TP 3: Simulation de projets IoT avec Wokwi

Introduction.

- **Objectif :** Apprendre à simuler des projets IoT en utilisant la plateforme Wokwi, tinkspeak, protocol HTTP, MQTT et CoAP.

TP 3.1. Création d'un Nouveau Projet sur Wokwi: Lecture de Température et Humidité avec ESP32, DHT22

Objectifs du TP :

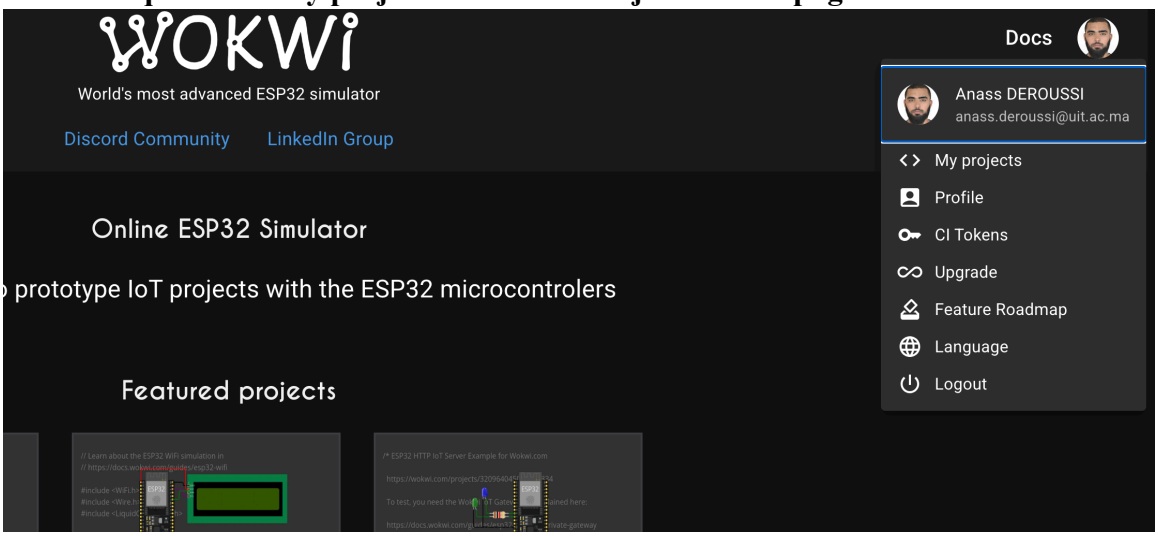
1. **Lire les données** de température et d'humidité avec un capteur **DHT22** connecté à un **ESP32**.
2. **Se connecter au WiFi simulé** dans **Wokwi**.

1. Accéder à Wokwi :

- **Rendez-vous sur le site officiel : https://wokwi.com**
- **Aucun compte n'est nécessaire pour tester des projets simples, mais il est recommandé de créer un compte gratuit pour sauvegarder vos projets.**

2. Cliquez sur l'image de votre compte:

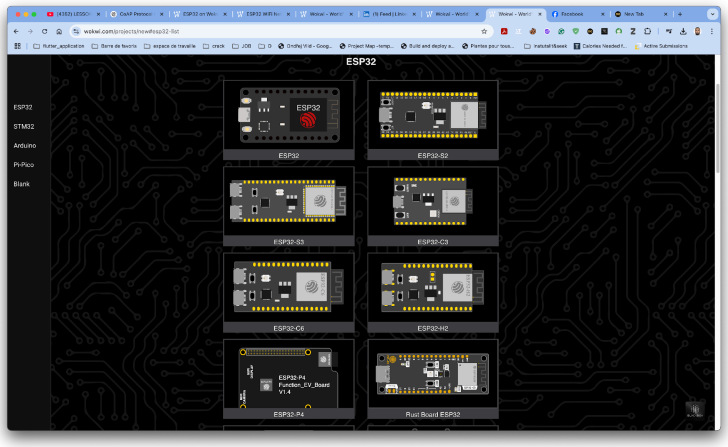
Cliquez sur "My projects" "New Project" sur la page d'accueil.

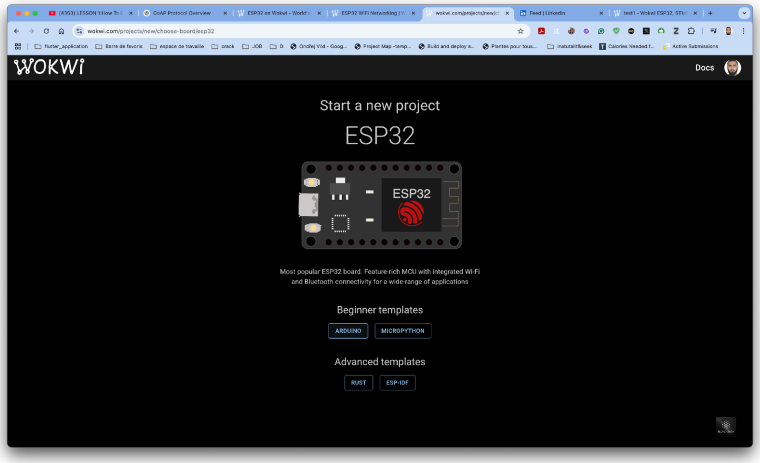


Sélectionnez le type de microcontrôleur que vous souhaitez utiliser :

