratuitement.

- La version 2 possède une interface utilisateur Web (dashboard) pour suivre et visualiser les données.

- Le système de gestion de base de donnée influxDB, utilise le langage de programmation GO de Google.

InfluxDB

InfluxDB est conçu pour générer les bases de donnée de type TSDB (Time Series Database), qui utilisent des séries temporelles.

Dans influxDB, une base de donnée peut être très compact et doit comprendre 2 ou 3 colonnes seulement.

Capteur	Valeur	Temps
Capteur **1**	145.2	23/04/2023@9:06
Capteur **2**	50.5	23/04/2023@9:45

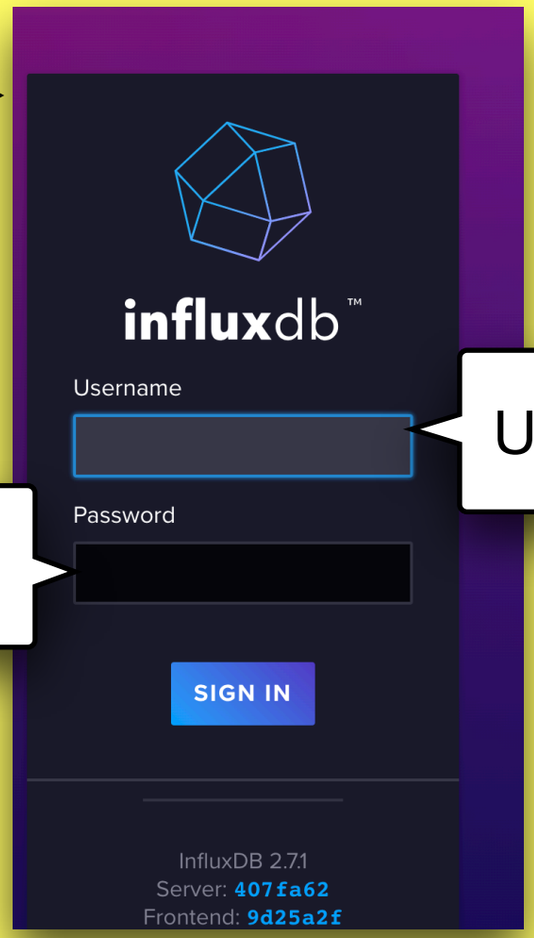
InfluxDB

La base influxDB est installée en local sur la carte Raspberry Pi.

Ouvrir un navigateur Web et saisir l'URL suivante:

localhost:8086

Fenêtre de connexion

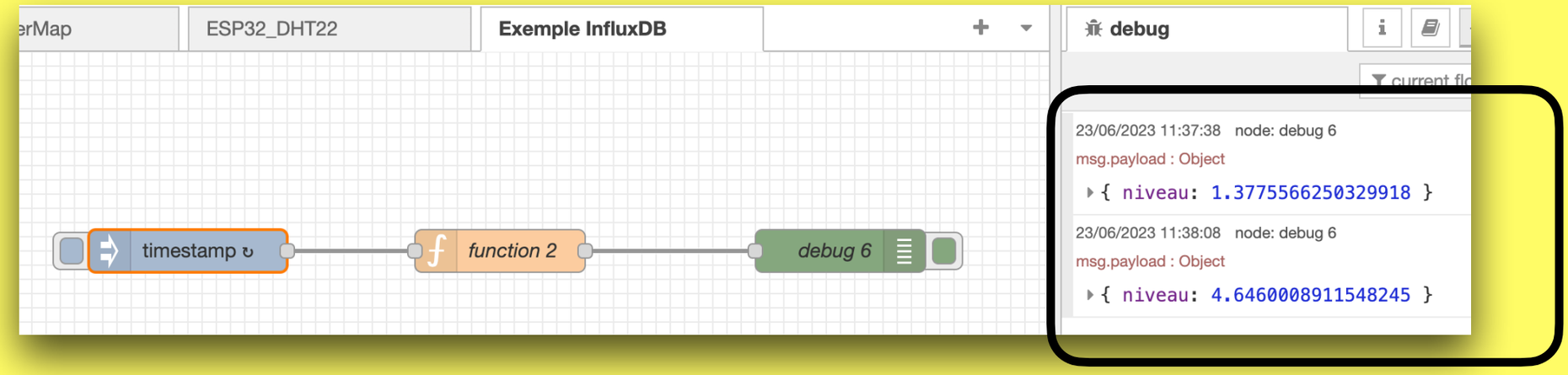


Username: **pi**

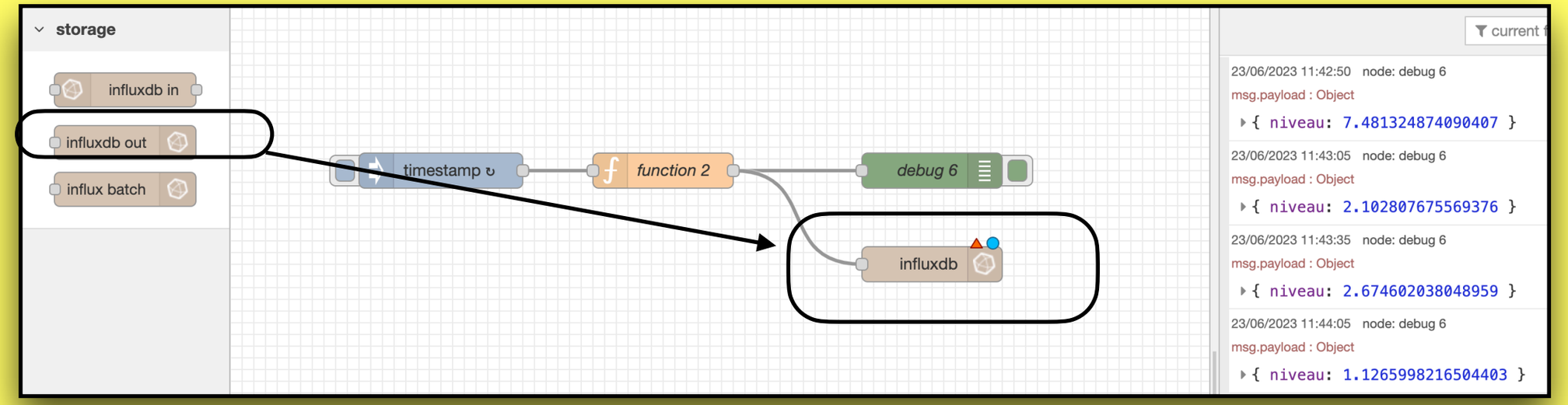
Pwd: **raspberrypi**

InfluxDB exemple

Soit un capteur de niveau sous Node Red.

