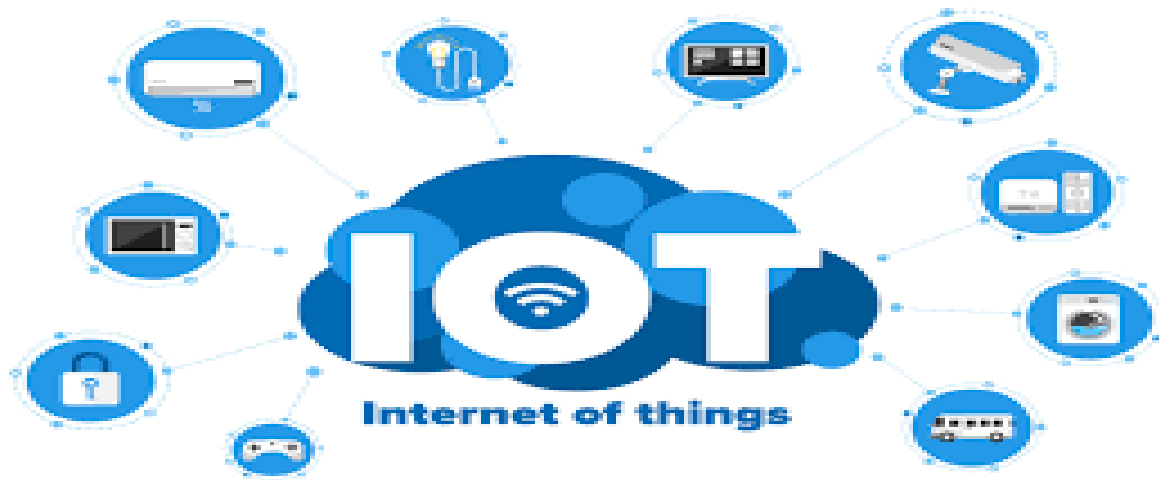


RAPPORT DE TRAVAUX PRATIQUES IOT

MASTER : M2SI



Réaliser par :

AHMED AHDIOUAD

HANANE TASMANT

NAIMA SALYM

Encadrer par :

M.KEBBAJ NABIL

Partie 1 : ThinkSpeak

Objectif :

Dans ce projet nous allons voir comment envoyer des données d'un ESP32 vers la plateforme d'IOT THINKSPEAK.

1-ThinkSpeak :

ThingSpeak est une plateforme riche en possibilités et en fonctionnalités. Vous pourrez vous amuser à faire de l'analyse de données, à suivre votre consommation d'énergie, à compter les voitures qui passent dans votre rue, à prédire le débordement de la rivière où à construire une station météo. Mais avant de pouvoir accéder à toutes ces fonctionnalités avancées, il va falloir commencer par les bases. Et pour ce faire on a réalisé un projet tout simple, qui va nous permettre de ressortir du tiroir un de nos ESP32 et de le programmer pour qu'il nous permette de suivre les données de la température et d'humidité sur une courbe ThingSpeak.

2-Les étapes utilisées pour créer un compte :

a-Créer un compte

La première chose à faire pour utiliser les services de la plateforme ThingSpeak va être de créer un compte. L'accès à la plateforme nécessite un compte Math Works.

- Allez sur le site ThingSpeak et cliquez sur **<Sign In>** puis cliquez en bas sur le texte "[Sign up for the first time](https://thingspeak.com/users/sign_up)" ou directement sur [CE lien](https://thingspeak.com/users/sign_up) (https://thingspeak.com/users/sign_up).
- Remplissez les champs qui sont demandés.

connectez-vous à ThingSpeak en utilisant l'adresse e-mail associée à votre université ou organisation.

Pour envoyer des données plus rapidement à ThingSpeak ou pour envoyer plus de données depuis plus d'appareils, envisagez les [options de licence payante](#) pour une utilisation commerciale, universitaire, domestique et étudiante.

Create MathWorks Account

Email Address

hanantasmant@gmail.com

i To access your organization's MATLAB license, use your school or work email.

Location

Morocco

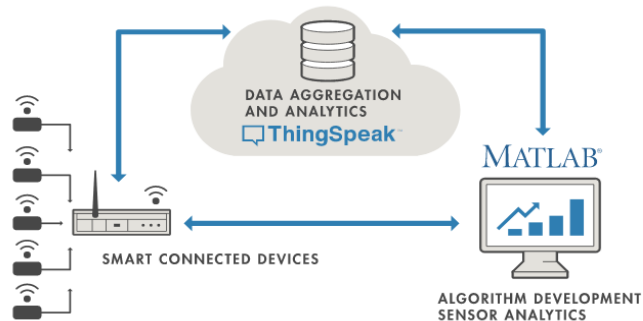
First Name

hanane

Last Name

tasmant

Continue



Ensuite aller dans messagerie, vous devriez trouver le mail de MathLab, dans lequel il y a un lien sur lequel il faudra cliquer et valider l'adresse mail et de ce fait votre inscription.

Lorsque cette étape d'inscription est réalisée, revenir sur le site ThingSpeak, cliquer sur **<Sign In>** et cette fois vous pouvez vous identifier avec le compte que vous venez de créer:



Important MathWorks Account Information

Thank you for registering with MathWorks!

Verify your email address by clicking this link:

[Verify your email](#)

If you are unable to click the link, copy and paste this link into the address bar on your browser.

<https://www.mathworks.com/mwaccount/widgets/embedded/profiles/verify/f9aac77a-fef0-4678-9b6b-f74662d65036>

Sincerely,

MathWorks Customer Service Team

[Canaux](#)
[applications](#)
[Soutien](#)

[Un usage commercial](#)
[Comment acheter](#)

connectez-vous à ThingSpeak en utilisant l'adresse e-mail associée à votre université ou organisation.

Pour envoyer des données plus rapidement à ThingSpeak ou pour envoyer plus de données depuis plus d'appareils, envisagez les [options de licence payante](#) pour une utilisation commerciale, universitaire, domestique et étudiante.

Email

No account? [Create one!](#)
By signing in, you agree to our [privacy policy](#).

Next

b-Créer un Channel

Une fois connecté, nous allons devoir créer ce que ThingSpeak appelle un "Channel". Cela représente un ensemble de données regroupées entre elles et qui proviennent d'un objet connecté, d'un autre "Channel" ou d'un service web. Un Channel est composé de champs ("Field" en Anglais). Il peut y en avoir jusqu'à 8. Et chaque champ représente une donnée.

Pour continuer sur notre exemple de sonde de température et d'humidité, nous avons créé le Channel DHT22 qui représentera le capteur de température et humidité DHT22 dans notre installation.

Le capteur DHT11 permet d'avoir la température mais aussi le pourcentage d'humidité de l'air. Donc tout naturellement, dans notre Channel nous allons créer deux champs:

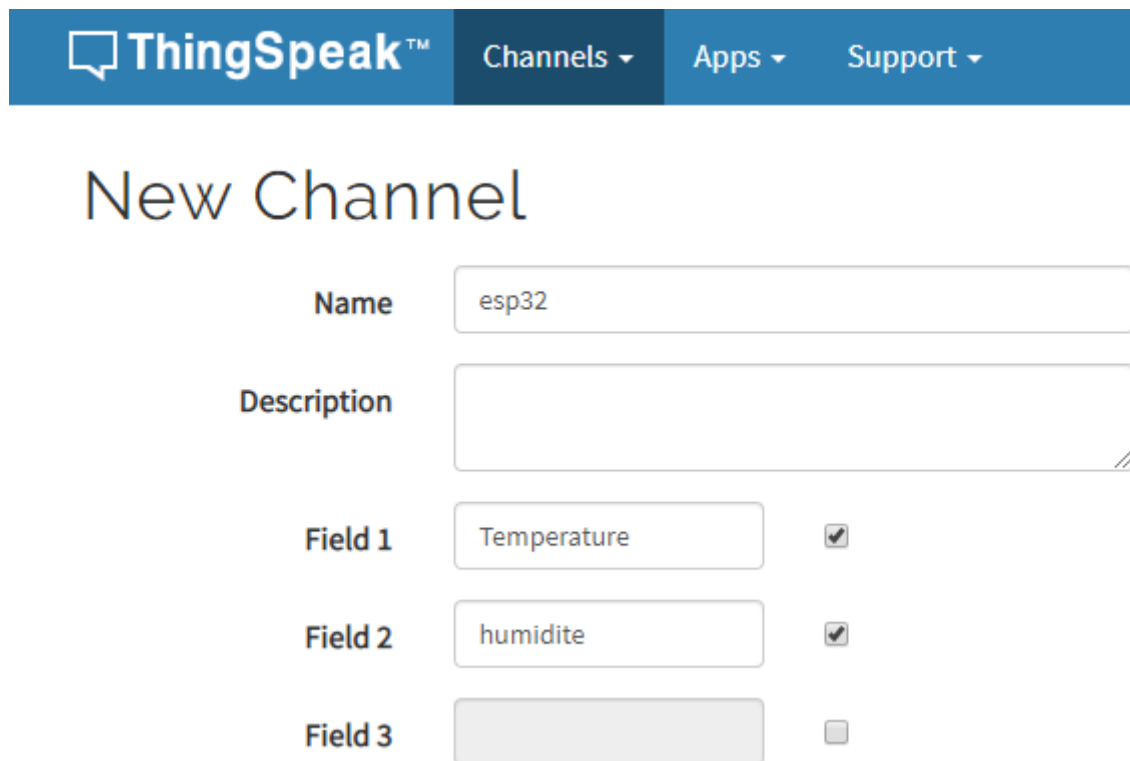
1. **field1** = Température
2. **field2** = Humidité