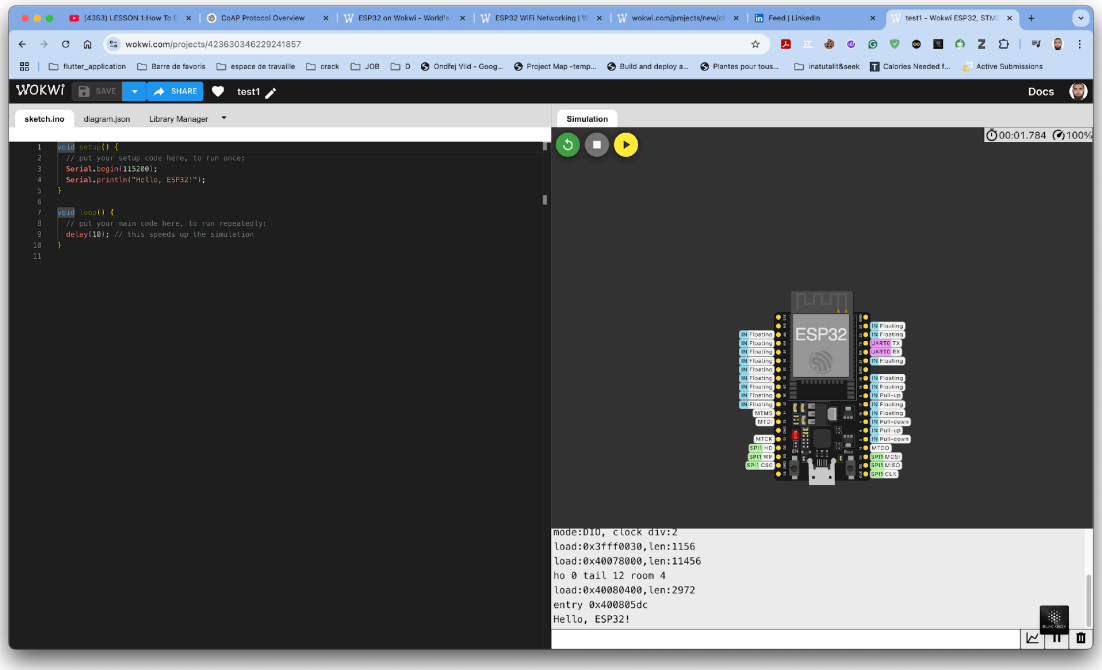
- **Arduino UNO**
- **ESP32**
- **Raspberry Pi Pico**

Exemple: ESP32 □ Arduino





2. **Un éditeur de code s'ouvre avec un modèle de code de base.**

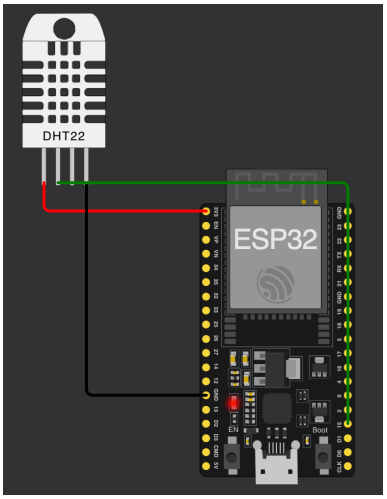


4.3. Ajouter des Composants :

1. Cliquez sur l'icône "+" dans le coin supérieur droit.
2. Recherchez le composant que vous souhaitez ajouter :
   - Par exemple : **DHT22**
3. Faites glisser le composant sur l'espace de travail.

4.4. Faire les Connexions :

1. Cliquez sur un PIN du microcontrôleur.
2. Reliez-le au PIN correspondant du capteur ou de l'actionneur.
    - Les connexions se font en cliquant et en faisant glisser les fils.
3. Les couleurs des fils peuvent être modifiées pour améliorer la lisibilité du schéma.



Ex: Le DHT22 possède généralement trois broches utiles :

- **VCC : Alimentation (3,3 V ou 5 V selon le module)** Reliez la broche **VCC** du DHT22 à la broche **3,3 V** de l'ESP32:
- **GND : Masse.** Connectez la broche **GND** du DHT22 à une broche **GND** de l'ESP32.
- **DATA : Signal de données.** **GPIO 15** de l'ESP32

Certaines versions du DHT22 ont une quatrième broche (**NC**, non connectée), que vous pouvez ignorer.