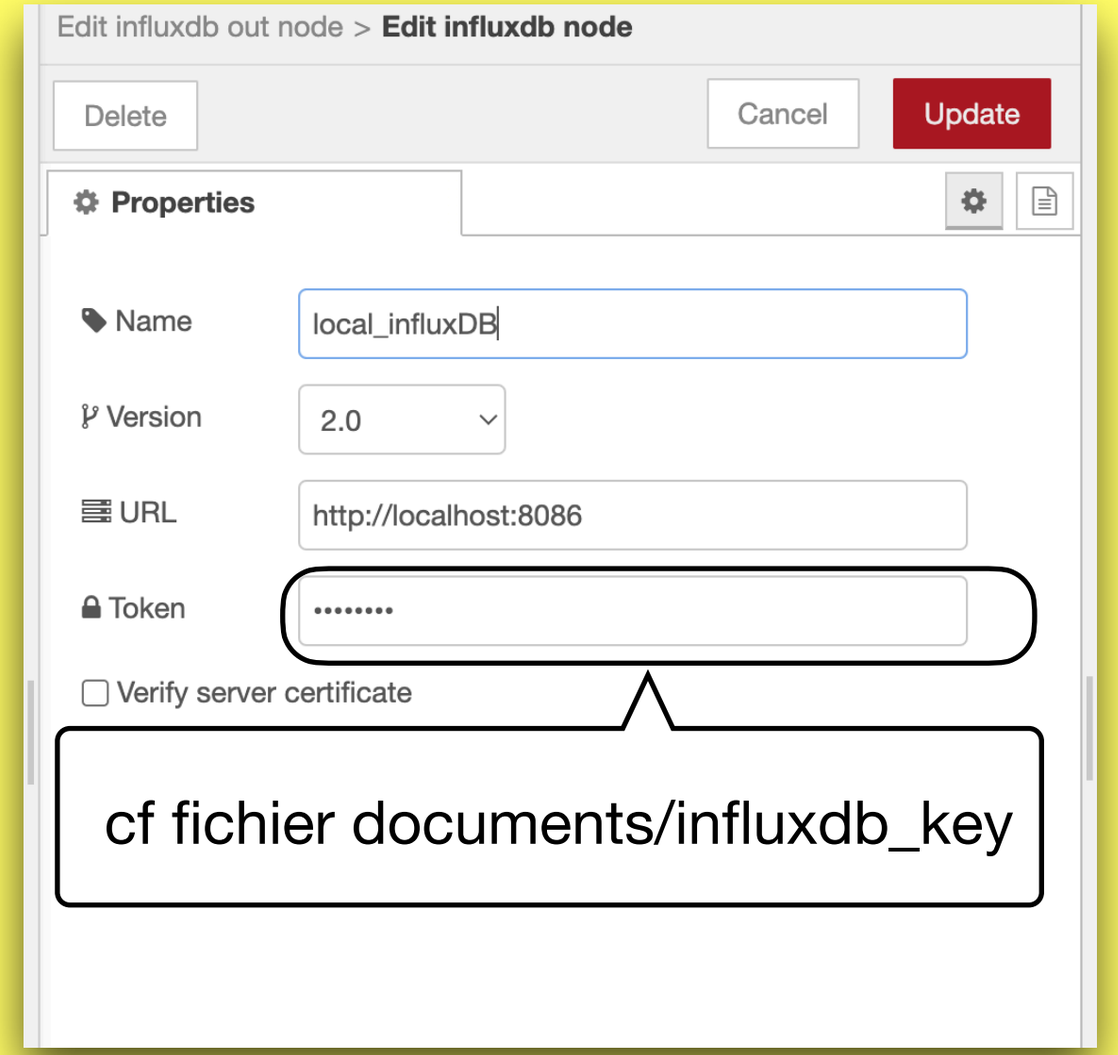


InfluxDB exemple



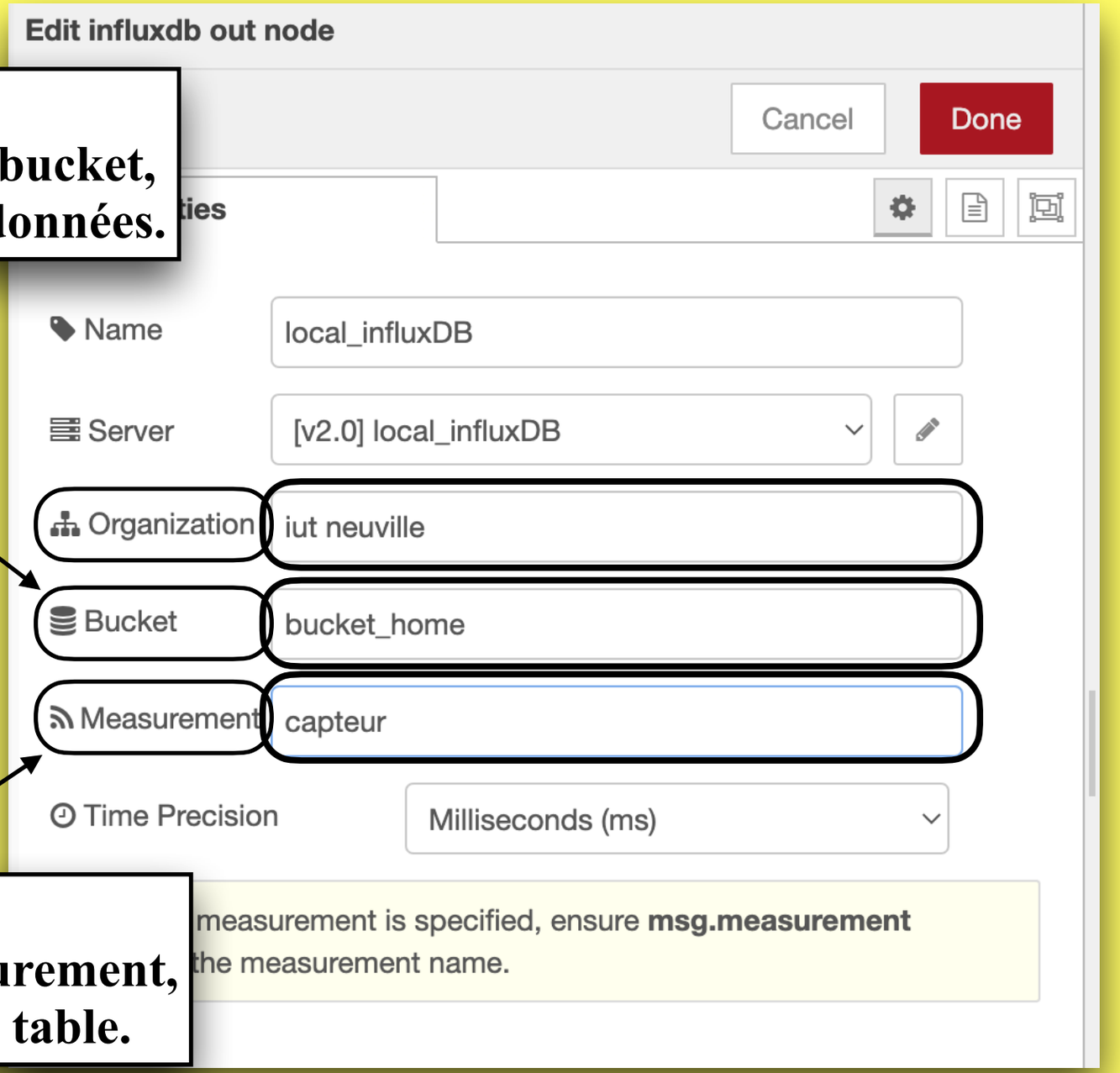
Sous Node-Red il existe une bibliothèque pour la gestion influxDB.