- Pour ce faire :

-Cliquez sur [<<New Channel>>](#)

New Channel

- Remplissez les premiers champs: Name, Description, **Field 1** et **Field 2**. Tous les autres ne seront pas utiles.



-Et validez avec <<**Save Channel**>>

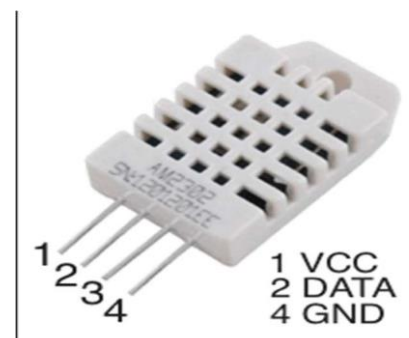
Save Channel

2-DHT22

a-Définition du capteur DHT22

L'AM2302 est une version du DHT22 avec des fils, contenu dans un corps en plastique. C'est un capteur de température et d'humidité de base. Il utilise un capteur d'humidité capacitif et une thermistance pour mesurer l'air environnant et produire un signal digital sur une broche/pin data. Ce capteur ne nécessite aucune entrée analogique.

Il est très simple à utiliser mais nécessite néanmoins une gestion précise du temps pour obtenir l'information. Le seul réel inconvénient de ce capteur c'est que l'on obtient une nouvelle mesure que toutes les deux secondes. Donc, en utilisant la bibliothèque, la valeur lue peut être vieille de deux secondes.



b-Caractéristiques techniques:

- ✚ Tension d'alimentation: 3 à 6V DC
- ✚ Courant de fonctionnement: < 2.5mA
- ✚ Plage de température: -40 à +80°C
- ✚ Précision de température: $\pm 0.5^{\circ}\text{C}$
- ✚ Humidité: de 0 à 100% RH
- ✚ Précision d'humidité : $\pm 2\%$ HR
- ✚ Temps de réponse: <1s
- ✚ Dimension: 15.1*25*7.7mm
- ✚ Type de connexion : 3 fils (Vcc, Signal, GND)
- ✚ broche 1) à gauche à la tension d'alimentation comprise entre 3 et 5V
- ✚ broche 2) à votre broche d'entrée de données (doit être connecté à une résistance de pull up, interne ou externe)
- ✚ broche 3) non connecté
- ✚ broche 4) Masse.

3-ESP32

a-Définition du ESP32

ESP32 est une série de microcontrôleurs de type système sur une puce (Soc) d'*Expressif Systems* , intégrant la gestion du Wifi et du Bluetooth en mode double, et un DSP. C'est une évolution d'ESP8266.

Son support Wifi et Bluetooth, en fait un système apprécié dans le domaine de l'internet des objets.

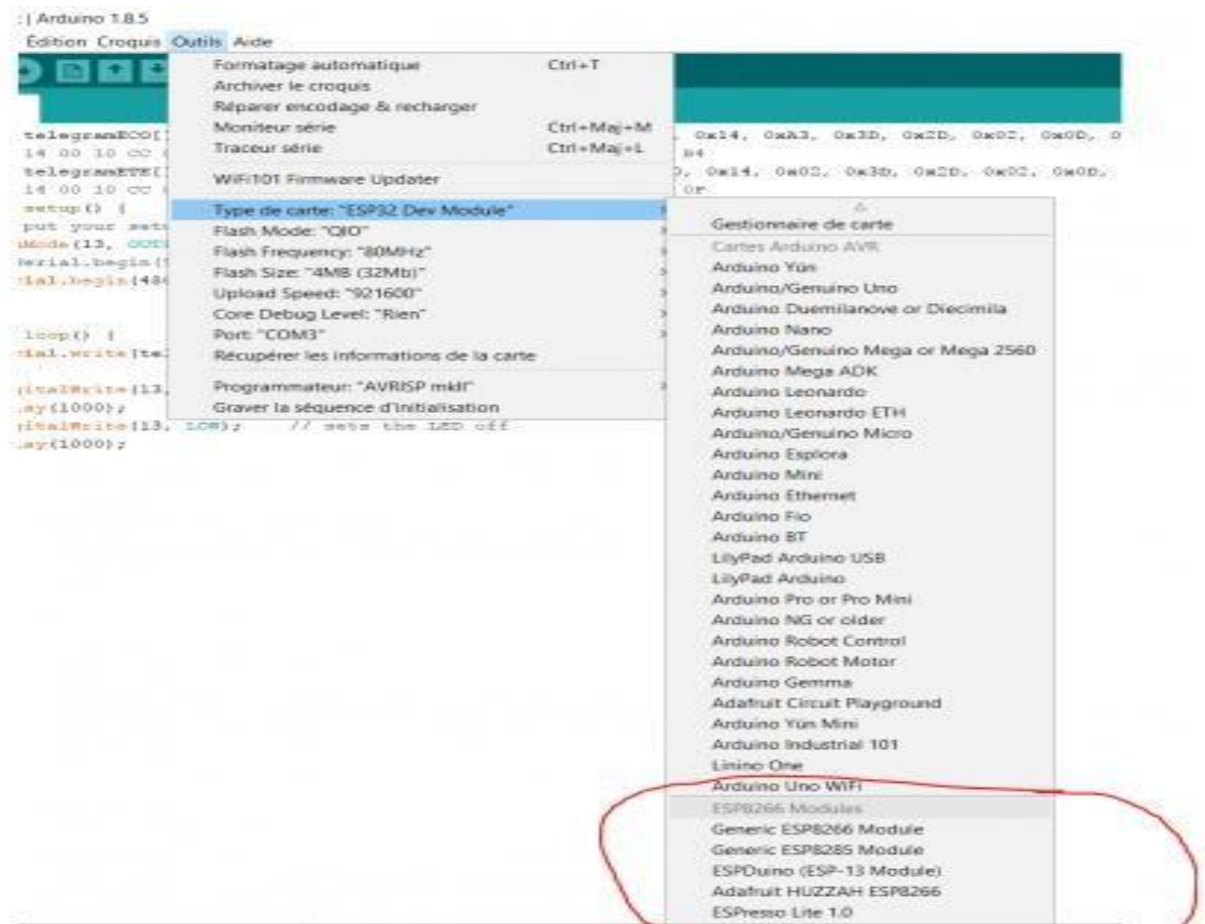
b-Caractéristiques techniques:

- CPU ESP-WROOM-32 (Tensilica Xtensa LX6)
- Voltage : 3.3V
- Flash : 4000 kB
- RAM : 520 kB
- EEPROM : 448 kB
- Clock speed : 240MHz
- WiFi : Yes
- Bluetooth : Yes
- SD : No



C-Installation de l'environnement de développement

- Démarrer l'environnement ARDUINO et vérifier que l'ESP32 apparait bien dans les types de carte :