4.5. Écrire le Code :

- **L'éditeur de code intégré prend en charge :**
    - **C/C++ pour Arduino et ESP32.**
    - **MicroPython pour ESP32 et Raspberry Pi Pico.**
- **Exemple de Code pour ESP32 avec DHT22 :**

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

#define DHTPIN 15         // GPIO utilisé pour le DHT22
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);


void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Erreur de lecture du DHT22");
    return;
  }

  Serial.print("Température: ");
  Serial.print(t);
  Serial.print("°C  Humidité: ");
  Serial.print(h);
```

```
  Serial.println("%");

  delay(2000);
}
```

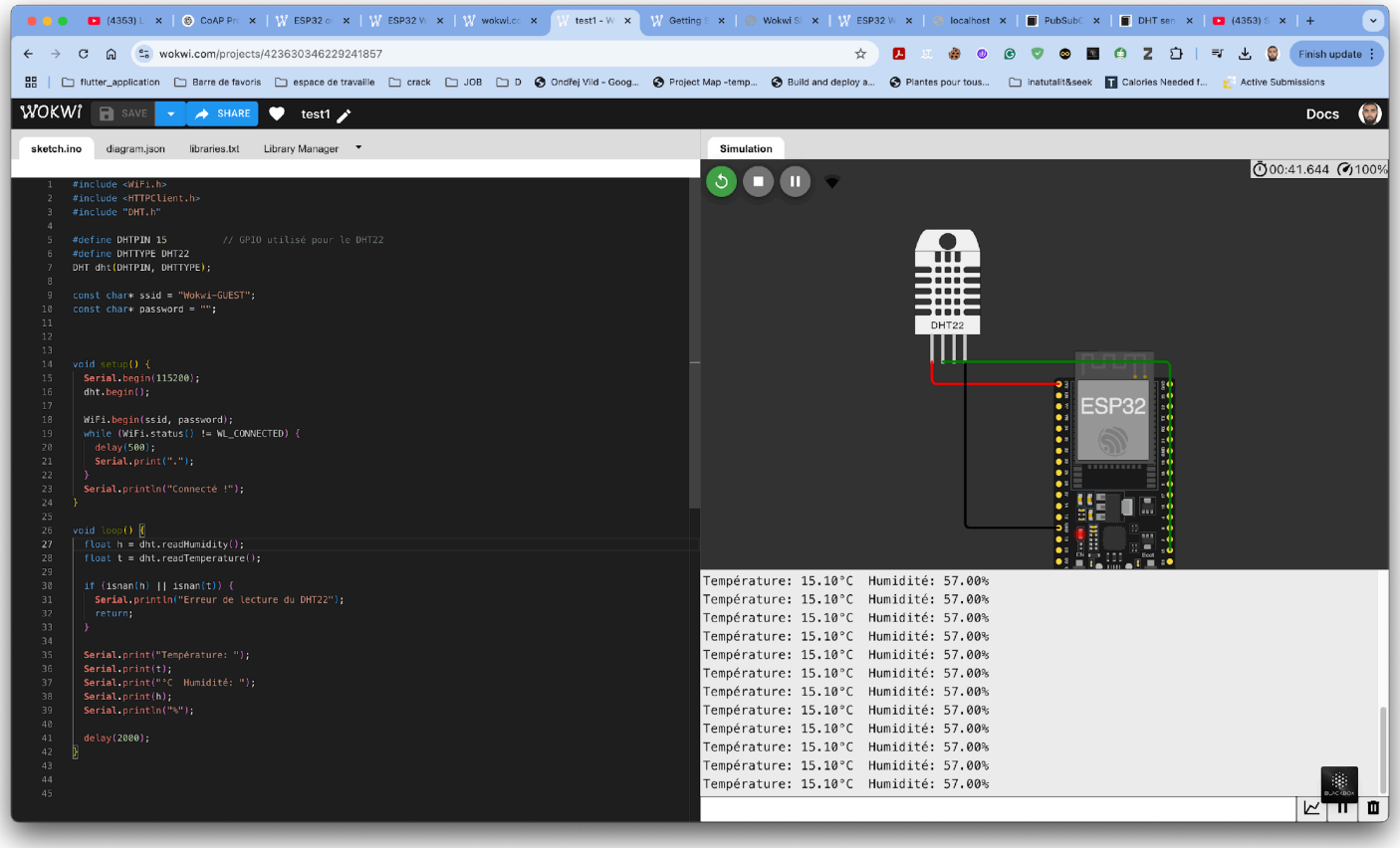
Importer les bibliothèques **DHT sensor library**.

4.6. Lancer la Simulation :

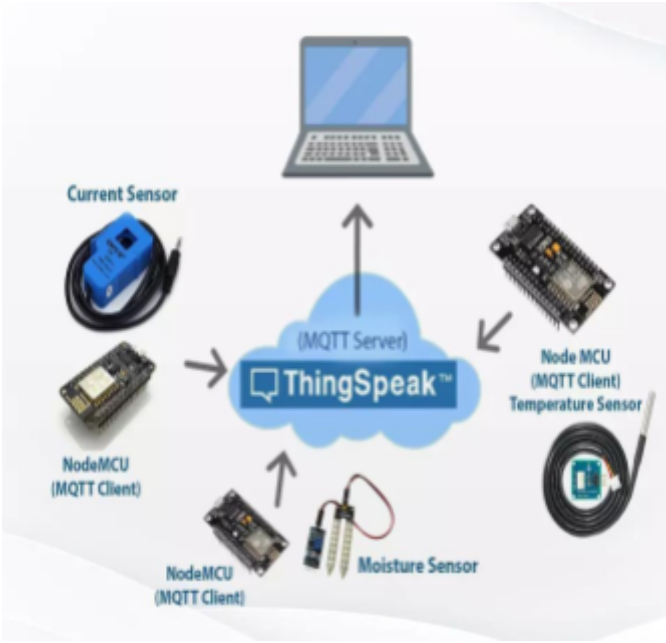
- **Cliquez sur "Start Simulation" dans le coin supérieur droit.**
- **Le terminal série s'affiche en bas, montrant les sorties Serial.print().**

4.7. Sauvegarder et Partager le Projet :

- **Cliquez sur "Share" pour générer un lien partageable.**
- **Enregistrez votre projet en créant un compte gratuit sur Wokwi.**