InfluxDB exemple



InfluxDB exemple

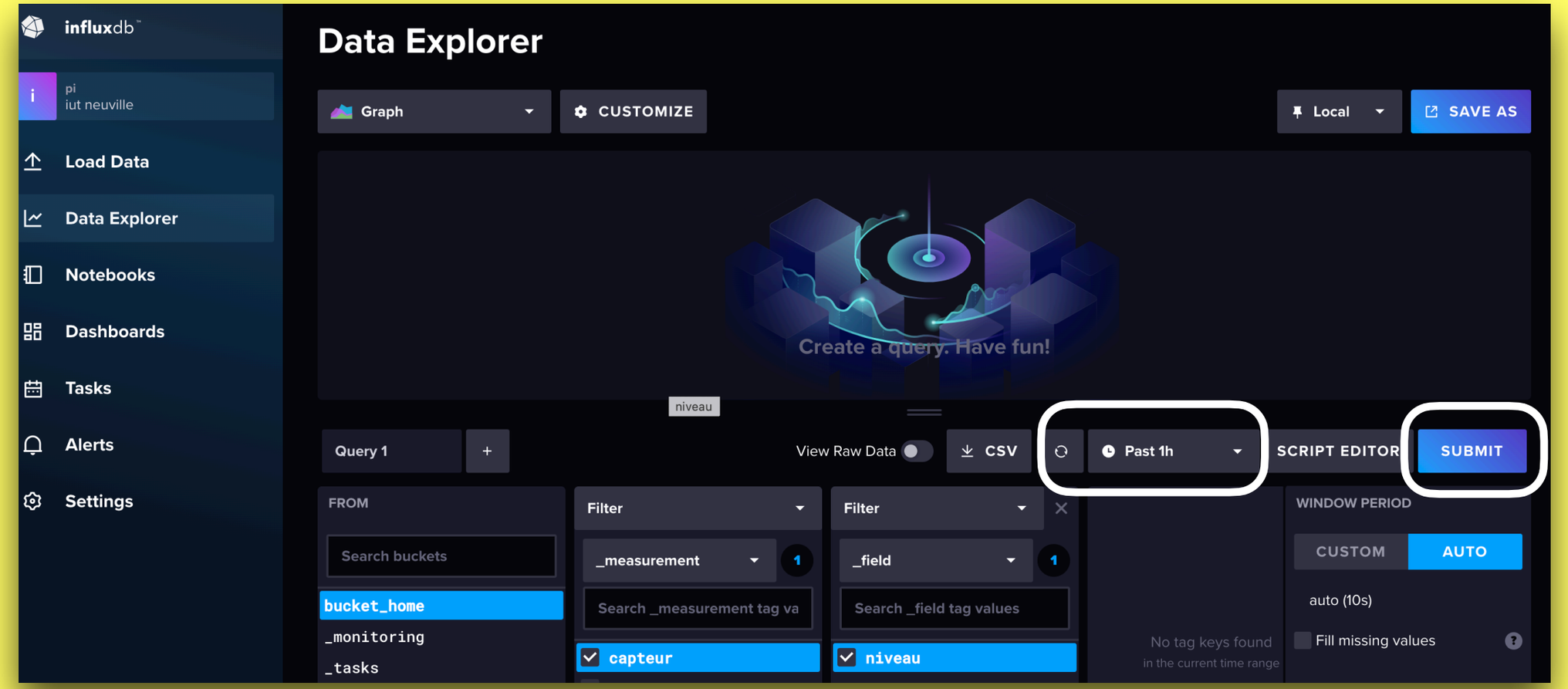
influxDB
Utilise la notion de bucket, afin de stocker les données.



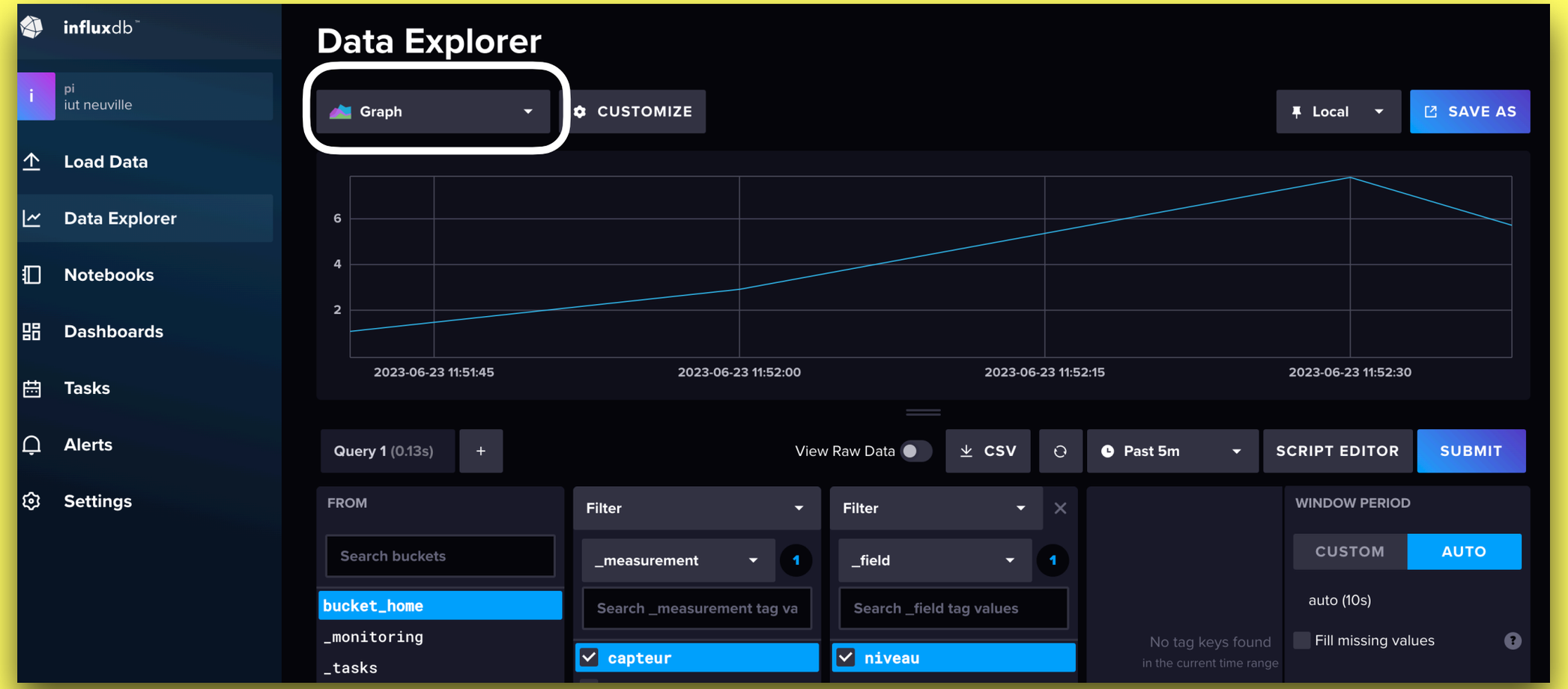
influxDB
Utilise la notion de Measurement, qui est l'équivalent d'une table.