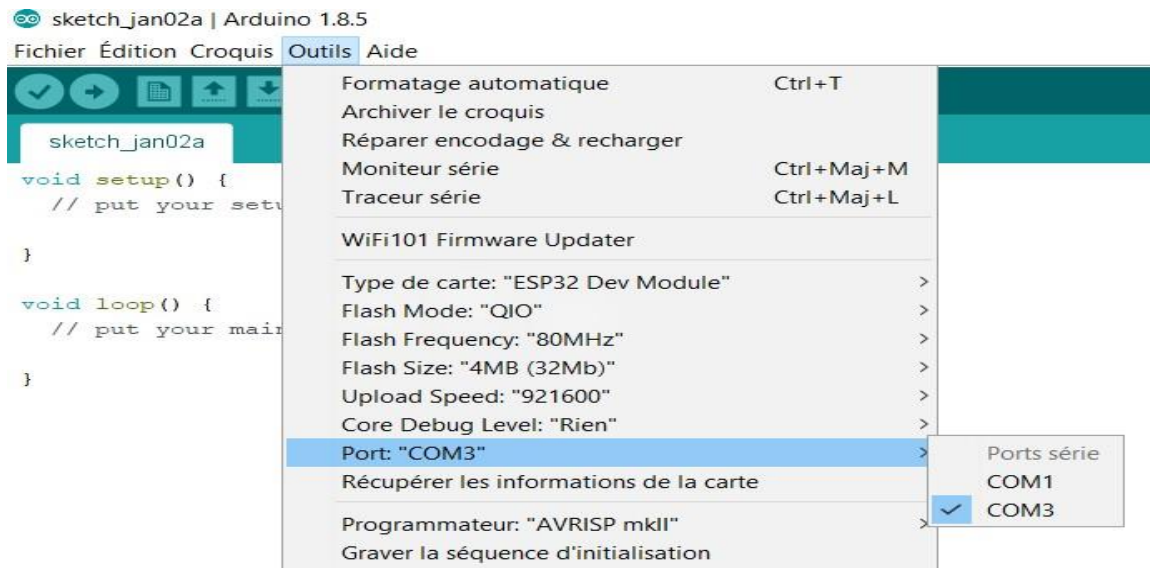


esp32_cp210x.png

- > Périphériques système
- > Ports (COM et LPT)
 - Port de communication (COM1)
 - Silicon Labs CP210x USB to UART Bridge (COM3)
- > Processeurs
- > Souris et autres périphériques de pointage

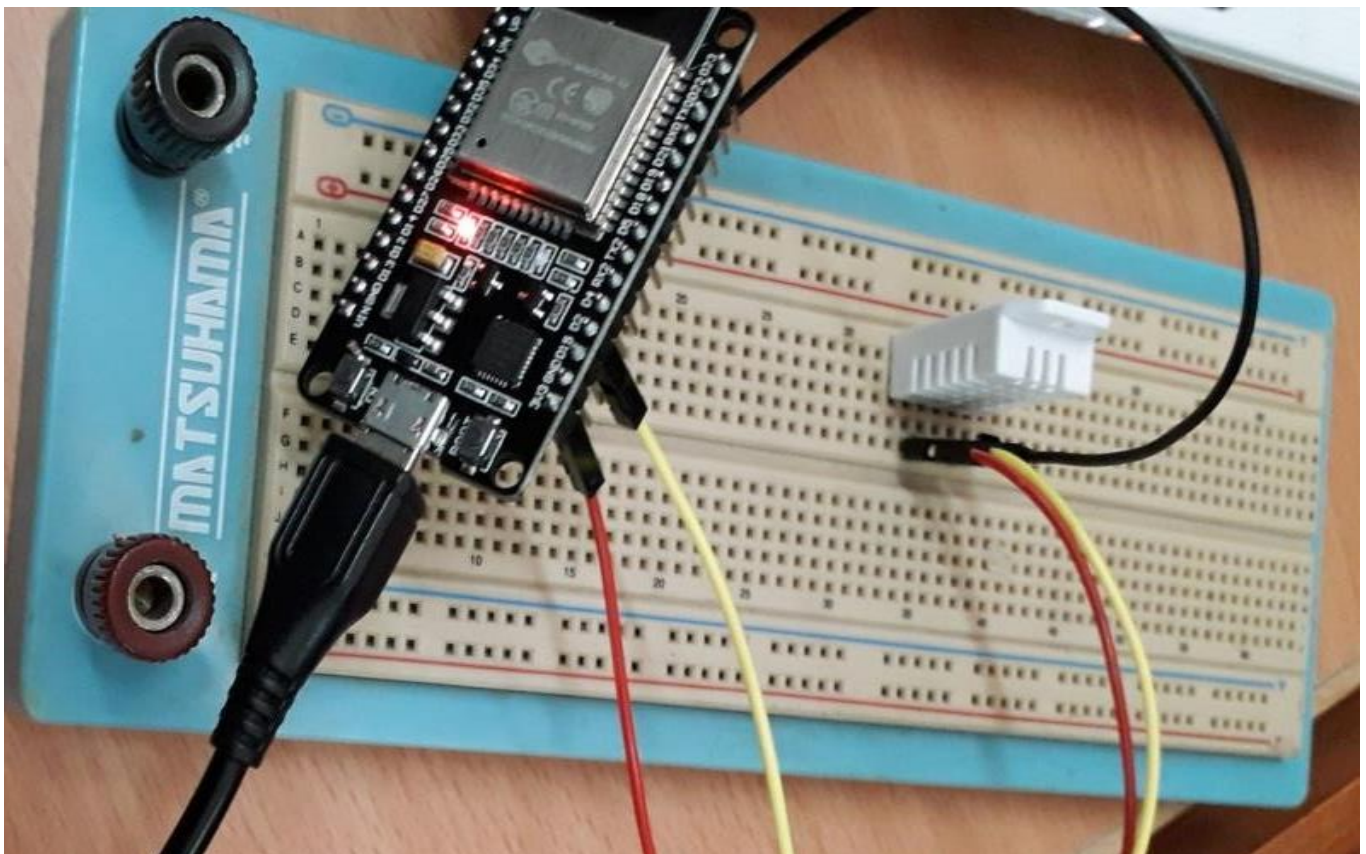
-Brancher carte ESP32, vérifier le port COM qui lui a été attribué :

Créer un nouveau sketch, sélectionner la carte: ESP32 Dev Module et le port qui lui a été associé



4-Câblage de matériel :

On a connecte alimentation 3-5V sur le fil rouge, Le fil jaune correspond à la broche data(donnée) et le fil noir à la masse (GND/Ground).



5-Programmation

a-Pré requis

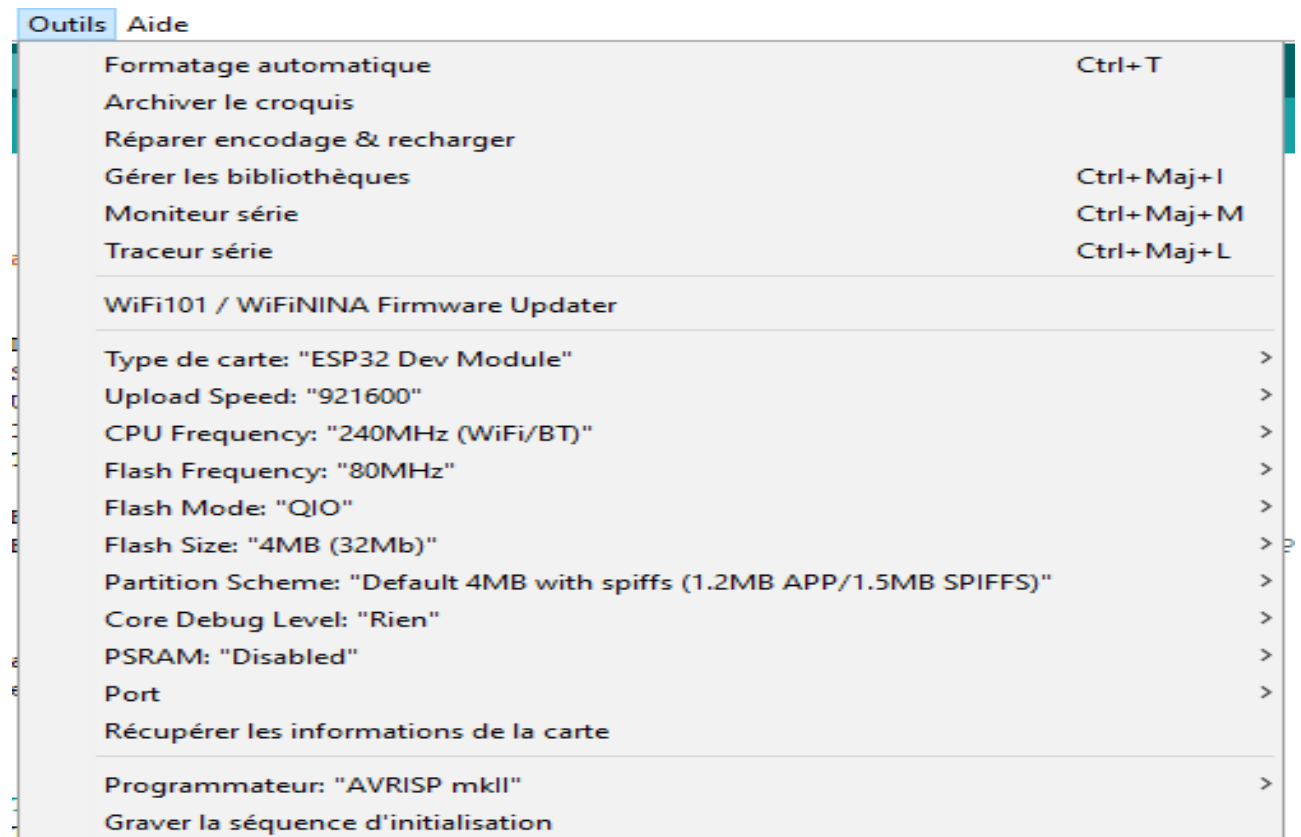
➤ Programmer l'ESP

Pour programmer l'ESP on a utilise l'IDE Arduino en activant le bon paramètre pour quele code généré soit compatible avec l'ESP. Dans notre cas, c'est un ESP32 Dev Module.

