Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Internet des Objets (IoT) et Cloud Computing

CH4: Mise en Œuvre des Plateformes IoT

Par:

Hassan EL FADIL

elfadil.h@gmail.com

1. Étapes pour mettre en œuvre une plateforme IoT

N°	Etape	Exemple
1	**Définir les Objectifs et les Cas d'Utilisation**	• Objectif : Créer un système de surveillance environnementale pour un entrepôt. • Cas d'Utilisation : Surveillance de la température, de l'humidité et de la luminosité pour assurer des conditions optimales de stockage
2	**Sélectionner les Composants Technologiques**	• Capteurs : Capteur de température et d'Humidité DHT22, capteur de luminosité TSL2561. • Carte Embarquée : Raspberry Pi 4 pour collecter les données des capteurs. • Passerelle Cloud : AWS IoT Core pour stocker et traiter les données IoT.
3	**Concevoir l'Architecture Globale**	• Les capteurs (DHT22, TSL2561) sont connectés au Raspberry Pi 4 via des interfaces GPIO. • Le Raspberry Pi 4 collecte les données des capteurs et les envoie à AWS IoT Core via le protocole MQTT.

…Étapes pour mettre en œuvre une plateforme IoT

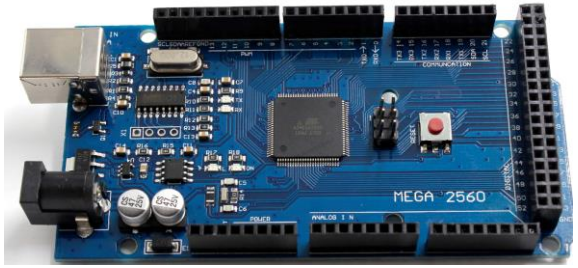
N°	Etape	Exemple
4	**Développer les Services et les API**	• Développer un service sur le Raspberry Pi 4 pour lire les données des capteurs et publier ces données sur AWS IoT Core via MQTT. • Développer des API sur AWS IoT Core pour recevoir les données des capteurs, les stocker dans une base de données, et déclencher des actions en cas de valeurs anormales (par exemple, alertes par e-mail en cas de température trop élevée).
5	**Intégrer les Dispositifs IoT**	• Configuration matérielle: Connecter physiquement les capteurs au Raspberry Pi 4 en utilisant des câbles et des adaptateurs appropriés. • Configuration logicielle: Configurer le Raspberry Pi 4 pour utiliser le protocole MQTT et se connecter à AWS IoT Core. Utiliser les librairies adéquates
6	**Mettre en Place la Connectivité et la Sécurité**	• Configurer la connectivité Wi-Fi sur le Raspberry Pi 4 pour se connecter au réseau local. • Configurer les politiques de sécurité sur AWS IoT Core pour garantir l'authentification des dispositifs et le chiffrement des données.
7	**Déployer et Tester la Plateforme**	• Déployer le système sur site dans l'entrepôt. • Effectuer des tests.

2. Les cartes pour le développement des applications IoT

2.1. Les cartes Arduino

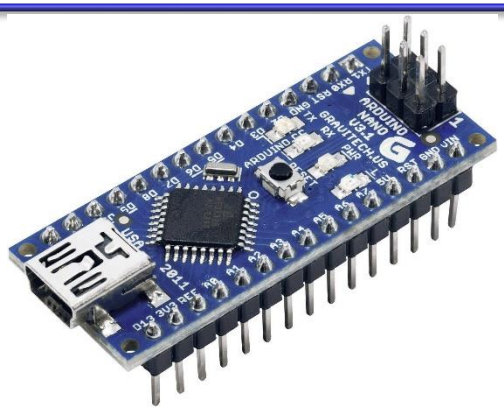
Arduino Uno :

- Microcontrôleur ATmega328P, fréquence de 16 MHz, 32 KB de mémoire flash, 2 KB de SRAM, 14 broches d'entrée/sortie numériques, 6 broches d'entrée analogique, interface USB.
- Polyvalente, convient aux projets de base et intermédiaires, largement utilisée pour l'apprentissage et la prototypage



Arduino Mega :

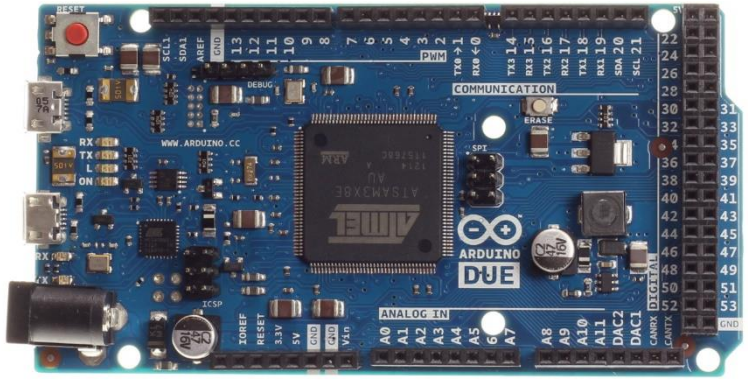
- Microcontrôleur ATmega2560, fréquence de 16 MHz, 256 KB de mémoire flash, 8 KB de SRAM, 54 broches d'entrée/sortie numériques, 16 broches d'entrée analogique, interface USB.
- Grande capacité d'entrées/sorties, convient aux projets complexes nécessitant de nombreuses connexions



Arduino Nano :

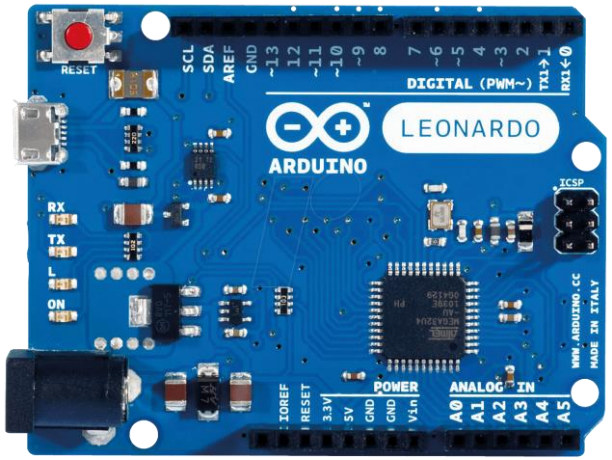
- Microcontrôleur ATmega328P, fréquence de 16 MHz, 32 KB de mémoire flash, 2 KB de SRAM, 14 broches d'entrée/sortie numériques, 8 broches d'entrée analogique, interface USB.
- Compacte et légère, idéale pour les projets embarqués

… Cartes pour le développement des applications IoT / Arduino



Arduino Due :

- Microcontrôleur ARM Cortex-M3, fréquence de 84 MHz, 512 KB de mémoire flash, 96 KB de SRAM, 54 broches d'entrée/sortie numériques, 12 broches d'entrée analogique, interface USB.
- Hautes performances, convient aux applications nécessitant une puissance de calcul plus élevée et une connectivité avancée



Arduino Leonardo :

- Microcontrôleur ATmega32u4, fréquence de 16 MHz, 32 KB de mémoire flash, 2.5 KB de SRAM, 20 broches d'entrée/sortie numériques, 12 broches d'entrée analogique, interface USB.
- Intègre nativement le support USB, adapté aux projets nécessitant des fonctionnalités USB avancées comme les claviers et souris HID