IUT
CERGY-PONTOISE

InfluxDB exemple

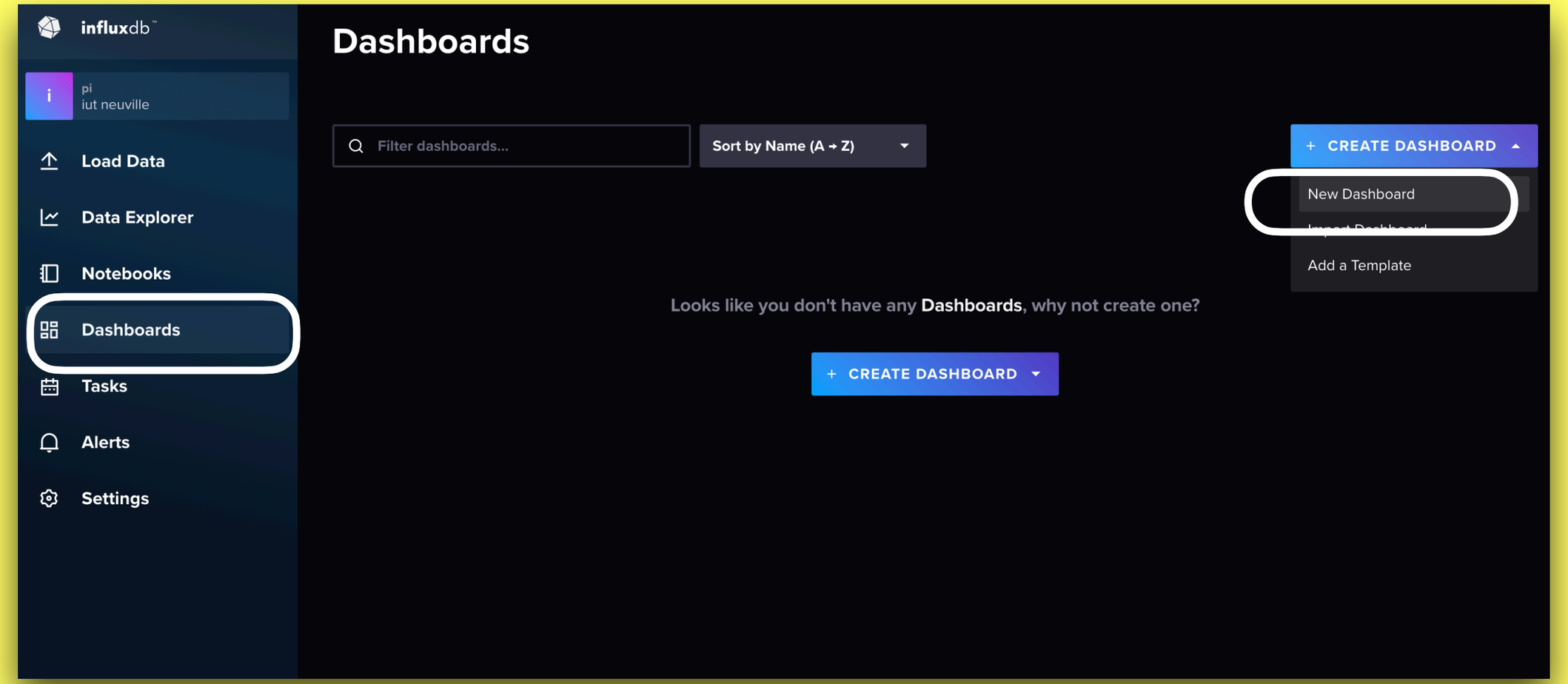


InfluxDB exemple

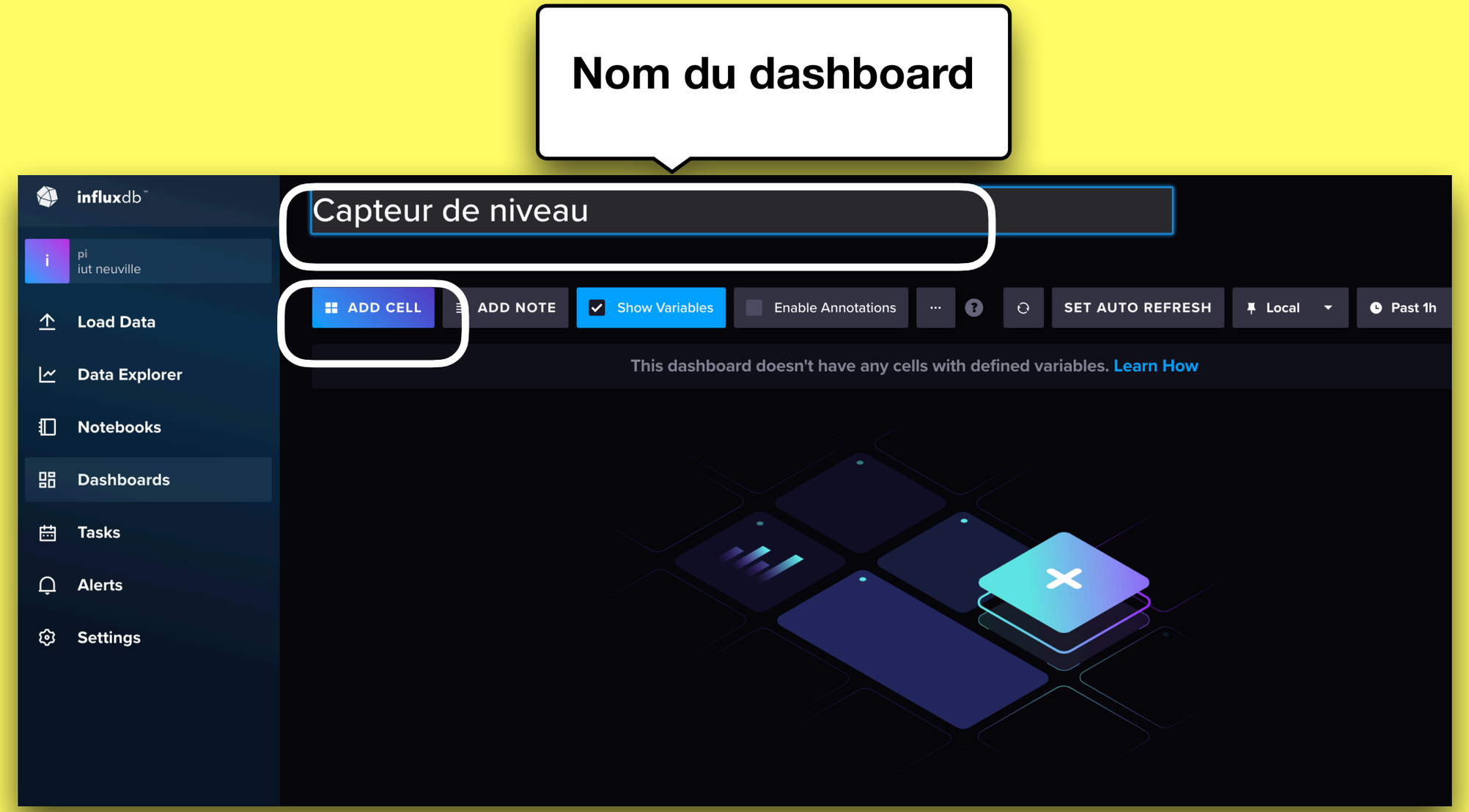


IUT
CERGY-PONTOISE

InfluxDB Dashboard



InfluxDB Dashboard



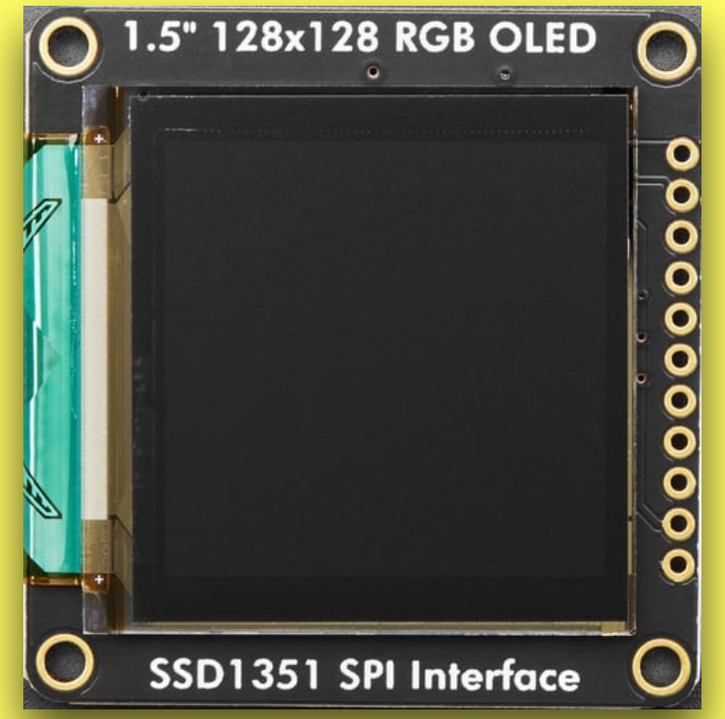
InfluxDB Dashboard

IUT
CERGY-PONTOISE

InfluxDB Dashboard

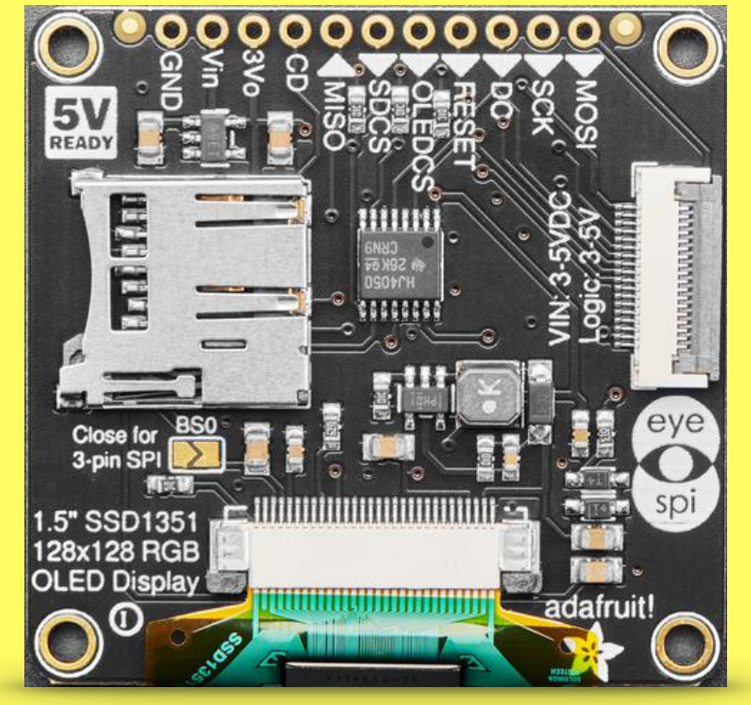
Ecran Oled Couleur