➤ Installation des librairies

Pour que le programme fonctionne, il va lui falloir la libraire DHT.La librairie DHT offreun moyen simple d'utiliser n'importe quel capteur DHT pour lire la température et l'humiditéavec cartes ESP32 ou Arduino.

Pour que la librairie DHT fonctionne elle va avoir besoin de la librairie Adafruit UnifiedSensor.



b-Code complet

```
#include "DHT.h"
#include <SPI.h>
#include <WiFi.h>
#include <ThingSpeak.h>

#define SECRET_SSID "Infinix NOTE 4"
#define SECRET_PASS "aaaaaaaa"
#define ONE_WIRE_BUS 23
#define SECRET_CH_ID 946428
#define SECRET_WRITE_APIKEY "EJK9TZINGZU4YJ30"

char ssid[] = SECRET_SSID;      // your network SSID (name)
char pass[] = SECRET_PASS;      // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;               // your network key Index number (needed only for WEP)
WiFiClient client;

unsigned long MyChannelNumber = SECRET_CH_ID;
const char *MyAPIKey = SECRET_WRITE_APIKEY; // enter your channel's Write API Key

#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.print("affichier les valeur");
  Serial.println(F("DHTxx test!"));
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);

  dht.begin();
}

void loop() {
  delay(2000);

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
```

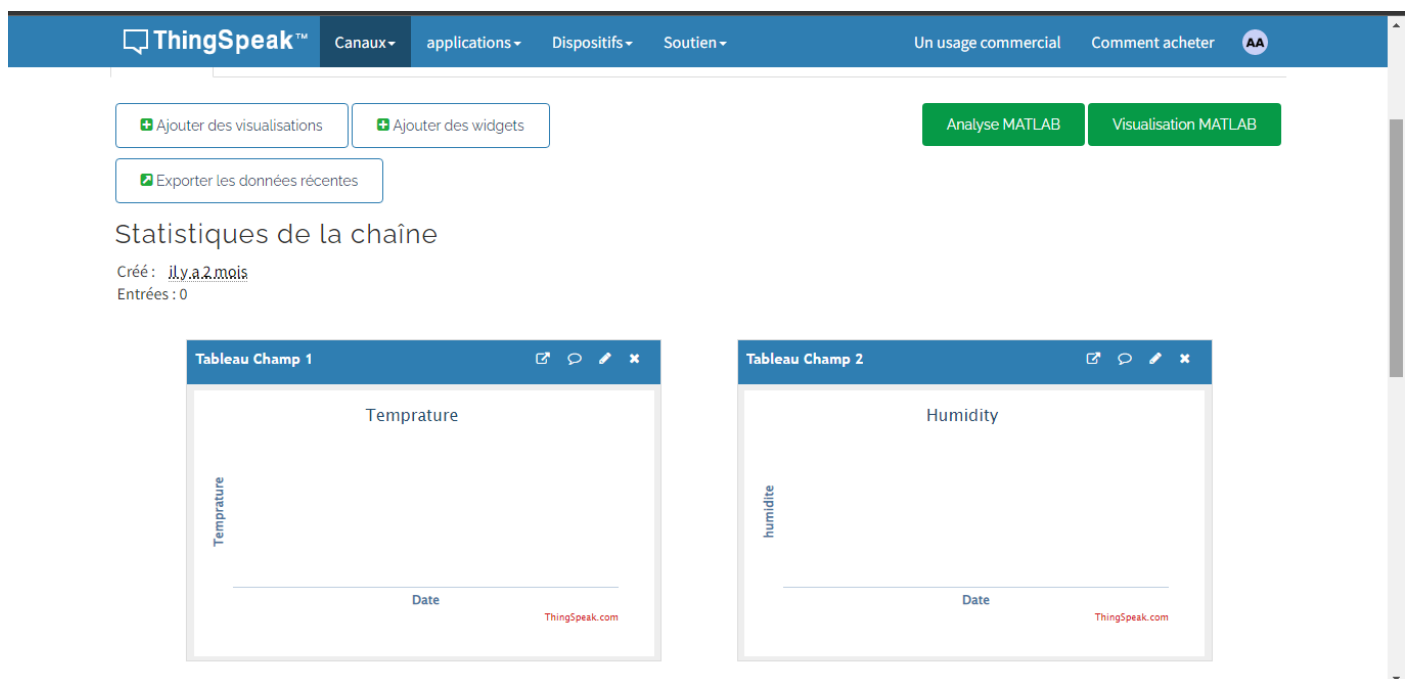
```

Serial.print(F("Humidity: "));
Serial.print(h);
Serial.print(F("% Temperature: "));
Serial.print(t);
Serial.print(F("°C "));
Serial.print(f);
Serial.print(F("°F Heat index: "));
Serial.print(hic);
Serial.print(F("°C "));
Serial.print(hif);
Serial.println(F("°F"));
if (WiFi.status() != WL_CONNECTED)
{
    Serial.print("Attempting to connect to SSID :");
    Serial.println(SECRET_SSID);
    while (WiFi.status() != WL_CONNECTED)
    {
        WiFi.begin(ssid, pass);
        Serial.print(".");
        delay(5000);
    }
    Serial.println("/n connected ");
}
Serial.print("Requesting Temperatures....");
Serial.print("Requesting Humidity....");
int x = ThingSpeak.writeField(MyChannelNumber, 1, t, MyAPIKey);
int y = ThingSpeak.writeField(MyChannelNumber, 2, h, MyAPIKey);
delay(2000);

```

6-Les résultats :

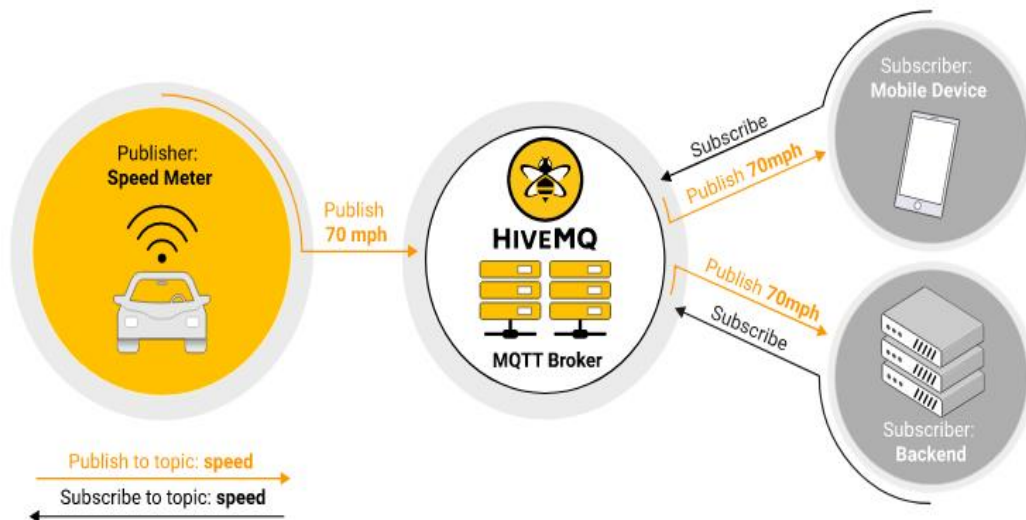
Toutes les résultats des capteurs quand on a utilisée :