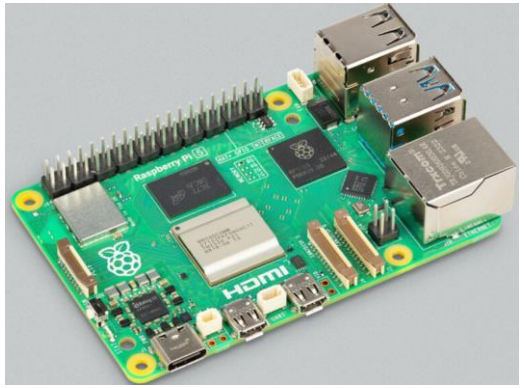
2.2. Les cartes Raspberry Pi

Raspberry Pi 3 Model B+ :

- Processeur ARM Cortex-A53 quad-core 1.4 GHz, 1 GB de RAM, Wi-Fi 802.11ac et Bluetooth 4.2 intégrés, ports HDMI, Ethernet, USB, et GPIO.
- Polyvalente, convient à une large gamme de projets incluant la domotique, les serveurs légers, les médias-centers, etc

Raspberry Pi 4 Model B :

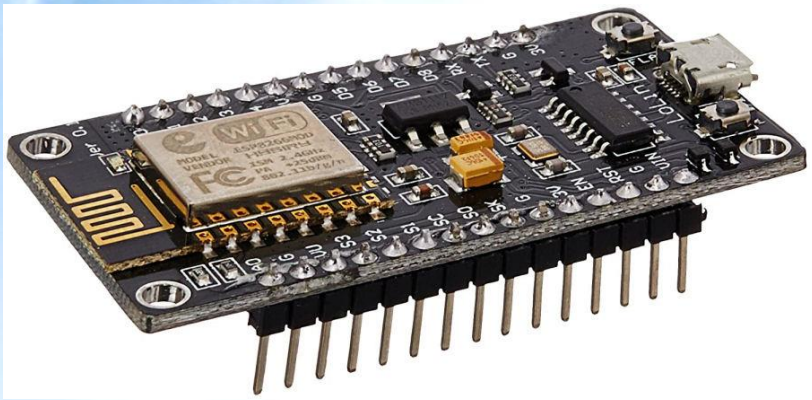
- Processeur ARM Cortex-A72 quad-core 1.5 GHz, 2 GB/4 GB/8 GB de RAM, ports HDMI 4K, USB 3.0, Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0, GPIO étendu.
- Hautes performances, adaptée aux applications exigeantes telles que la bureautique, la programmation avancée, les jeux légers, etc.



Raspberry Pi 5 :

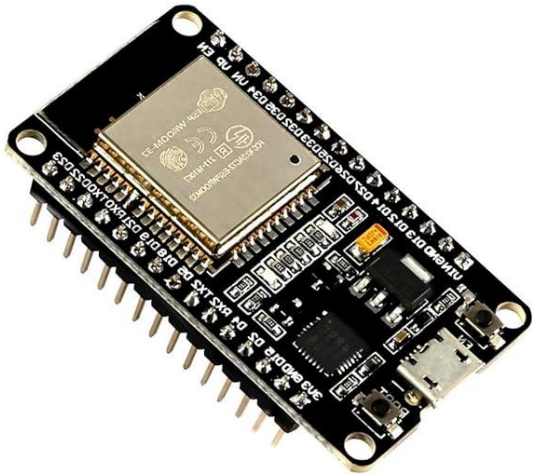
- Processeur ARM Cortex-A7x quad-core ou octa-core, fréquence supérieure à 2 GHz, 4 GB/8 GB de RAM, ports HDMI 4K, USB 3.0/3.2, Wi-Fi 6, Bluetooth 5.2, GPIO étendu.
- Améliorations significatives en termes de puissance de calcul, de connectivité, et de capacités multitâches par rapport au Raspberry Pi 4.

2.3. Les cartes ESP



NodeMCU ESP8266 :

- Microprocesseur : Tensilica 32-bit RISC CPU Xtensa LX106, 80 MHz, Mémoire Flash : 4 MB, SRAM: 64 KB
- E/S disponibles: 16 E/S digitales (DIO), 1 entrée analogique (ADC)
- Interfaces: UARTs, SPI, I2C, Wi-Fi 802.11 b/g/n, USB
- Facile à utiliser, prise en charge de l'IDE Arduino.



ESP32-WROOM-32:

- Microprocesseur : Tensilica LX6 Dual-Core, 240 MHz,SRAM : 512 kB, Mémoire Flash : 4 MB
- E/S disponibles : 22 E/S digitales, 2 x sorties analogiques (DAC), 15 x entrées analogiques (ADC)
- Interfaces : I2C, SPI, UART, WiFi 802.11 b/g/n 2,4 GHz, USB
- Pour les applications nécessitant un grand nombre de capteurs analogiques

2.4. Les cartes NVIDIA Jetson:

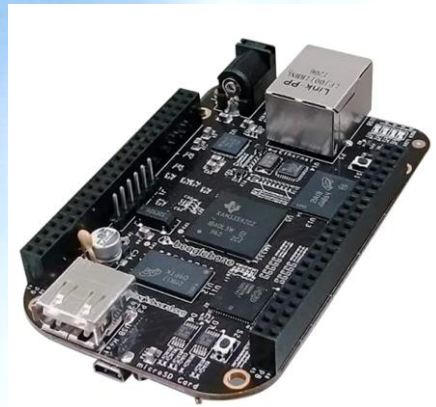
NVIDIA Jetson Nano :

- Processeur : NVIDIA Maxwell avec GPU NVIDIA Maxwell à 128 cœurs CUDA, 921 MHz
- Mémoire : 4 GB LPDDR4
- Stockage : Connecteur microSD pour le stockage externe
- Interfaces : HDMI, DisplayPort, USB 3.0, Ethernet, GPIO, MIPI CSI-2, MIPI DSI
- Plateforme d'apprentissage en IA, adaptée aux projets DIY (Do It Yourself) et éducatifs.

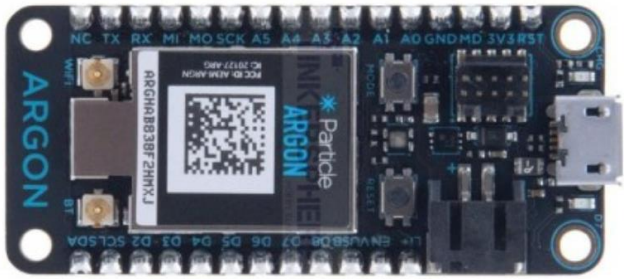
NVIDIA Jetson Xavier NX :

- Processeur : NVIDIA Volta avec GPU NVIDIA Volta à 384 cœurs CUDA
- Performances : 21 TOPS (Tera Operations Per Second)
- Mémoire : 8 GB LPDDR4x
- Stockage : Connecteur microSD et emplacement M.2 pour le stockage externe
- Interfaces : HDMI, DisplayPort, USB 3.0, Ethernet, GPIO, MIPI CSI-2, MIPI DSI
- Hautes performances dans un facteur de forme compact, idéal pour l'IA embarquée et les applications industrielles.