T3.2. Lecture de Température et Humidité avec ESP32, DHT22 et Envoi des Données à ThingSpeak dans Wokwi. (Http)



ThingSpeak : ThingSpeak est une **plateforme de cloud** pour **l'Internet des Objets (IoT)** qui permet de :

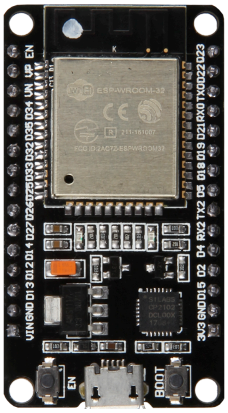
- **Collecter** des données depuis des capteurs en temps réel.
- **Analyser et visualiser** les données sous forme de graphiques.
- **Automatiser des actions** en utilisant des déclencheurs et des alertes.

Pourquoi utiliser ThingSpeak ?

- Pour **surveiller à distance** des paramètres environnementaux (température, humidité, etc.).
- Pour **enregistrer les données** dans le **cloud** et les consulter depuis n'importe quel appareil connecté à Internet.
- Pour **intégrer des applications IoT** avec d'autres services via des **API**, **HTTP** et **MQTT**.

Objectifs du TP :

- Programmer un **ESP32** dans **Wokwi** pour **lire** les données de température et d'humidité depuis un capteur **DHT22**.
- **Envoyer ces données à ThingSpeak** en utilisant le protocole **HTTP**.
- **Visualiser les données en temps réel** sur des **graphiques dans le tableau de bord ThingSpeak.**