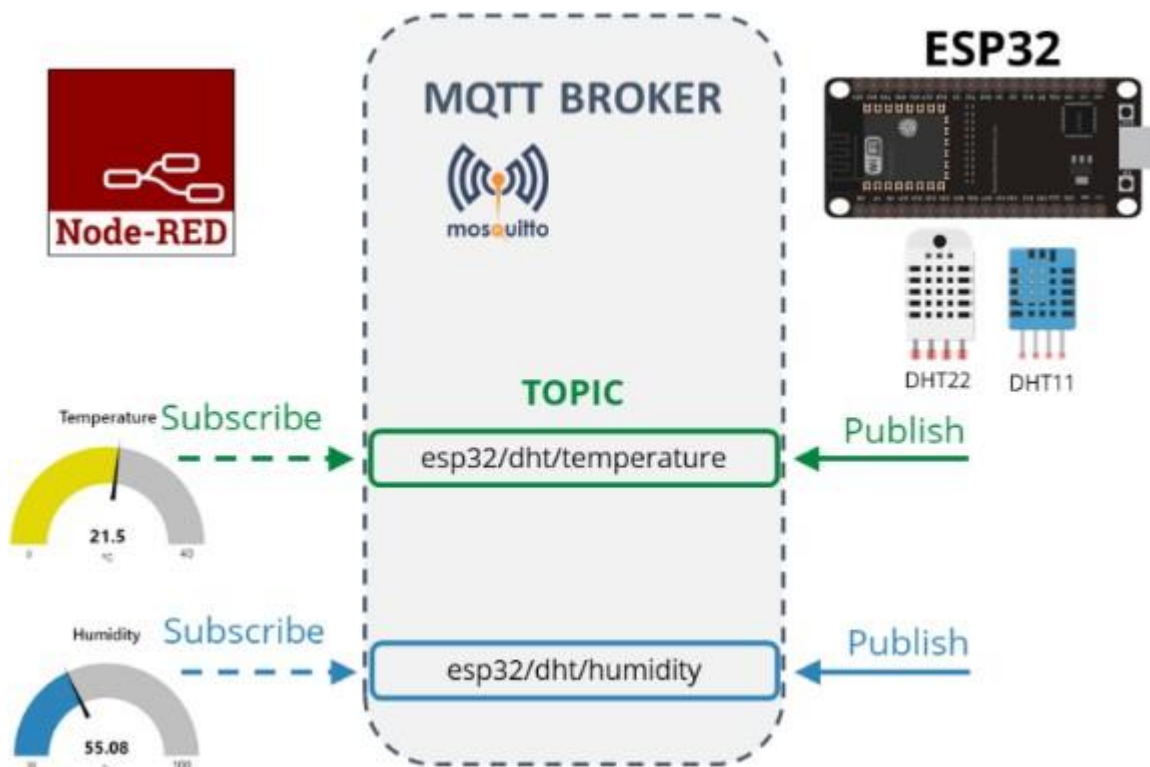
Partie 2 : MQTT Mosquitto – Node RED

1-objectif :

Dans notre projet nous avons essayé de publier les lectures des capteurs sur le tableaude bord Node-RED (laptop) à base d'une carte ESP32 qui sera programmé à l'aide de l'IDE Arduino et comme communication on a utilisé le broker MQTT (laptop).



La figure ci-dessous montre un aperçu de haut niveau du système qu'on a construit.



L'ESP32 demande des relevés de température et d'humidité au capteur DHT22. Les relevés de température sont publiés dans le sujet **esp32/dht/température**, et Les relevés d'humidité sont publiés dans le sujet **esp32/dht/humidity** avec Node-RED est abonné à ces sujets et qui sert à recevoir les lectures des capteurs et les affiche sur des jauges afin de les exploiter d'après l'utilisateur.

2-Partie SOFTWARE (Logiciels & communications)

a-MQTT BROKER

a. Présentation générale

Le transport de télémétrie Message Queuing (MQTT) est un protocole léger de messagerie de machine à machine, conçu pour fournir une communication légère de publication/abonnement entre les appareils de l'Internet des objets. Le protocole fonctionne généralement sur TCP/IP. Il est conçu pour les connexions avec des sites distants où la bande passante du réseau est limitée. Mosquitto est un courtier de messages open source (ou serveur) qui implémente les protocoles MQTT. Mosquitto a un bon support communautaire, une documentation et une facilité d'installation, il est devenu l'un des courtiers MQTT les plus populaires.

b. Fonctionnement de MQTT



MQTT est un protocole standardisé reposant sur TCP/IP. "Il est particulièrement utilisé pour transporter des données des objets connectés sur le cloud. Néanmoins, il reste lourd pour les réseaux LPWA contraints type NB-IoT et ne permet d'adresser que la remontée de données sans prendre en compte les services de device management. Ces derniers étant essentiels lors de déploiements massifs d'appareils connectés", souligne Hatem Oueslati, CEO d'IoTerop, start-up française spécialisée dans le device management. Son processus se divise en quatre étapes distinctes : connexion, authentification, communication, terminaison.

c. Les caractéristiques de MQTT

La principale caractéristique de MQTT est sa légèreté, le protocole ne requiert que des ressources minimales et peut donc être utilisé sur de petits microcontrôleurs. "L'idée de MQTT est de faire dialoguer des équipements qui ne disposent pas de ressources propres pour assurer une connexion permanente", détaille Fabien Pereira Vaz, de Paessler AG. Pour lui, MQTT se démarque par sa souplesse et sa simplicité de mise en œuvre, en plus d'assurer une transmission de données bidirectionnelle. Le protocole prend ainsi de l'importance dans les technologies opérationnelles de l'industrie.

Aussi MQTT permet la gestion des déconnexions et des reconnexions de devices de manière simplifiée. Avec La taille maximale d'un message envoyé avec MQTT est de 256 Mo

d. Installation de MQTT

Mosquitto est le broker le plus souvent utilisé pour les projets ESP (Arduino et Raspberry). Lancé en 2008, il est disponible sur toutes les plateformes (MacOS, Windows XP-10, Linux).

3-Node-Red

a. Présentation générale

Node-RED est un outil de développement basé sur les flux pour la programmation visuelle développé à l'origine par IBM pour connecter ensemble des périphériques matériels, des API et des services en ligne dans le cadre de l'Internet des objets .

Node-RED fournit un éditeur de flux basé sur un navigateur Web, qui peut être utilisé pour créer des fonctions JavaScript . Des éléments d'applications peuvent être sauvegardés ou partagés pour être réutilisés. Le runtime est construit sur Node.js . Les flux créés dans Node-RED sont stockés à l'aide de JSON . Depuis la version 0.14, les nœuds MQTT peuvent établir des connexions TLS correctement configurées. En 2016, IBM a contribué à Node-RED en tant que projet open source JS Foundation.



Nom	La description
Noeud-RED	Un outil visuel pour câbler l'Internet des objets
Tableau de bord Node-RED	Une interface utilisateur de tableau de bord pour Node-RED
Générateur de nœuds	Outil de ligne de commande pour générer des modules de nœud Node-RED à partir de plusieurs sources différentes, y compris le document Open API et la source du nœud de fonction.
Outil de ligne de commande Node-RED	L'outil de ligne de commande vous permet d'administrer à distance une instance Node-RED.

b. Installation de NODE-RED

○ Installation de Node.js

Avant d'installer Node-Red on installe d'abord la dernière version 14.x LTS de Node.js à partir de la page d'accueil officielle de Node.js. On télécharge la meilleure version pour votre système.