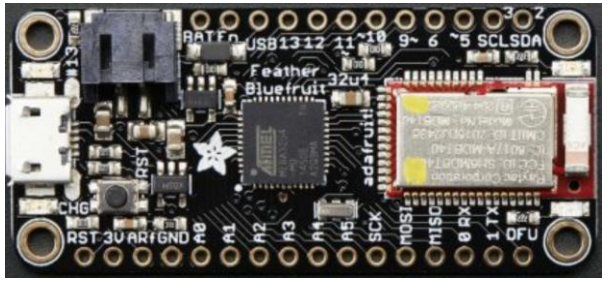
2.5. Autres cartes:



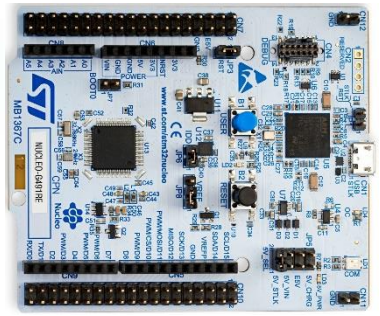
BeagleBone Black



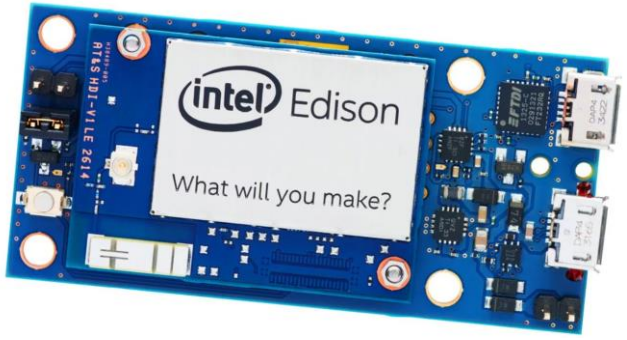
Particle Argon



Adafruit Feather



STM32 Nucleo



Intel Edison

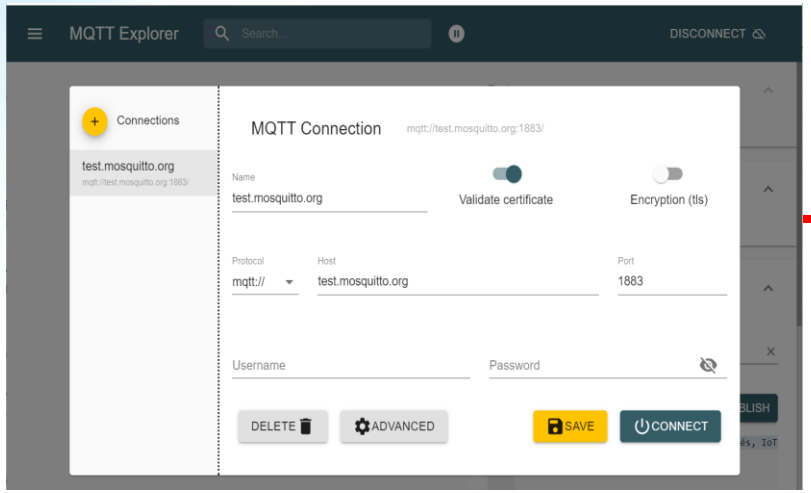
NXP LPCXpresso

Et Bien d'autres….

3. Application N°1: Manipuler avec le Protocole MQTT

- OBJECTIF: Se familiariser avec les fondamentaux du protocole MQTT: Broker, Topic, Publish, Subscribe, etc.
- Ceci en utilisant les applications: MQTT Explorer et Test Client.

L'application MQTT Explorer à installer



Publish data to « Topic »

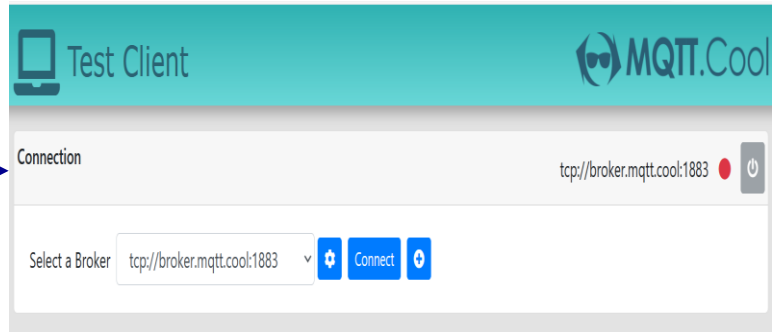
MQTT Broker

test.mosquitto.org

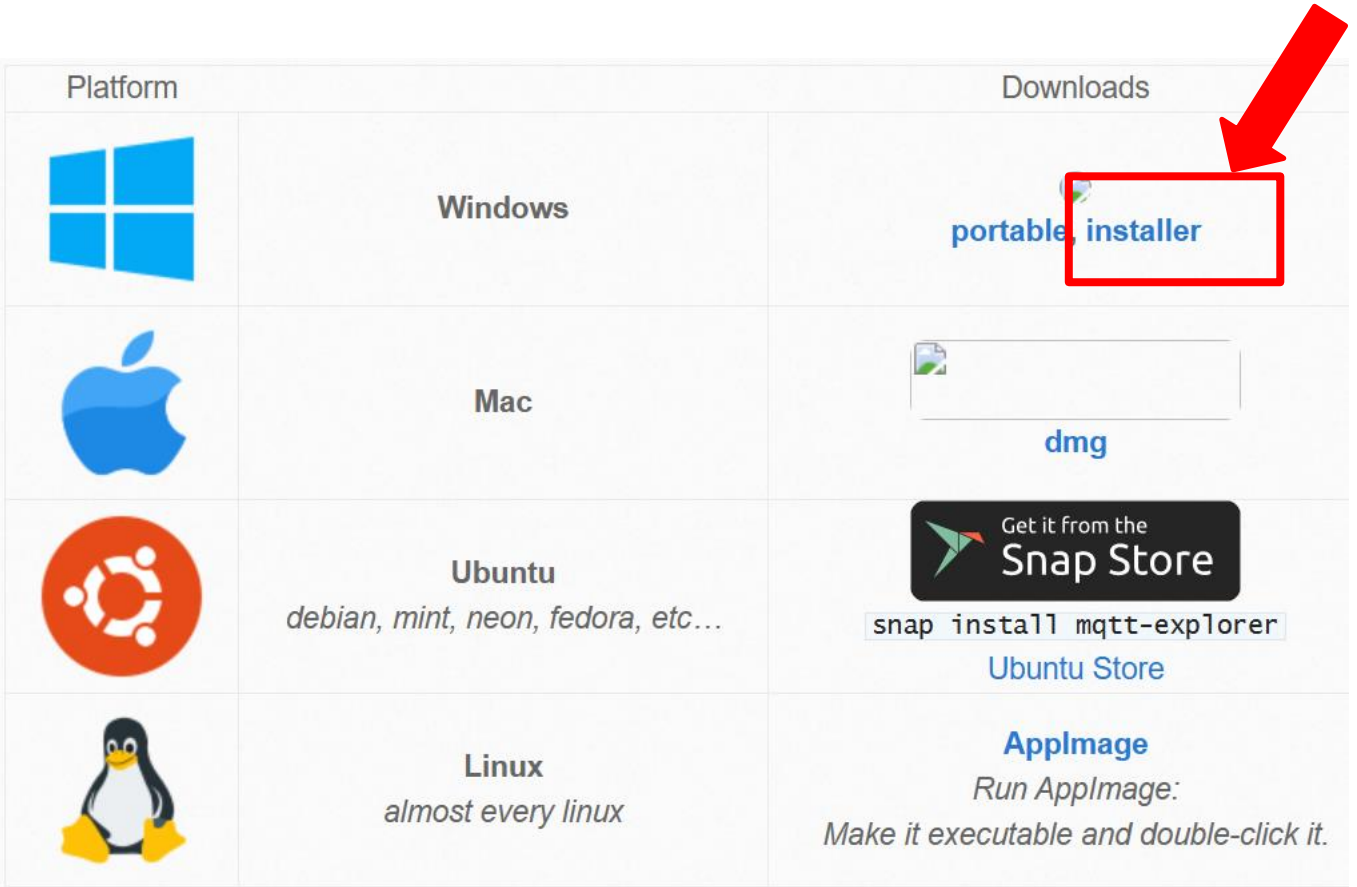
Sbuscribed to « Topic »