Matériel:

1. **ESP32 : Microcontrôleur avec WiFi intégré.**
2. **DHT22 : Capteur de température et d'humidité.**
3. **ThingSpeak : Plateforme cloud pour l'IoT permettant de visualiser des graphiques en temps réel.**
4. **Wokwi : Simulateur en ligne pour ESP32 et DHT22.**

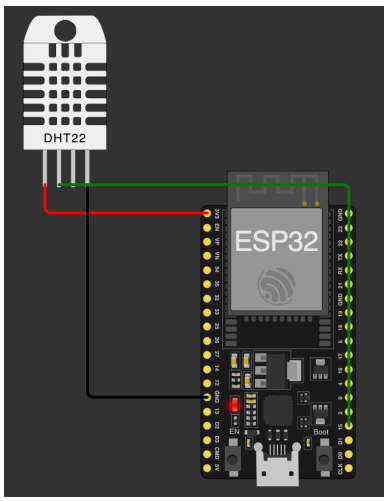
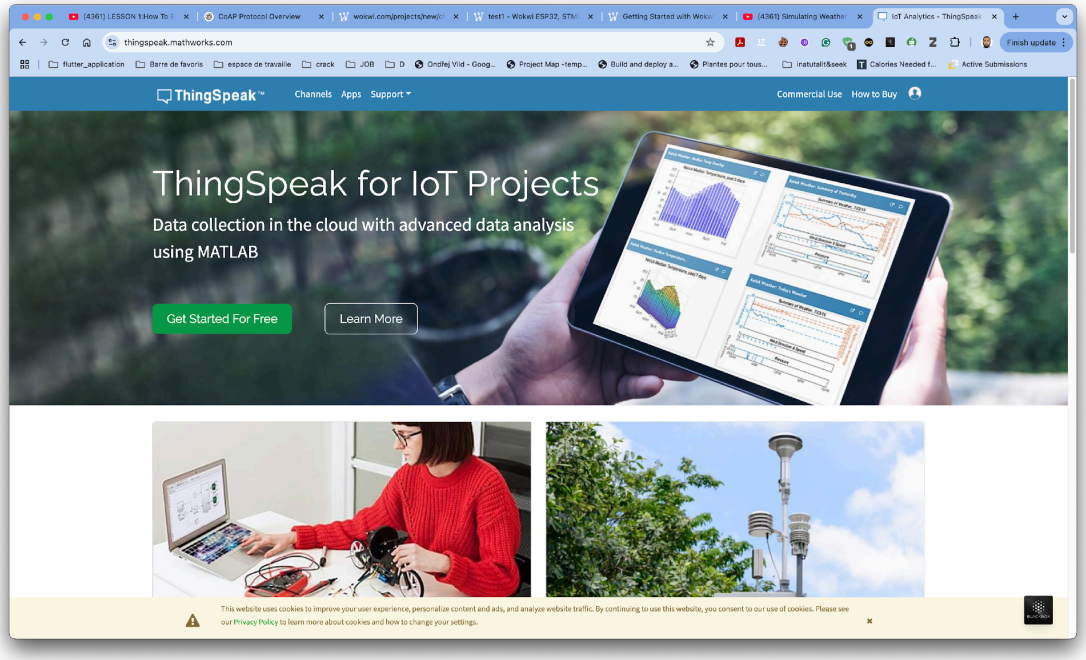


Schéma de Connexion(wokwi) :

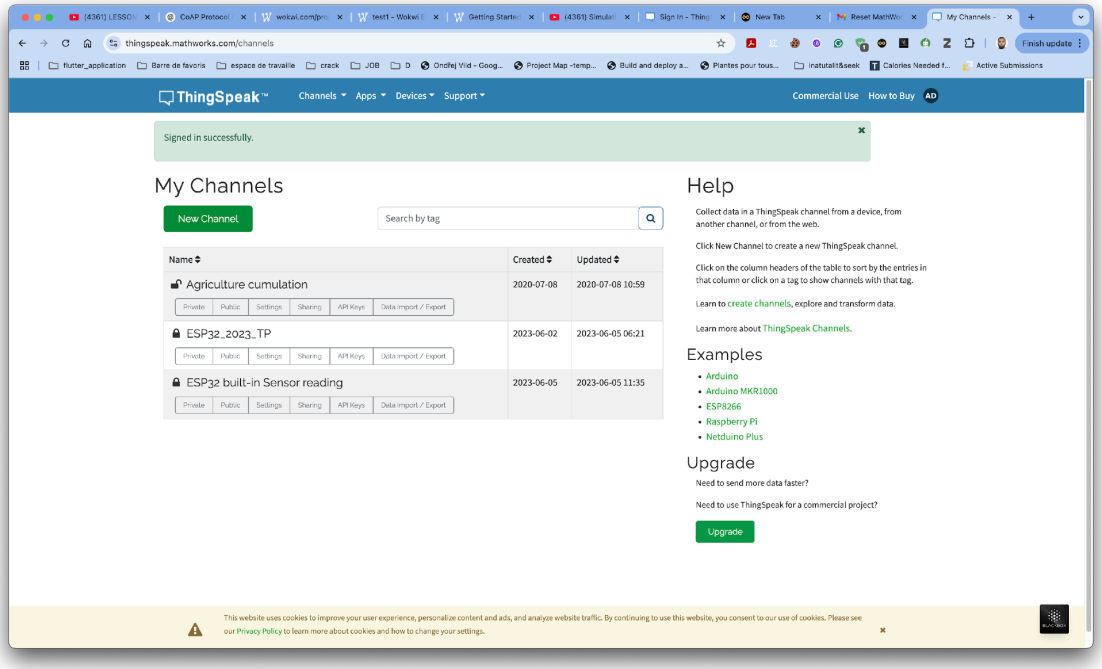
- **DHT22** :
  - **VCC → 3.3V** de l'ESP32
  - **GND → GND** de l'ESP32
  - **Data → GPIO 15** de l'ESP32
  -

Configuration de ThingSpeak:

1. **Accédez à ThingSpeak** : https://thingspeak.com

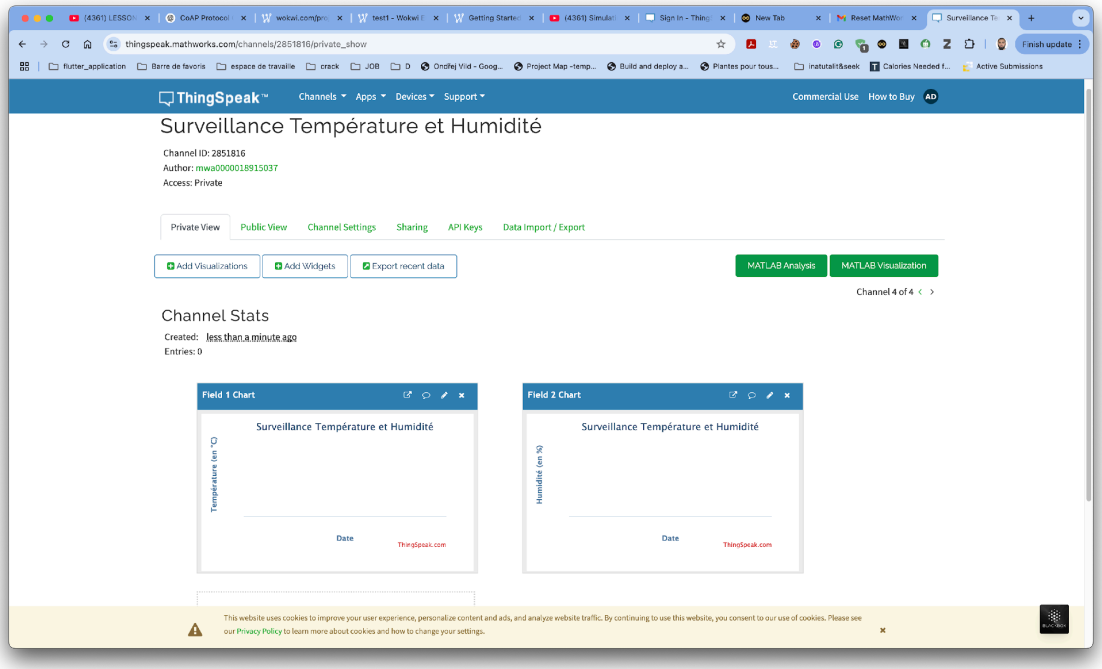


2. **Get Started For Free et Créez un compte gratuit** ou connectez-vous.



3. **Créez un nouveau canal**:

- **Nom** : Surveillance Température et Humidité
- **Champ 1** : Température (en °C)
- **Champ 2** : Humidité (en %)



4. **Obtenez l'API Key** :
    a. Allez dans l'onglet **API Keys** du canal créé.
    b. **Copiez l'API Key d'écriture** (exemple : `YOUR_API_KEY`).

Étape 4 : Écrire le Code Arduino:

4.1. Importer les Bibliothèques