Une fois Node.js installé, on ouvre une invite de commande et on exécute la commande suivante **node --version** pour vous assurer que Node.js et npm sont correctement installés.

```
Microsoft Windows [version 10.0.19045.2604]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\cyber net solution>node --version
v18.12.1

C:\Users\cyber net solution>
```

○ Installation de Node-Red

L'installation de Node-RED en tant que module global ajoute la commande **node-red** à notre chemin système. On exécute ce qui suit à l'invite de commande : **npm install -g --unsafe-perm node-red**

```
Microsoft Windows [version 10.0.19045.2604]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\cyber net solution>node --version
v18.12.1

C:\Users\cyber net solution>npm install -g --unsafe-perm node-red
npm WARN cleanup Failed to remove some directories [
npm WARN cleanup   'C:\\Users\\cyber net solution\\AppData\\Roaming\\npm\\node_modules\\.node-red-G4vuezCc',
npm WARN cleanup   [Error: EPERM: operation not permitted, unlink 'C:\\Users\\cyber net solution\\AppData\\Roaming\\npm\\node_modules\\.node-red-G4vuezCc\\node_modules\\bcrypt\\lib\\binding\\napi-v3\\bcrypt_lib.node'] {
npm WARN cleanup     errno: -4048,
npm WARN cleanup     code: 'EPERM',
npm WARN cleanup     syscall: 'unlink',
npm WARN cleanup     path: 'C:\\Users\\cyber net solution\\AppData\\Roaming\\npm\\node_modules\\.node-red-G4vuezCc\\node_modules\\bcrypt\\lib\\binding\\napi-v3\\bcrypt_lib.node'
npm WARN cleanup   }
npm WARN cleanup ]
npm WARN cleanup ]

changed 292 packages in 2m

40 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 9.2.0 -> 9.5.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.5.0
npm notice Run `npm install -g npm@9.5.0` to update!
npm notice
```

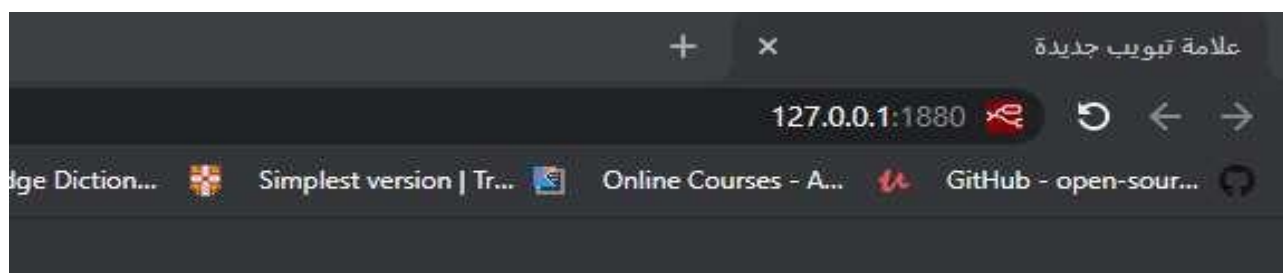
c. Exécutez Node-RED

Pour démarrer Node-RED, utilisez la commande ci-dessous :

```
npm
Microsoft Windows [version 10.0.19045.2604]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\cyber net solution>node-red
20 Feb 18:56:09 - [info]
Welcome to Node-RED
=====
20 Feb 18:56:09 - [info] Node-RED version: v3.0.2
20 Feb 18:56:09 - [info] Node.js version: v18.12.1
20 Feb 18:56:09 - [info] Windows_NT 10.0.19045 x64 LE
20 Feb 18:56:12 - [info] Loading palette nodes
```

Pour accéder au flux de travail Web Node-RED, copiez l'adresse IP de l'hôte local avec Port et accédez via le navigateur



Pour connecter le module Arduino aux entrées et sorties Node-Red mqtt sont ajoutées au projet. Les sujets Arduino sont définis dans Node-Red en double-cliquant sur le nœud mqtt, puis définissez le sujet pour qu'il corresponde au sujet Arduino.

Nous définissons les thèmes et valeurs MQTT:

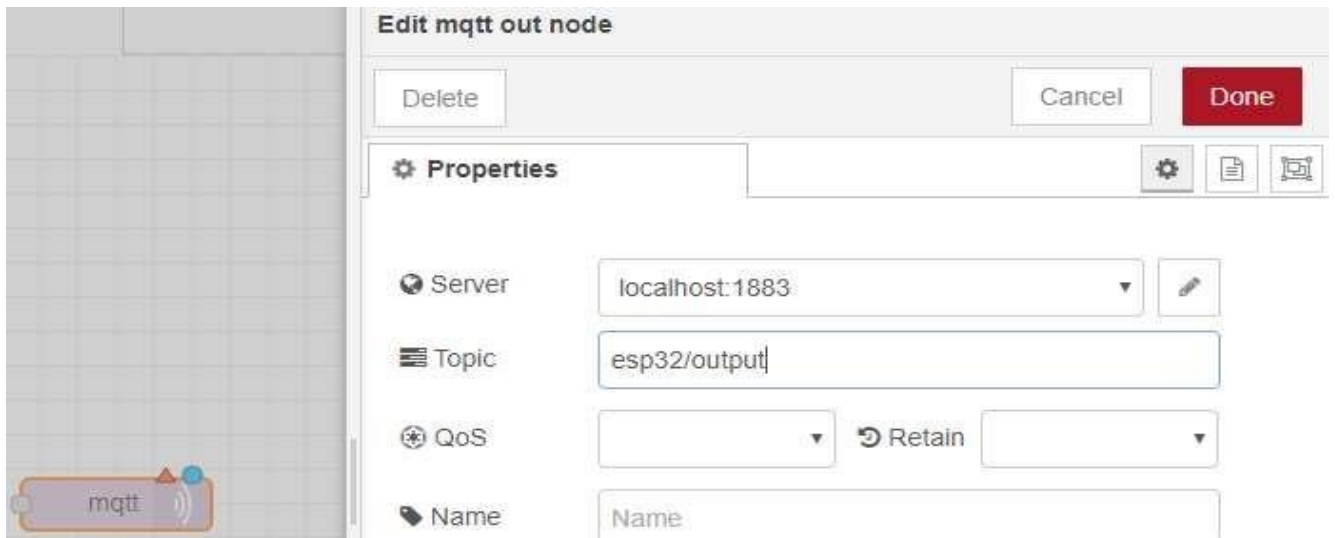
✚ Pour la température: "esp32 / temperature" avec la valeur est un entier ou une chaîne

noeud d'entrée mqtt. Ce nœud est abonné à la topic esp32 / temperature pour recevoir les données du capteur de température de l'ESP32. L'ESP32 publiera les relevés de température sur ce topic.