Published data

L'application Test Client (à utiliser en ligne)

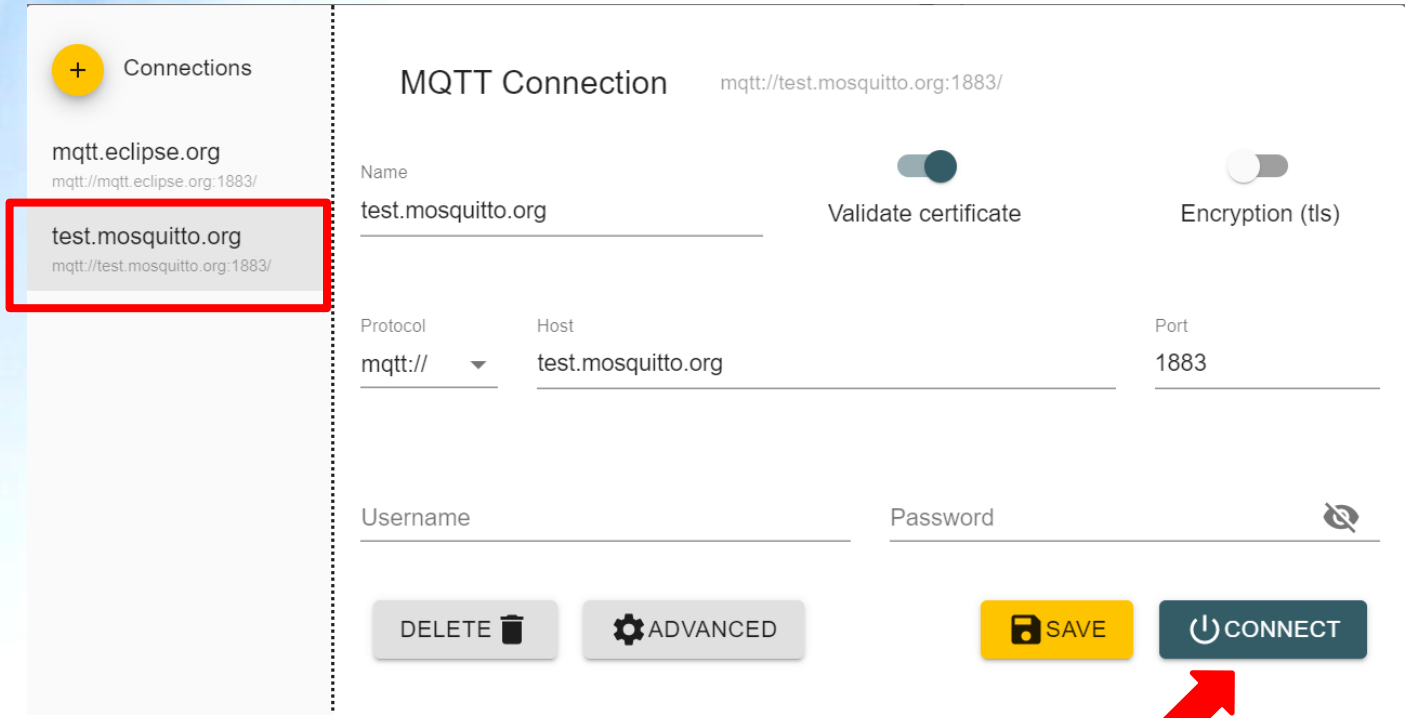


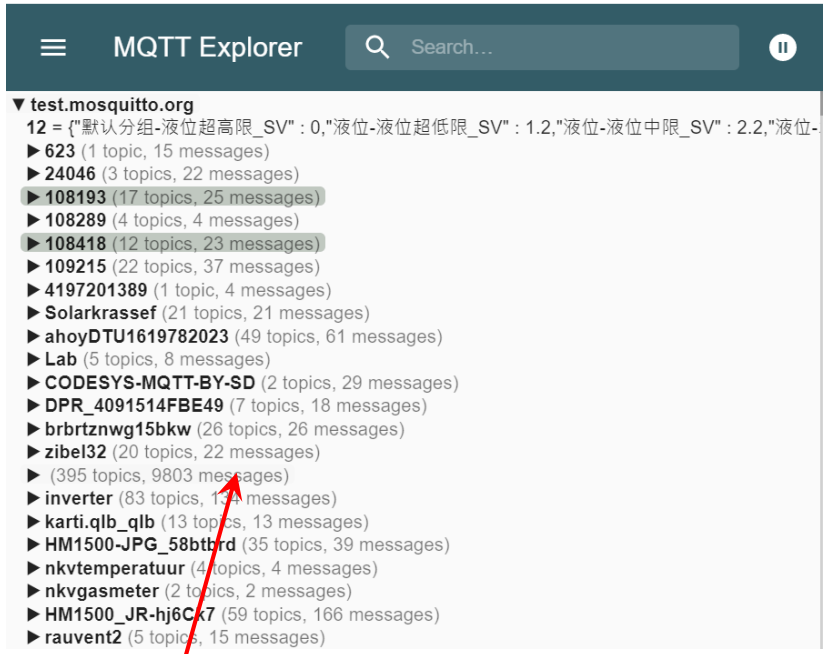
1 Télécharger et installer MQTT Explorer: https://mqtt-explorer.com/



Platform		Downloads
	Windows	portable, installer
	Mac	dmg
	Ubuntu *debian, mint, neon, fedora, etc…*	Get it from the Snap Store `snap install mqtt-explorer` Ubuntu Store
	Linux *almost every linux*	**AppImage** *Run AppImage:* *Make it executable and double-click it.*

2 Se connecter au serveur MQTT: test.mosquitto.org:





Appuyer sur CONNECT

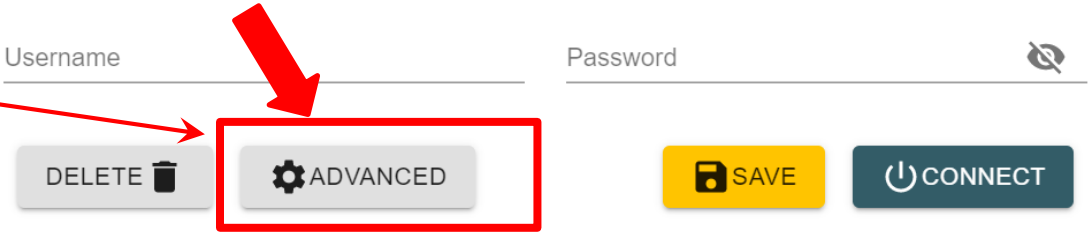
Vous devez vous apercevoir plusieurs choses qui se défilent dans ce coin (c'est les topics en cours de publication sur ce serveur)!