Soit un afficheur OLED couleur (format rgb(565)). Ce dernier possède une diagonale avec une résolution de 128x128 pixels.



Board Technical Details

- 1.5" diagonal OLED, 16-bit color
- SPI interface
- 3.3-5V logic and power
- Micro-SD card holder
- Dimensions: 43.17mm / 1.7" x 42mm / 1.65" x 5.42mm / 0.2"



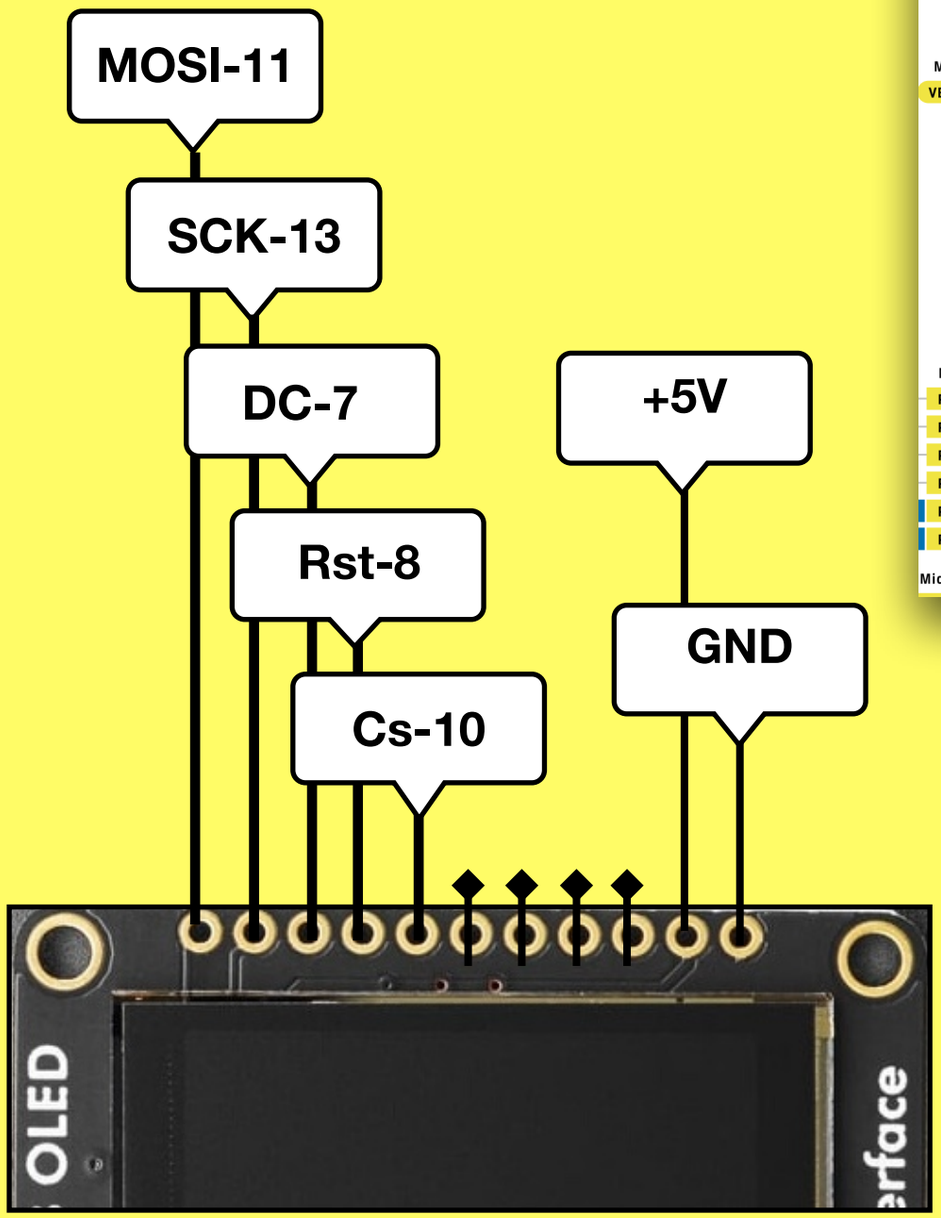
https://learn.adafruit.com/adafruit-1-5-color-oled-breakout-board/overview