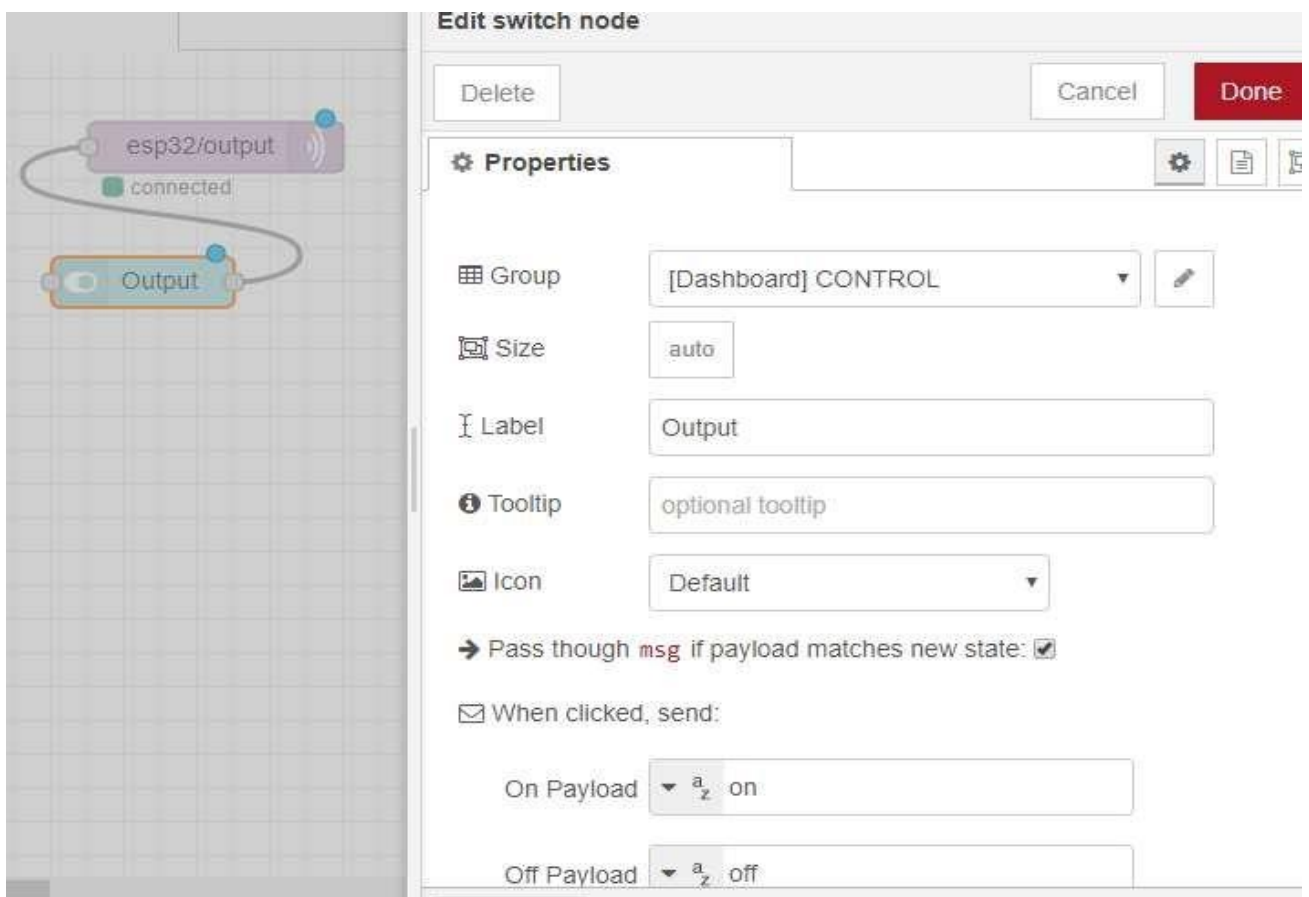




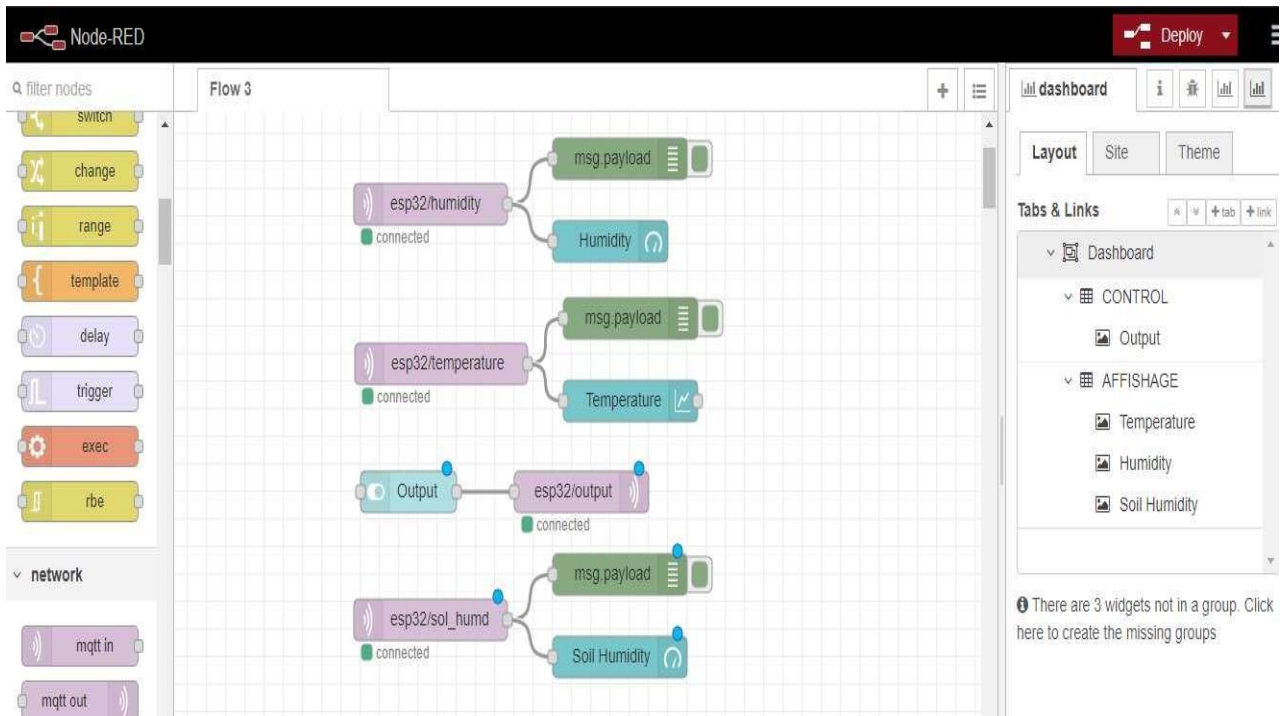
Pour Led: "esp32 / output" avec valeur off, on



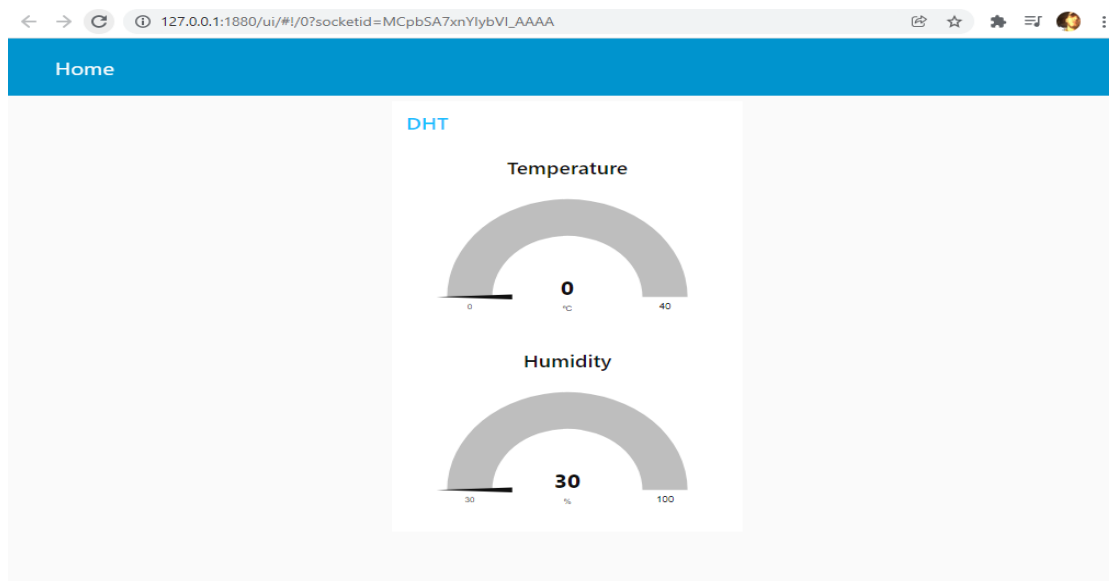
switch - le switch envoie un message de chaîne on lorsqu'il est activé; et envoie un message de chaîne désactivée lorsqu'il est désactivé. Ce nœud sera publié sur le sujet de la esp32 / output. Votre ESP sera alors abonné à ce sujet, pour recevoir ses messages



Une fois les connexions MQTT configurées, les tableaux de bord Web peuvent présenter les données finales. Les tableaux de bord Web offrent un certain nombre de composants différents qui pourraient être utilisés pour cet exemple, nous avons utilisé une jauge et un graphique.



Pour compiler et afficher l'application Node-Red, cliquez sur le bouton Déployer sur le côté droit de la barre de menus. Pour accéder au tableau de bord Web, entrez: `http://your-ip:1880/ui`. Voici un exemple de ce que vous devriez voir.



4-Arduino code

🔧 Installation de la bibliothèque PubSubClient :

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

Il existe un certain nombre d'excellentes bibliothèques MQTT pour Arduino, pour cet exemple, nous avons utilisé la bibliothèque PubSubClient. Cette bibliothèque peut être installée à partir de l'IDE Arduino en sélectionnant les éléments de menu :

Esquisse -> Ajouter une bibliothèque -> Gérer les bibliothèques Pour obtenir la configuration MQTT, vous devez:

- ✚ Définir le SSID et le mot de passe pour votre WAN
- ✚ Définir l'adresse IP du serveur MQTT
- ✚ Définir un sujet pour les données

```
const char* ssid = "*****";  
const char* password = "*****";  
const char* mqtt_server = "localhost";  
WiFiClient espClient;  
PubSubClient client(espClient);
```

b. Arduino code Complet

```
#include <WiFi.h>  
  
#include<PubSubClient.h>  
#include "DHT.h"  
  
#define DHTTYPE DHT22 // DHT 22  
  
const char* ssid = "*****";  
const char* password = "*****";  
const char* mqtt_server = "localhost";  
WiFiClient espClient;  
PubSubClient client(espClient);  
  
const int DHTPin = 5;  
const int led = 4;  
DHT dht(DHTPin, DHTTYPE);  
  
// Timers auxiliar variables  
long now = millis();  
long lastMeasure = 0;
```

```

Void setup_wifi() {delay(10);

// We start by connecting to a WiFi network

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {delay(250);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected - ESP IP address: ");Serial.println(WiFi.localIP());

}

void callback(String topic, byte* message, unsigned int length) {Serial.print("Message
arrived on topic: ");

Serial.print(topic);

Serial.print(". Message: ");

String messageTemp;

for (int i = 0; i < length; i++) {Serial.print((char)message[i]);

messageTemp += (char)message[i];

}

if(topic=="esp32/output"){ Serial.print("Changing Room lamp to ");

if(messageTemp == "on"){

digitalWrite(led, HIGH);Serial.print("On");

}

else if(messageTemp == "off"){digitalWrite(led, LOW); Serial.print("Off");

}

}

}

void reconnect() {