3 Crée un nouveau topic (sujet):

a) **Commencer par vous vous déconnecter**

b) **Cliquer sur « ADVANCED »**

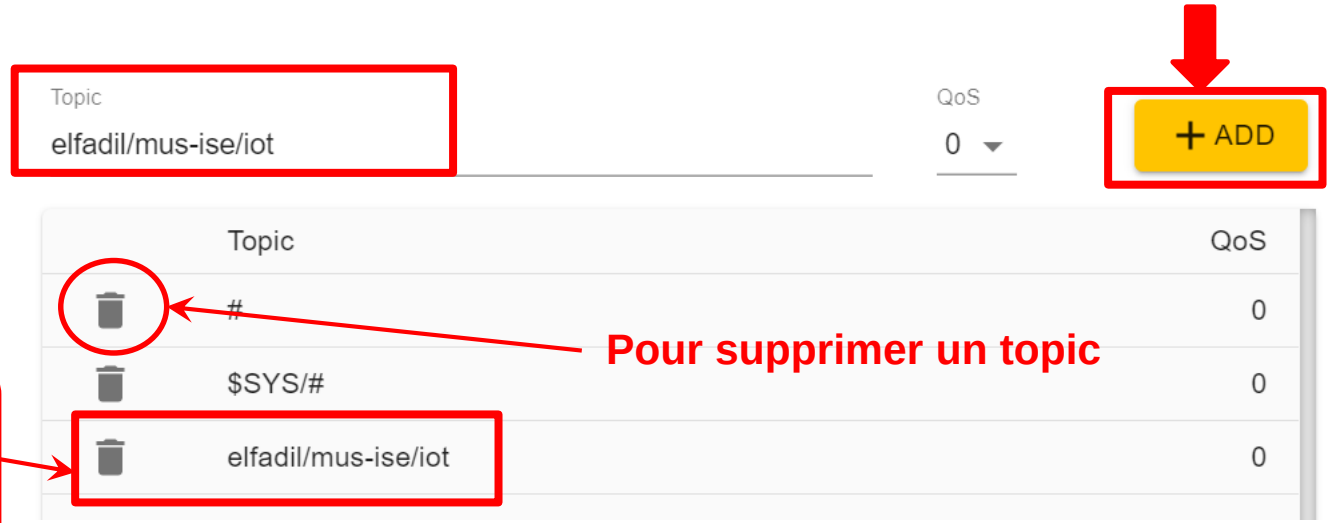
c) **Créer un « Topic »: votrenom/mus-ise/iot**



Exemple: **elfadil/mus-ise/iot**

Rq: Veiller à créer des topics différents des uns des autres!!

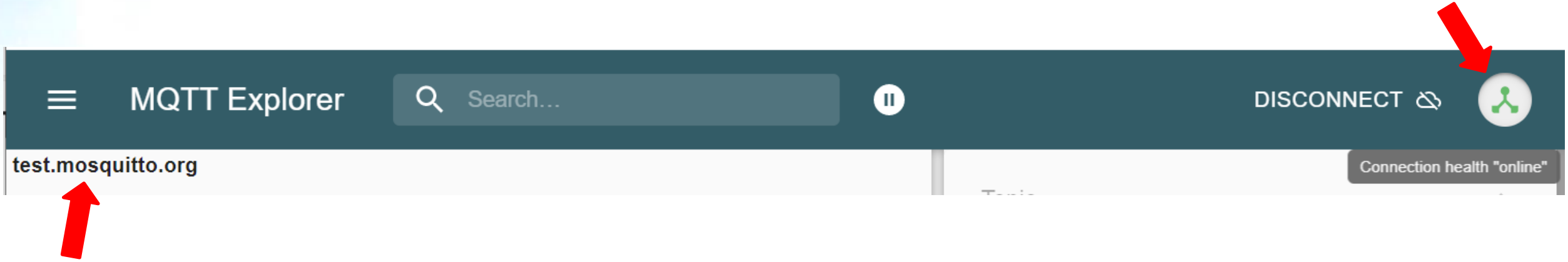
d) **Cliquer sur « ADD »**



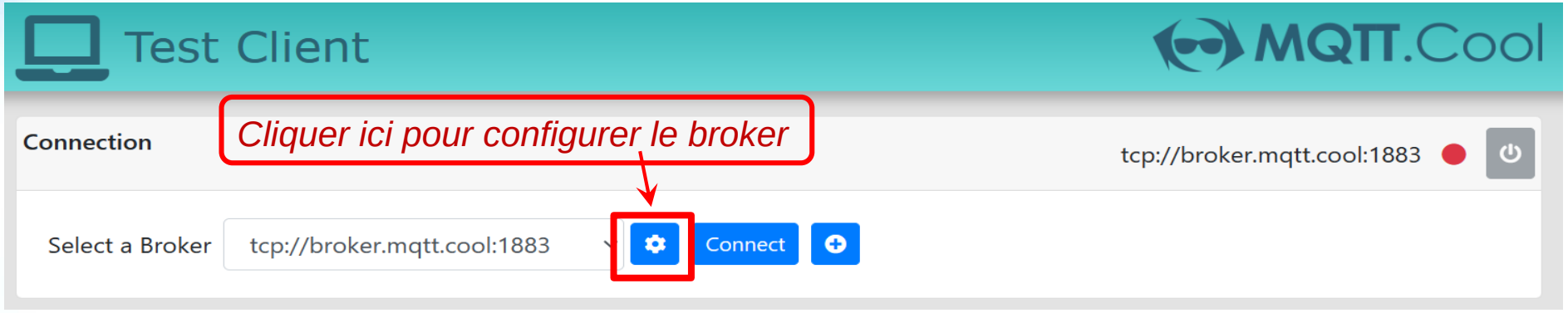
Votre topic vient d'être ajouté.
Vous pouvez supprimer les topics indésirables en cliquant sur corbeille à côté!