```
#include <WiFi.h>         // Pour la connexion WiFi
#include <HTTPClient.h>   // Pour les requêtes HTTP
#include "DHT.h"          // Pour le capteur DHT22
```

4.2. Configurer le Capteur DHT22

```
#define DHTPIN 15         // GPIO utilisé pour le DHT22
#define DHTTYPE DHT22     // Type du capteur
DHT dht(DHTPIN, DHTTYPE);  // Initialisation du capteur
```

4.3. Configuration du WiFi (Simulé dans Wokwi)

Dans Wokwi, utilisez toujours ce **SSID** :

```
const char* ssid = "Wokwi-GUEST";   // Réseau WiFi simulé
const char* password = "";          // Aucun mot de passe requis
```

4.4. Configuration de ThingSpeak

```
String apiKey = "YOUR_API_KEY";  // Remplacez par votre API Key
const char* server = "api.thingspeak.com";
```

4.5. Initialisation dans le setup()

```
void setup() {
  Serial.begin(115200);   // Initialisation du Moniteur Série
  dht.begin();            // Initialisation du capteur DHT22

  // Connexion au WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");    // Affiche des points pour indiquer la connexion en cours
  }
  Serial.println("\nConnecté !");
}
```

4.6. Lecture et Envoi des Données à ThingSpeak

```
void loop() {
  // Lecture des données du DHT22
  float h = dht.readHumidity();       // Lecture de l'humidité
  float t = dht.readTemperature();    // Lecture de la température

  // Vérification des valeurs reçues
  if (isnan(h) || isnan(t)) {
    Serial.println("Erreur de lecture du DHT22");
    return;
  }

  // Affichage des valeurs dans le Moniteur Série
  Serial.print("Température: ");
  Serial.print(t);
  Serial.print("°C  Humidité: ");
  Serial.print(h);
  Serial.println("%");