```



```

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Attempt to connect

if (client.connect("ESP8266Client")) {Serial.println("connected");

client.subscribe("esp32/output");

} else { Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retryingdelay(5000);

}

}

}

void setup()

{ pinMode(led, OUTPUT);

dht.begin();

Serial.begin(115200);

setup_wifi();

client.setServer(mqtt_server, 1883);

client.setCallback(callback);

}

void loop() {

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

}

if(!client.loop()) client.connect("ESP8266Client");

now = millis();

if (now - lastMeasure > 30000) {lastMeasure = now;

float humidity = dht.readHumidity();

float temperature = dht.readTemperature();

if (isnan(humidity) || isnan(temperature)) {

```

```

    Serial.println("Failed to read from DHT sensor!");
    return;
}

char tempString[8];
dtostrf(temperature, 1, 2, tempString);
Serial.print("Temperature: ");
Serial.println(tempString);

char humString[8];
dtostrf(humidity, 1, 2, humString);
Serial.print("Humidity: ");
Serial.println(humString);

Serial.print("Humidity: ");

Serial.print(humidity);
Serial.print(" %\t Temperature: ");
Serial.print(temperature);
Serial.print(" %\t");
}
}

```

Ce code publie des relevés de température et d'humidité sur les sujets esp32/temperature et esp32/humidity via le protocole MQTT.

L'ESP32 est abonné à la topic esp32/output pour recevoir les messages publiés sur ce topic par l'application Node-RED. Ensuite, en fonction du message reçu, il allume ou éteint la LED.

c. Abonnement au topic MQTT

Dans la fonction reconnect (), vous pouvez vous abonner au topic MQTT. Dans ce cas, l'ESP32 n'est abonné qu'à la esp32/output :

```
client.subscribe("esp32/output");
```

Dans la fonction callback (), l'ESP32 reçoit les messages MQTT des topics abonnés. Selon le sujet et le message MQTT, il allume ou éteint la LED :

```

if(topic=="esp32/output"){
Serial.print("Changing Room lamp to ");
if(messageTemp == "on"){
digitalWrite(led, HIGH);
Serial.print("On");
}

else if(messageTemp == "off"){
digitalWrite(led, LOW);
Serial.print("Off");
}
}

```

d. Publication de messages MQTT

Dans la loop(), de nouvelles lectures sont publiées toutes les 3 secondes:

```

if (now - lastMeasure > 3000) { ..... }

```

Nous devons convertir la variable de température flottante en un tableau de caractères, afin de pouvoir publier la lecture de la température dans la topic esp32/temperature:

Le même processus est répété pour publier la lecture d'humidité dans la topic esp32/humidity:

```

char tempString[8];
dtostrf(temperature, 1, 2, tempString);
Serial.print("Temperature: ");
Serial.println(tempString);

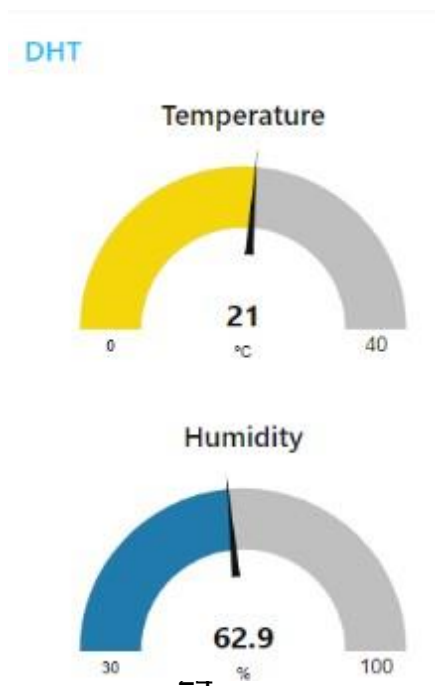
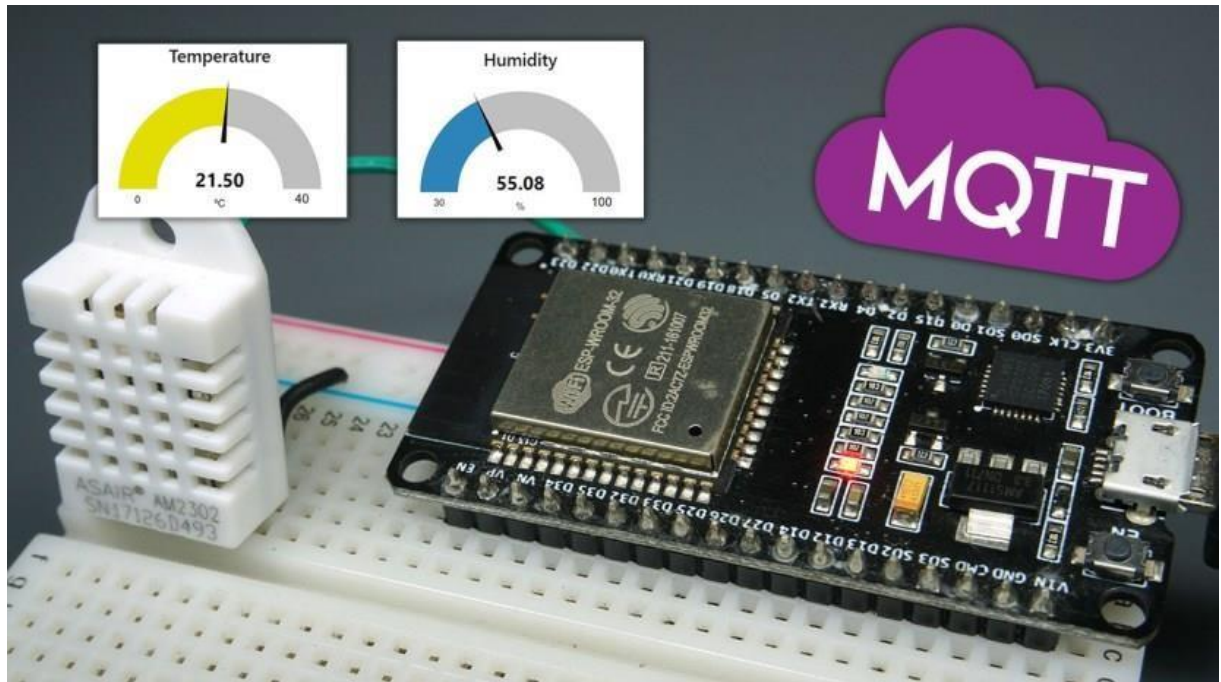
client.publish("esp32/temperature", tempString);

char humString[8];
dtostrf(humidity, 1, 2, humString);
Serial.print("Humidity: ");
Serial.println(humString);

client.publish("esp32/humidity", humString);

```

Après qu'on téléverse le code Arduino dans notre carte ESP32, et après qu'on fait une connexion entre MQTT et Node-Red, et après qu'on Prépare le Dashboard de Node Red, et après qu'on fait déployer notre flux. Finalement on peut recevoir à chaque 10 secondes les valeurs de température et d'humidité capter par DHT22 et traiter par ESP32 et on les affiche à l'aide de Node-Red en utilisant MQTT comme communication.



Conclusion

Pour conclure on peut dire que MQTT est un excellent protocole de communication dans l'iot qui sert à échanger de petites quantités de données entre appareils.

Dans ce rapport, on a présenté la façon de publier les lectures de température et d'humidité d'un capteur DHT22 avec l'ESP32 sur différents sujets MQTT.

On peut utiliser n'importe quel appareil ou plate-forme domotique pour vous abonner à ces sujets et recevoir les lectures.

Aussi à la place d'un capteur DHT22, on peut utiliser n'importe quel autre capteur comme un capteur de température DS18B20 ou un capteur BME280 tel que :

- ESP32 MQTT - Publier les relevés de température DS18B20
- ESP32 MQTT - Publier les relevés de température, d'humidité et de pression BME280

Nous espérons que vous avez trouvé ce rapport utile.