4 Cliquer sur « BACK » puis «SAVE » puis « CONNECT »:

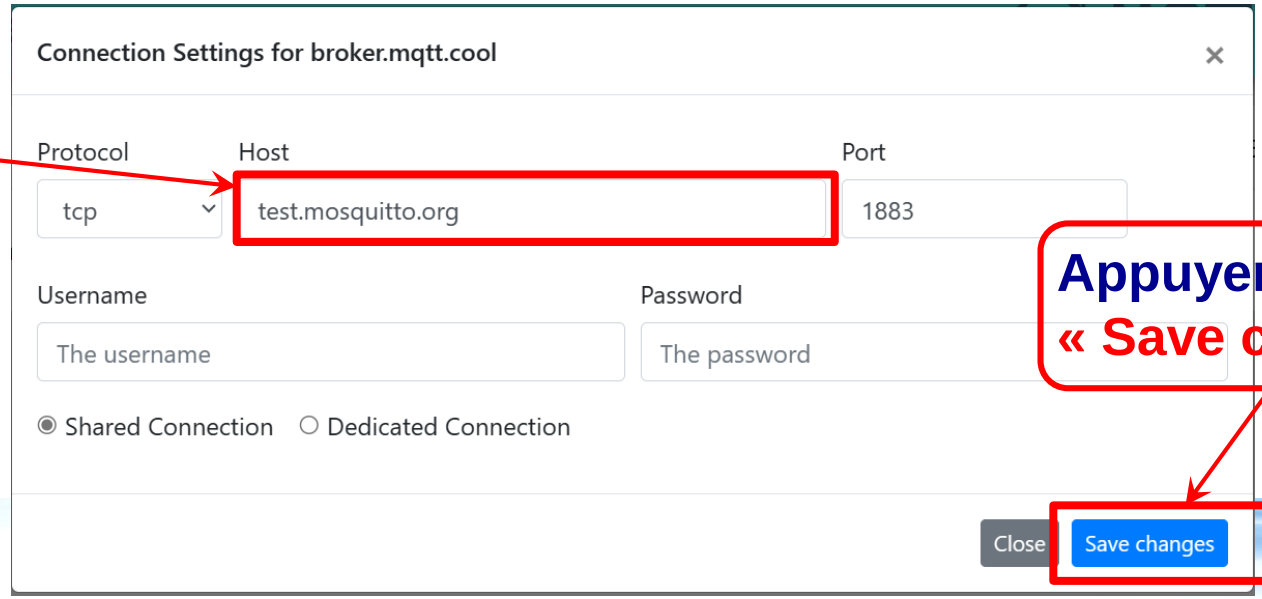
Vérifier que la connexion au serveur est bien établie!!!



5 **Pour réaliser des tests, utiliser l'application en ligne « Test Client »: https://testclient-cloud.mqtt.cool/**



Saisir le nom du serveur: test.mosquitto.org



Appuyer ensuite sur « Save changes »

…Application N°1: Manipuler avec le Protocole MQTT

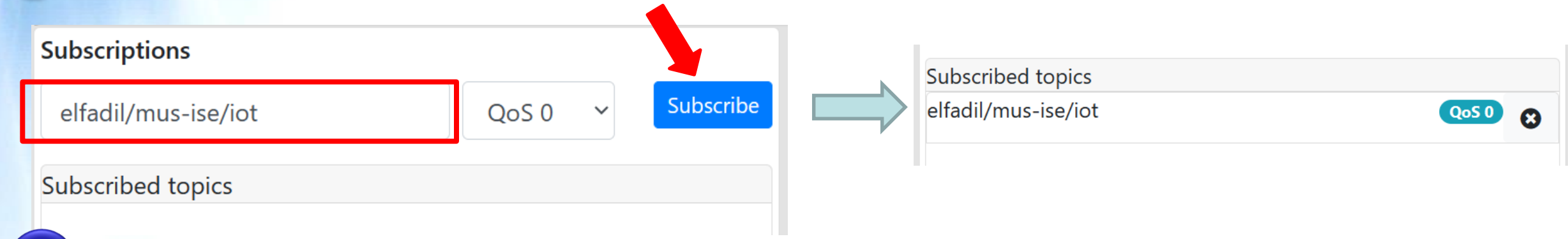
Ce bouton passe au vert quand la connexion au serveur est réussie

Appuyer enfin sur « connect »

Ici la section pour s'abonner « Subscribe »

Ici la section pour publier « Publish »

6 S'abonner au topic que vous avez créé précédemment : elfadil/mus-ise/iot

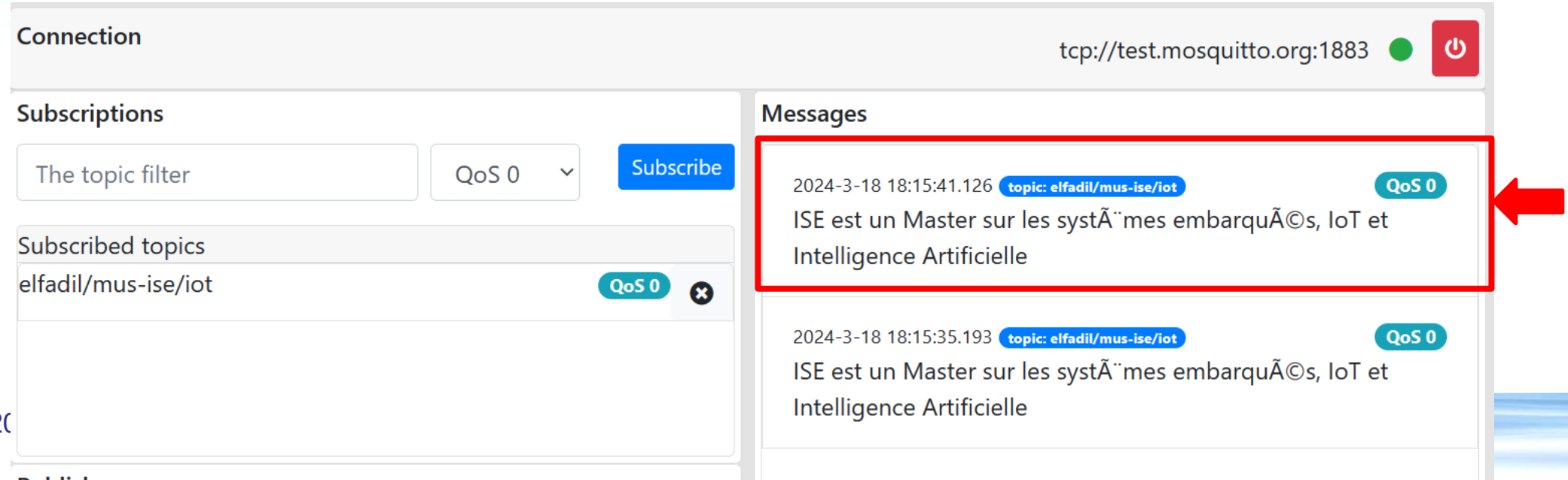


7 Revenez à l'application MQTT Explorer : Dans la section « Publish », saisir le topic: elfadil/mus-ise/iot, et publier un message, Exemple:

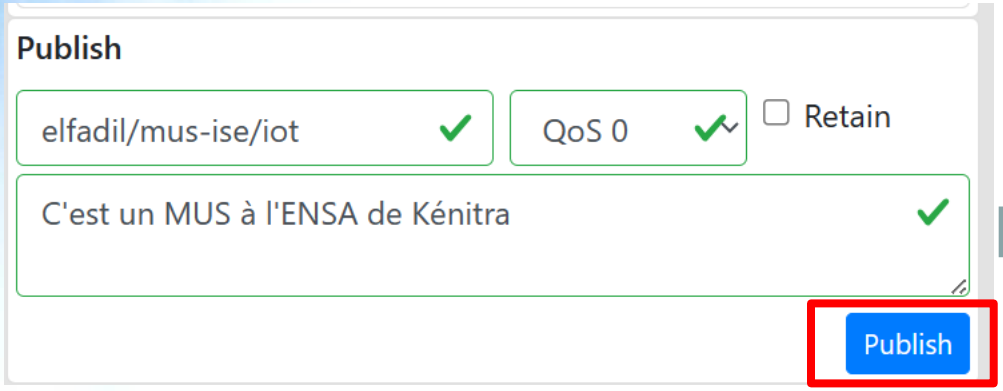
« ISE est un Master sur les systèmes embarqués, IoT et Intelligence Artificielle »