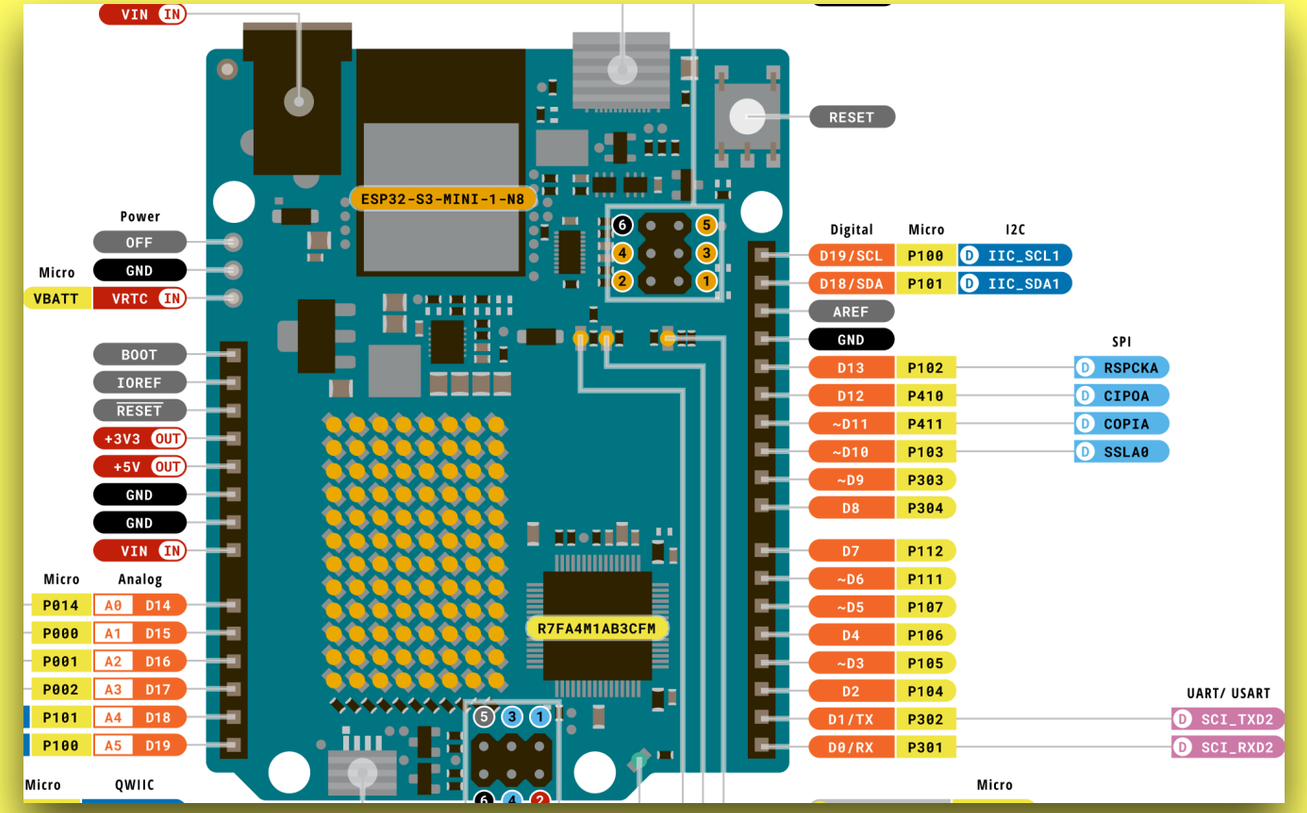
Breakout Pins

This color display uses SPI to receive image data. That means you need at least 4 pins - clock, data in, oled cs and d/c. If you'd like to have SD card usage too, add another 2 pins - data out and card cs. However, there's a couple other pins you may want to use, lets go thru them all!

- Lite - this is the PWM input for the backlight control. Connect to 3-5VDC to turn on the backlight. Connect to ground to turn it off. Or, you can PWM at any frequency.
- MISO - this is the SPI Microcontroller In Serial Out pin, its used for the SD card. It isn't used for the OLED display which is write-only
- SCLK - this is the SPI clock input pin
- MOSI - this is the SPI Microcontroller Out Serial In pin, it is used to send data from the microcontroller to the SD card and/or OLED
- OLEDCS - this is the OLED SPI chip select pin
- SDCS - this is the SD card chip select, used if you want to read from the SD card.
- DC - this is the OLED SPI data or command selector pin
- RESET - this is the OLED reset pin. Connect to ground to reset the TFT! Its best to have this pin controlled by the library so the display is reset cleanly, but you can also connect it to the Arduino Reset pin, which works for most cases.
- CD - this is the SD card detect pin.
- 3Vo - this is the 3V output from the onboard voltage regulator
- Vin - this is the power pin, connect to 3-5VDC - it has reverse polarity protection but try to wire it right!
- GND - this is the power and signal ground pin

Ecran Oled Couleur





Ecran Oled Couleur

L'afficheur utilise le pilote graphique SSD1351.

Pour simplifier la programmation d'un tel afficheur, on utilisera une bibliothèque développer par Mr Bodmer

→ **TFT_eSPI**

https://github.com/Bodmer/TFT_eSPI

Documentations:

https://doc-tft-espi.readthedocs.io/tft_espi/

Ecran Oled Couleur

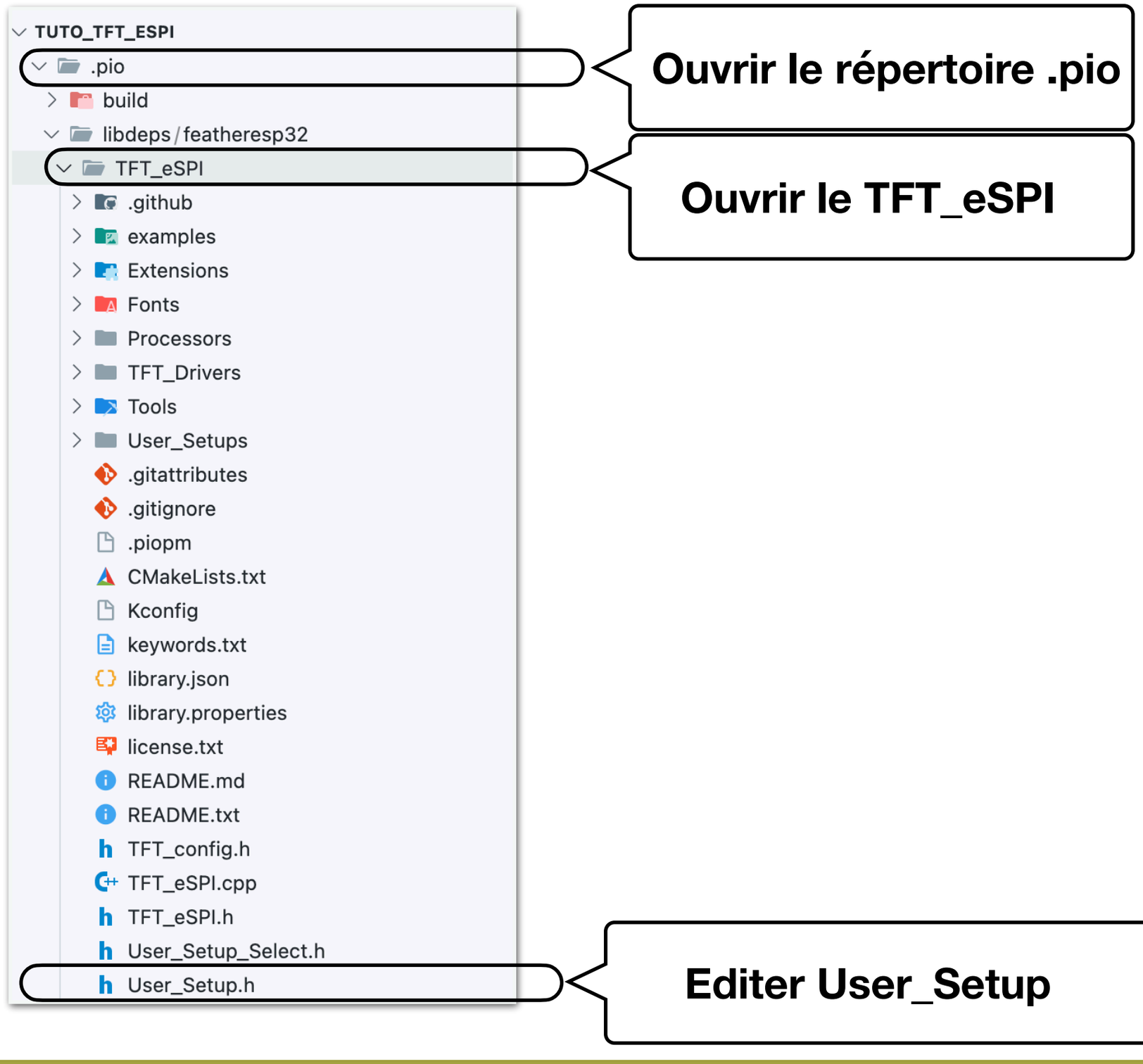
Voici les différentes étapes pour utiliser cette bibliothèque sous Visual Studio Code.

Fichier platformio.ini

```
[env:featheresp32]
platform = espressif32
board = featheresp32
framework = arduino
monitor_speed=115200
lib_deps =
  sstaub/Ticker @ ^4.4.0
  bodmer/TFT_eSPI @ ^2.5.43
```

Installer la bibliothèque TFT_eSPI

Ecran Oled Couleur



Choisir le driver SSD1351

```
// Display type -  only define if RPi display
// #define RPI_DISPLAY_TYPE // 20MHz maximum SPI

// Only define one driver, the other ones must be commented out
// #define ILI9341_DRIVER       // Generic driver for common displays
// #define ILI9341_2_DRIVER     // Alternative ILI9341 driver, see https://I/issues/1172
// #define ST7735_DRIVER      // Define additional parameters below for this display
// #define ILI9163_DRIVER     // Define additional parameters below for this display
// #define S6D02A1_DRIVER
// #define RPI_ILI9486_DRIVER // 20MHz maximum SPI
// #define HX8357D_DRIVER
// #define ILI9481_DRIVER
// #define ILI9486_DRIVER
// #define ILI9488_DRIVER
// #define ST7789_DRIVER
// #define ST7789_2_DRIVER
// #define R61581_DRIVER
// #define RM68140_DRIVER
// #define ST7796_DRIVER
#define SSD1351_DRIVER
// #define SSD1963_480_DRIVER
// #define SSD1963_800_DRIVER
// #define SSD1963_800ALT_DRIVER
// #define ILI9225_DRIVER
// #define GC9A01_DRIVER
```

Ecran Oled Couleur

Mettre en commentaire toutes les définitions.
Définir les broches utilisées par le micro contrôleur.

```
// tested with GC9A01 display)
// The hardware SPI can be mapped to any pins

#define TFT_MOSI 11 // In some display driver board, it might be written as "SDA" and so on.
#define TFT_SCLK 13
#define TFT_CS 10  // Chip select control pin
#define TFT_DC 7  // Data Command control pin
#define TFT_RST 8 // Reset pin (could connect to Arduino RESET pin)

#define TFT_BL -1  // LED back-light

#define TOUCH_CS -1 // Chip select pin (T_CS) of touch screen
```

Saisir les broches

Ecran Oled Couleur


```
// ##################################################################
//
// Section 3. Define the fonts that are to be used here
//
// ##################################################################

// Comment out the #defines below with // to stop that font being lo
// The ESP8366 and ESP32 have plenty of memory so commenting out fonts is
// normally necessary. If all fonts are loaded the extra FLASH space required
// about 17Kbytes. To save FLASH space only enable the fonts you need!

#define LOAD_GLCD  // Font 1. Original Adafruit 8 pixel font needs ~1820 bytes in FLASH
#define LOAD_FONT2 // Font 2. Small 16 pixel high font, needs ~3534 bytes in FLASH, 96 characters
#define LOAD_FONT4 // Font 4. Medium 26 pixel high font, needs ~5848 bytes in FLASH, 96 characters
                   // #define LOAD_FONT6 // Font 6. Large 48 pixel font, needs ~2666 bytes in FLASH,
// #define LOAD_FONT7 // Font 7. 7 segment 48 pixel font,
//  #define LOAD_FONT8 // Font 8. Large 75 pixel font needs ~3256 bytes
//  #define LOAD_FONT8N // Font 8. Alternative to Font 8 above,
//  #define LOAD_GFXFF // FreeFonts. Include access to the 48 Adafruit_GFX free

// Comment out the #define below to stop the SPIFFS filing system and smooth font code being loaded
// this will save ~20kbytes of FLASH
// #define SMOOTH_FONT
```

Saisir les “fonts”

Ecran Oled Couleur


```cpp
#include <Arduino.h>
#include <Ticker.h>
#include <SPI.h>
#include <TFT_eSPI.h>

// const
const uint32_t PERIOD = 1000;
const int16_t SCREEN_WIDTH = 128;
const int16_t SCREEN_HEIGHT = 128;



// prot
void action();

// obj
Ticker mticker(action, PERIOD, 0, MILLIS);

TFT_eSPI tft = TFT_eSPI(SCREEN_WIDTH, SCREEN_HEIGHT);

TFT_eSprite canvas = TFT_eSprite(&tft);
```