  // Envoi des données à ThingSpeak
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = String("http://") + server + "/update?api_key=" + apiKey +
                 "&field1=" + String(t) + "&field2=" + String(h);
    http.begin(url.c_str());
    int httpCode = http.GET();
    if (httpCode > 0) {
      Serial.println("Données envoyées à ThingSpeak !");
    } else {
      Serial.println("Erreur d'envoi !");
    }
    http.end();
  }

  delay(20000);  // Envoi toutes les 20 secondes
}
```

Étape 5 : Lancer la Simulation sur Wokwi

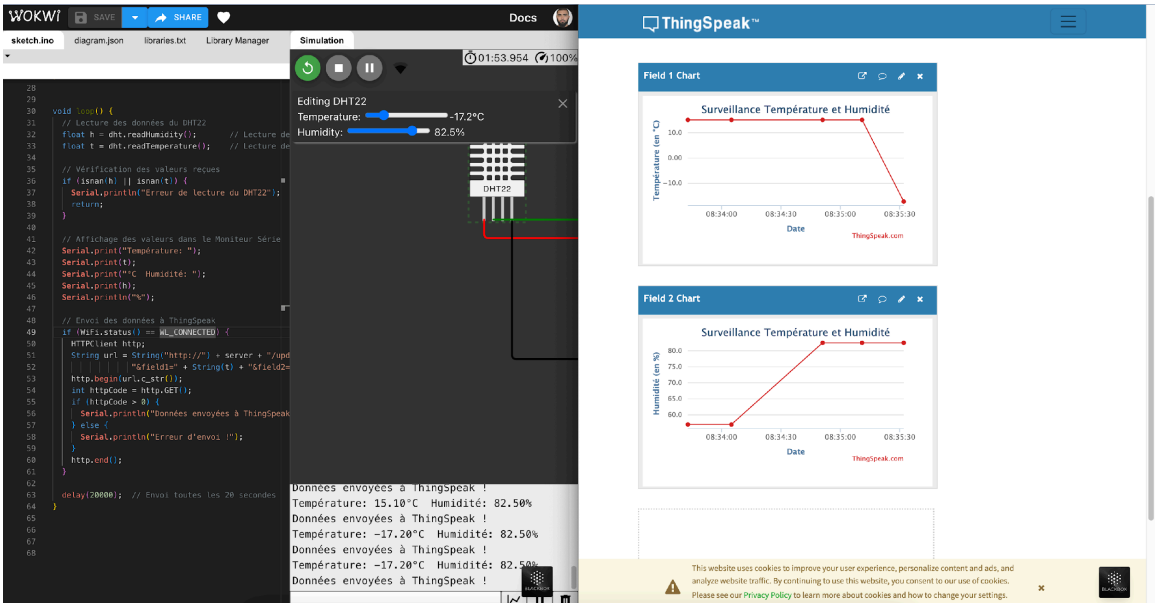
1. **Cliquez sur "Start Simulation"** dans le coin supérieur droit.
2. **Ouvrez le Moniteur Série** pour afficher les données.

Que devrait-on Voir ?

- La console devrait afficher :