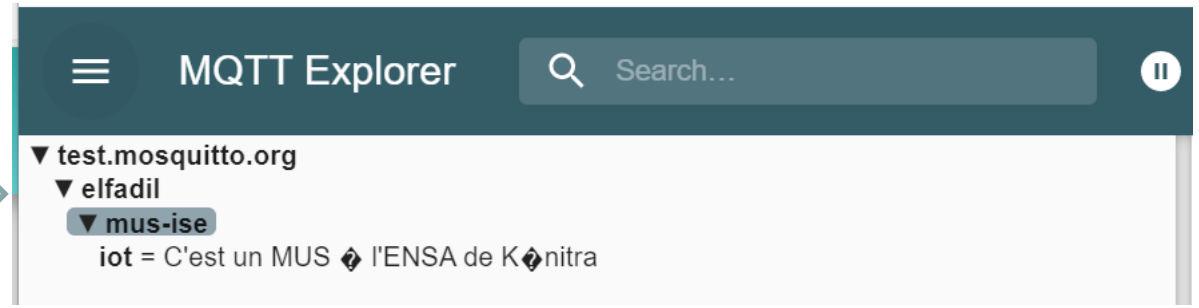
8 Dans la fenêtre gauche de MQTT Explorer Vérifier que le message a été publié:

9 Dans l'application Test Client, vérifier que le message a été reçu:



10 Publier cette foi un message depuis « Test Client ». Utiliser la section « Publish »:

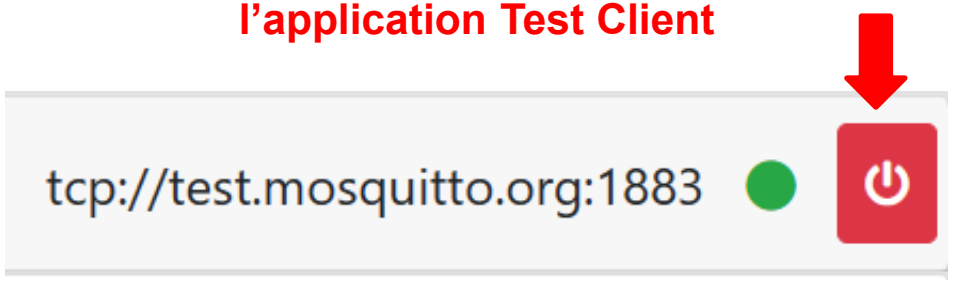




11 Pour finir, se déconnecter du Serveur MQTT:

Appuyer sur DICONNECT pour l'application MQTT Explorer

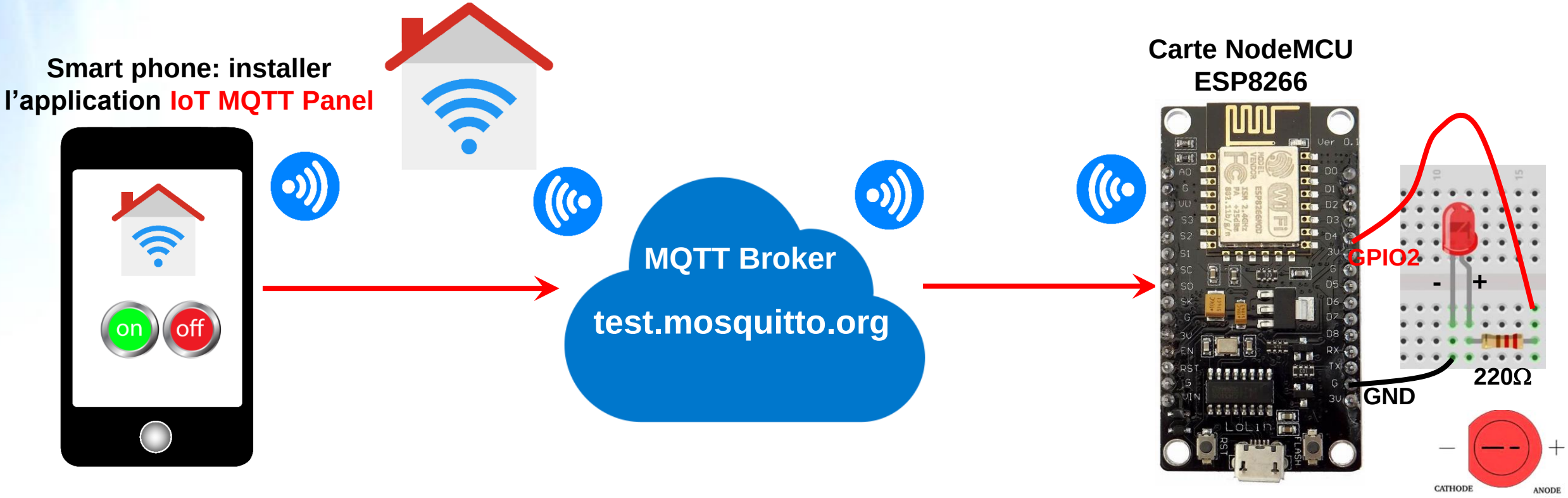
Appuyer sur Bouton Rouge pour l'application Test Client



4. Application N°2: Commande à Distance d'une LED

4.1. Objectif:

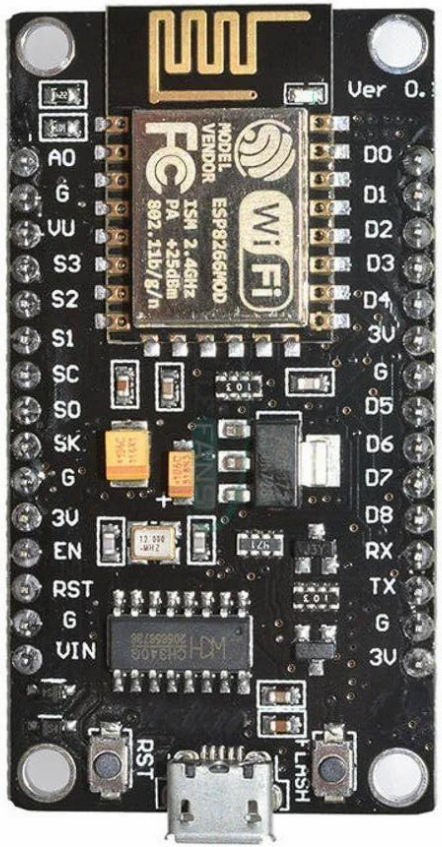
- Commande à distance d'une LED (ON/OFF).
- Utilisation de la carte NodeMCU ESP8266 et l'application Android (MQTT Dashboard »