Ecran Oled Couleur

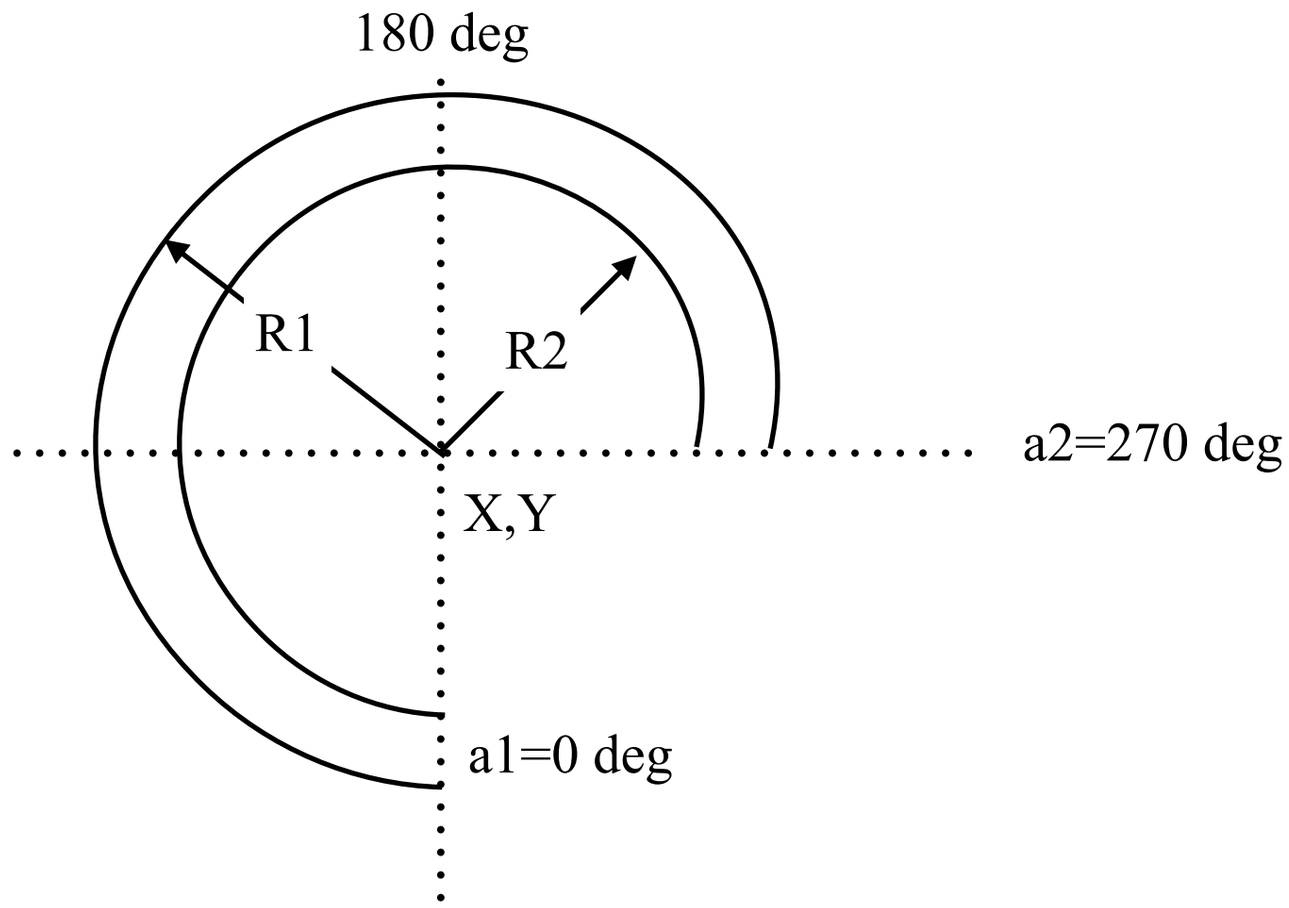

```
void setup()
{
  Serial.begin(115200);
  //
  tft.init();
  //
  canvas.setSwapBytes(true);
  canvas.setColorDepth(8);
  canvas.createSprite(SCREEN_WIDTH, SCREEN_HEIGHT);

  //
  delay(500);
  //
  mticker.start();
}
```

Format couleur 8bits

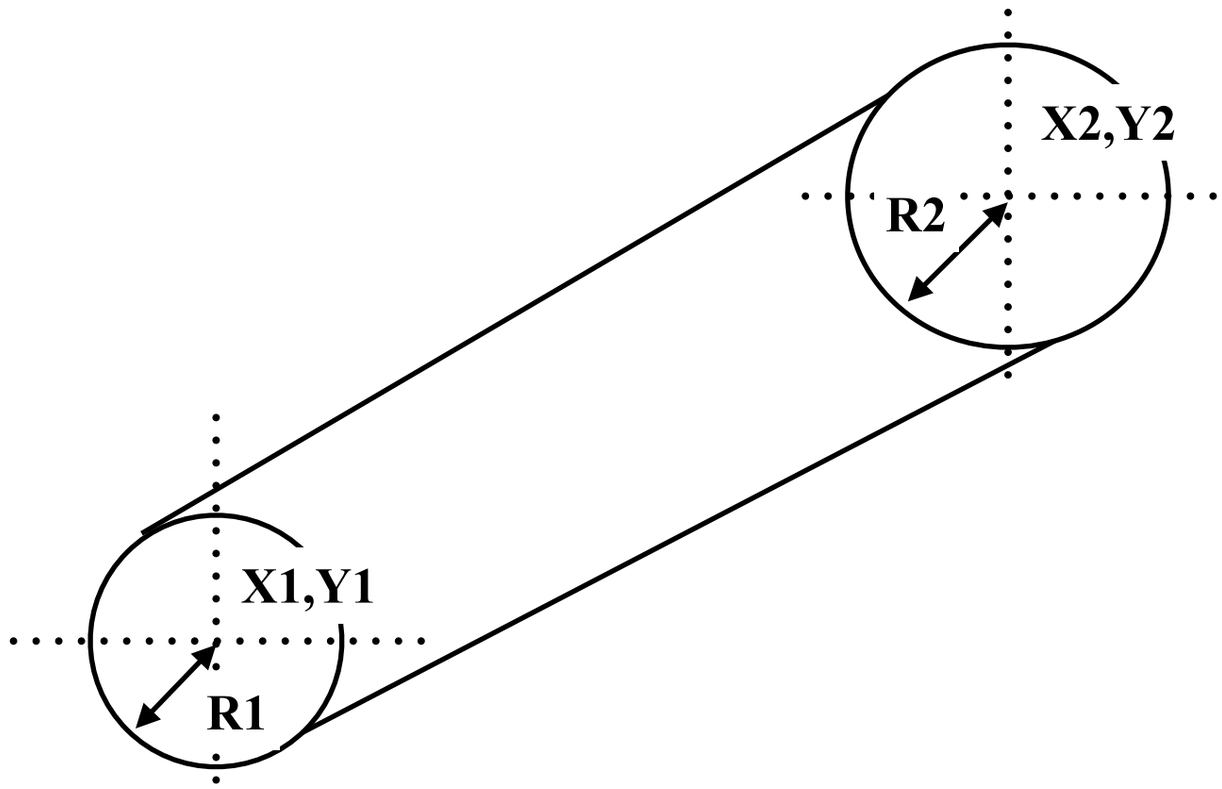
```
void action()
{
  canvas.fillScreen(TFT_BLUE);
  //
  canvas.setTextDatum(0);
  canvas.setTextColor(TFT_YELLOW);
  canvas.drawString("Uno", 0, 0, 2);
  //
  canvas.fillRect(SCREEN_WIDTH / 2, SCREEN_HEIGHT / 2, 50, 50, TFT_RED);
  //
  //canvas.pushImage(20, SCREEN_HEIGHT / 2, 128, 64, bitmap_IUT);

  //mise a jour de la visualisation
  canvas.pushSprite(0, 0);
}
```

font

drawSmoothArc(int X, int Y, int R1, int R2, int a1, int a2, short arcColor, short bgcolor)



drawWedgeLine(int X1, int Y1, int X2, int Y2, int R1, int R2, short Color, short bgcolor)



Ecran Oled Couleur

Exemple d'écran graphique