```
load:0x40080400,len:2972
entry 0x400805dc
...
Connecté !
Température: 15.10°C  Humidité: 57.00%
Données envoyées à ThingSpeak !
```

- **Accédez à ThingSpeak** et regardez votre **tableau de bord** pour voir les graphiques en temps réel.

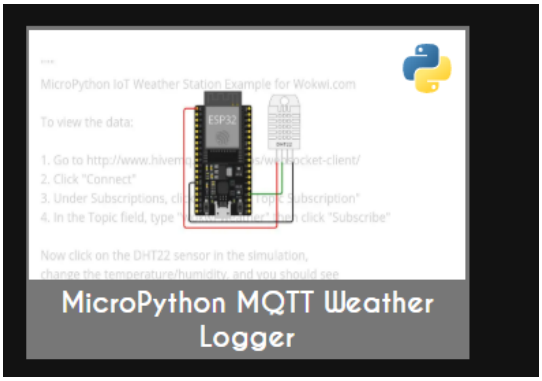


TP T3.3 Lecture de Température et Humidité avec ESP32, DHT22 et MQTT (Simulation Wokwi)

1. Objectif du TP

Ce TP a pour objectif de :

- Lire la **température et l'humidité** à partir du capteur **DHT22**
- Utiliser une carte **ESP32**
- Envoyer les données vers un **broker MQTT**
- Observer les données via un **client MQTT**



Ce TP permet de comprendre :

- L'acquisition de données IoT
- La communication **MQTT**
- Le fonctionnement d'un **broker MQTT**

2. Architecture du système

DHT22 → ESP32 → WiFi → Broker MQTT → Client MQTT
(capteur) (traitement) (internet) (Communication) et (visualisation)

- **DHT22** : mesure température et humidité
- **ESP32** : lit le capteur et envoie les données
- **MQTT Broker** : serveur qui reçoit les messages
- **MQTT Client** : application qui affiche les données

Le projet correspond à : **MicroPython MQTT Weather Logger**

Dans Wokwi :

1. Aller sur **Featured Projects**
2. Chercher : MQTT Weather Logger
3. Ouvrir le projet
4. Cliquer **Start Simulation**
5. Pendant la simulation :
   a. Cliquer sur **DHT22**
   b. Modifier : température / humidité
   c. L'ESP32 envoie les données via **MQTT**

3. Visualisation des données MQTT

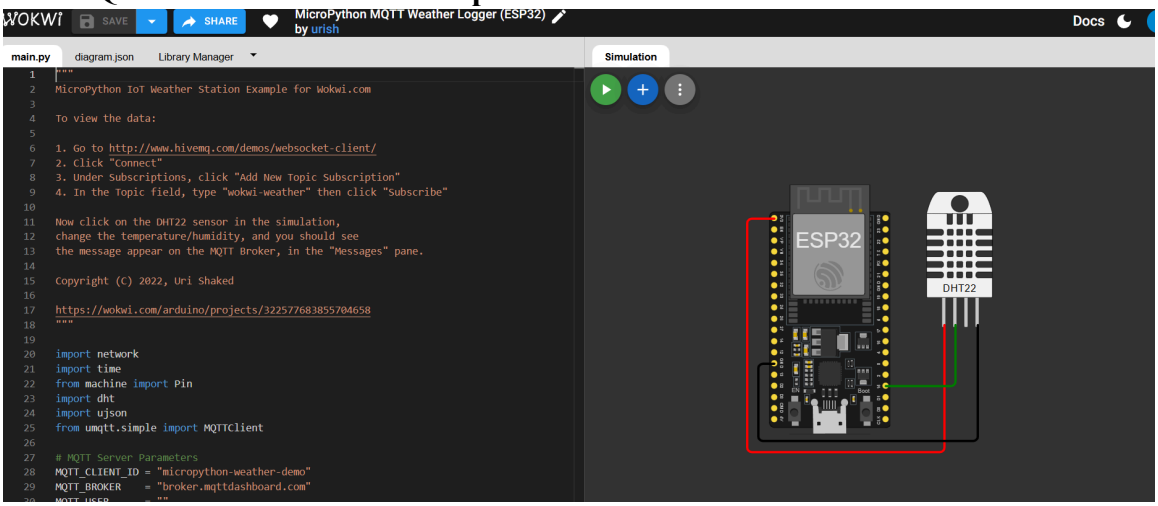
1. Aller sur le client MQTT : http://www.hivemq.com/demos/websocket-client/
2. Cliquer sur : `Connect`
3. Puis **Subscribe.**
4. Changer le Topic: `wokwi-weather`

Code MicroPython principal :

```
import network
from umqtt.simple import MQTTClient
import dht
from machine import Pin
import time
# Lire le capteur :
sensor = dht.DHT22(Pin(15))
sensor.measure()

temp = sensor.temperature()
hum = sensor.humidity()

# Envoyer MQTT :

client.publish("wokwi-weather", message)
```

Questions :

1. Quel est le rôle du **broker MQTT** ?
2. Pourquoi utilise-t-on **WiFi avec ESP32** ?
3. Quelle est la différence entre **publisher et subscriber** ?



T3.4 : Affichage des Données d'une API Web sur écran TFT avec ESP32

Objectif du TP

Dans ce TP, les étudiants vont :

- connecter **ESP32 au WiFi**
- récupérer des données depuis une **API Web via HTTP**
- décoder les données **JSON**
- afficher les résultats sur un **écran TFT**

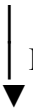
Ce TP permet de comprendre :

- communication **HTTP**

- utilisation d'une **API REST**
- traitement **JSON**
- affichage sur écran **TFT SPI**

2. Architecture du système

API Internet



HTTP GET

ESP32

SPI

TFT ILI9341

Composantes : ESP32, écran **ILI9341**, bouton poussoir, WiFi, API Web

3. Principe de fonctionnement

1. ESP32 se connecte au **WiFi**
2. Lorsque l'utilisateur appuie sur le **bouton**
3. ESP32 envoie une requête :

`HTTP GET`

vers une **API**

4. L'API renvoie une réponse **JSON**
5. ESP32 analyse le JSON
6. Les données sont affichées sur le **TFT**

Logique du code :

Connexion WiFi

```
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
 delay(500);
}
```

Requête HTTP

```
HTTPClient http;
http.begin(url);
http.GET();
String payload = http.getString();
```

Décodage JSON

```
DynamicJsonDocument doc(1024);
deserializeJson(doc, payload);

float temp = doc["current_weather"]["temperature"];
```

Affichage TFT

```
tft.setCursor(20,50);
tft.print("Temperature:");
tft.println(temp);
```

Questions :

1. Quelle est la différence entre **HTTP et MQTT** ?
2. Pourquoi les API utilisent souvent **JSON** ?
3. Pourquoi ESP32 est adapté pour les applications **IoT** ?
4. Quel est le rôle du **SPI** dans ce projet ?
5. Pourquoi utilise-t-on un **bouton pour déclencher la requête** ?