4.2. La carte ESP

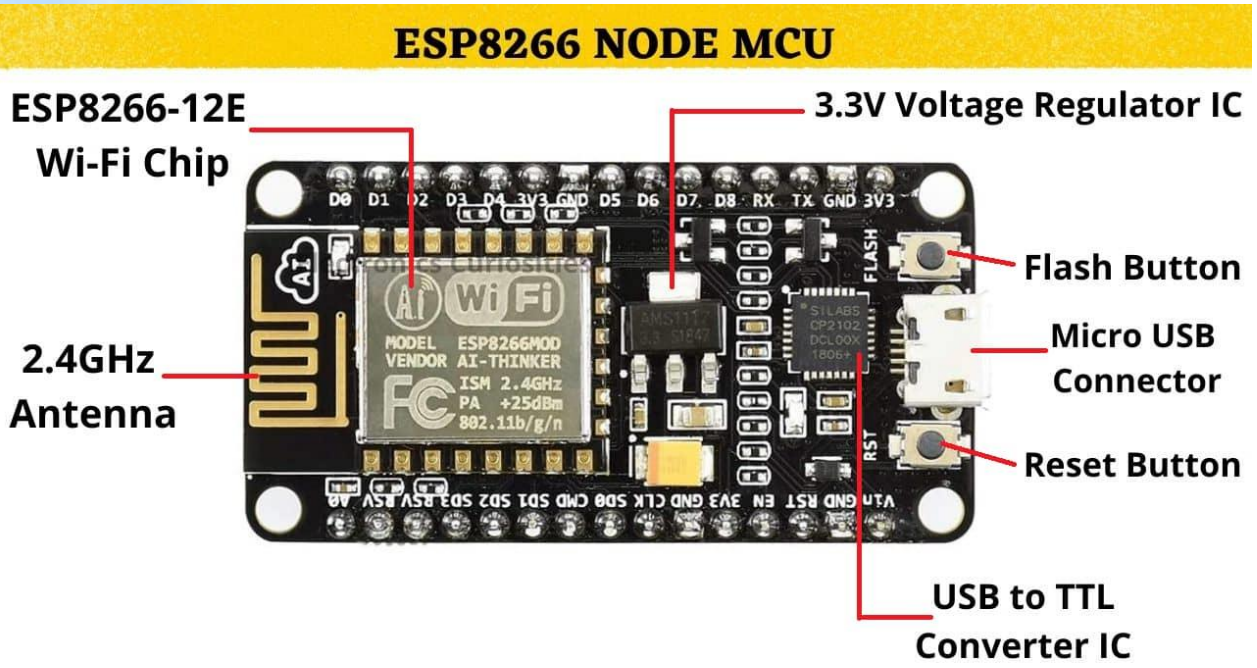
- Carte de développement dédié à l'internet des objets (IoT) et les applications embarquées.
- C'est un (**SoC**) system on chip doté de communications sans fil Wifi et Bluetooth.
- Fabriquée par la société chinoise Espressif Systems.
- Support de développement : Arduino IDE, Python,
- Plusieurs variantes des cartes ESP:
  - **ESP8266** : Processeur monocœur, vitesse d'horloge inférieure. Idéal pour les tâches plus simples et moins exigeantes. Prend en charge une seule broche ADC 10 bits
  - **ESP32** : Processeur double cœur, vitesse d'horloge plus élevée. Convient aux applications plus complexes. Prend en charge les mesures analogiques sur 18 canaux (broches compatibles analogiques) et le Bluetooth. Plus puissant que ESP8266.

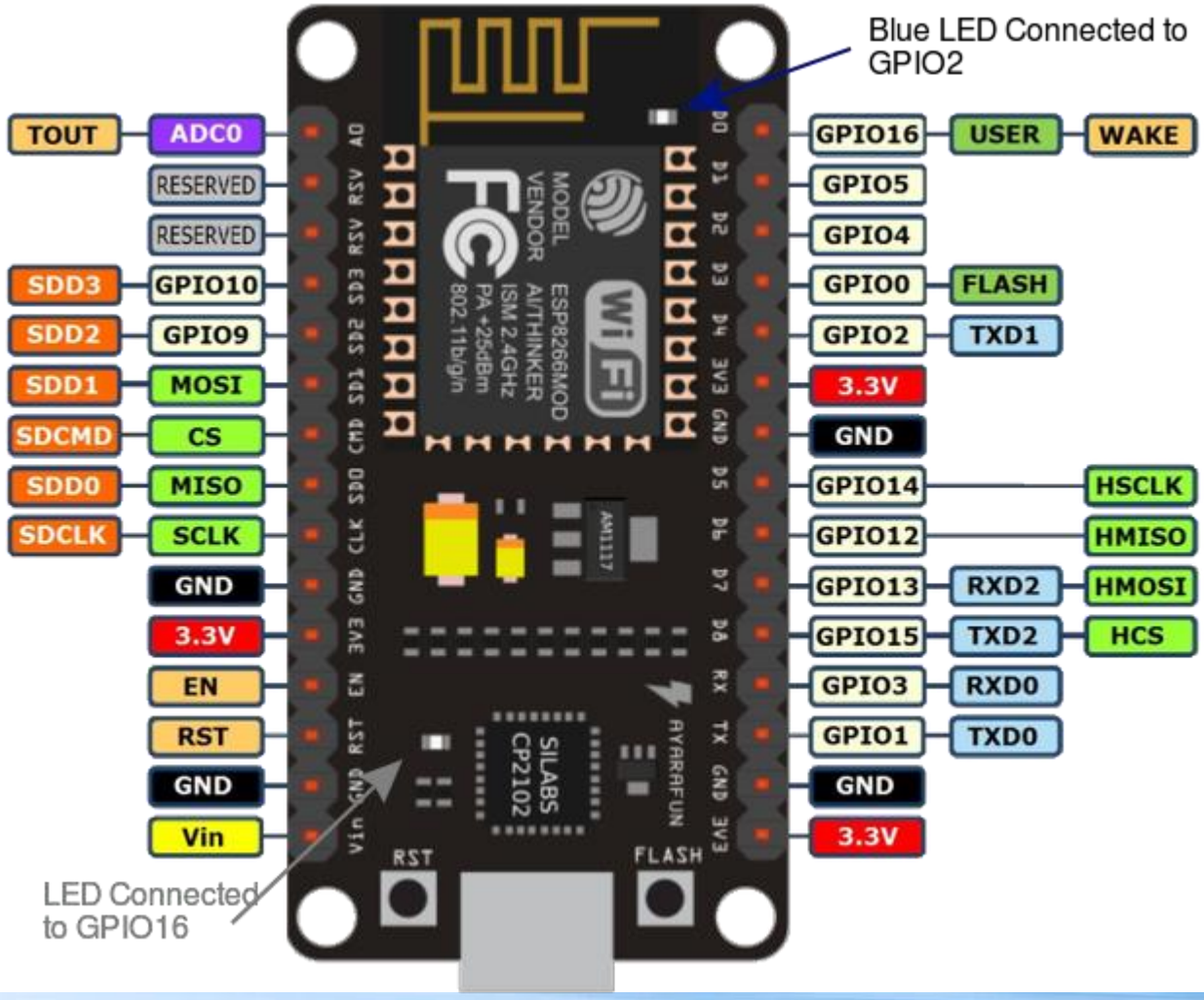
…Application N°2: Commande à Distance d'une LED / Carte ESP



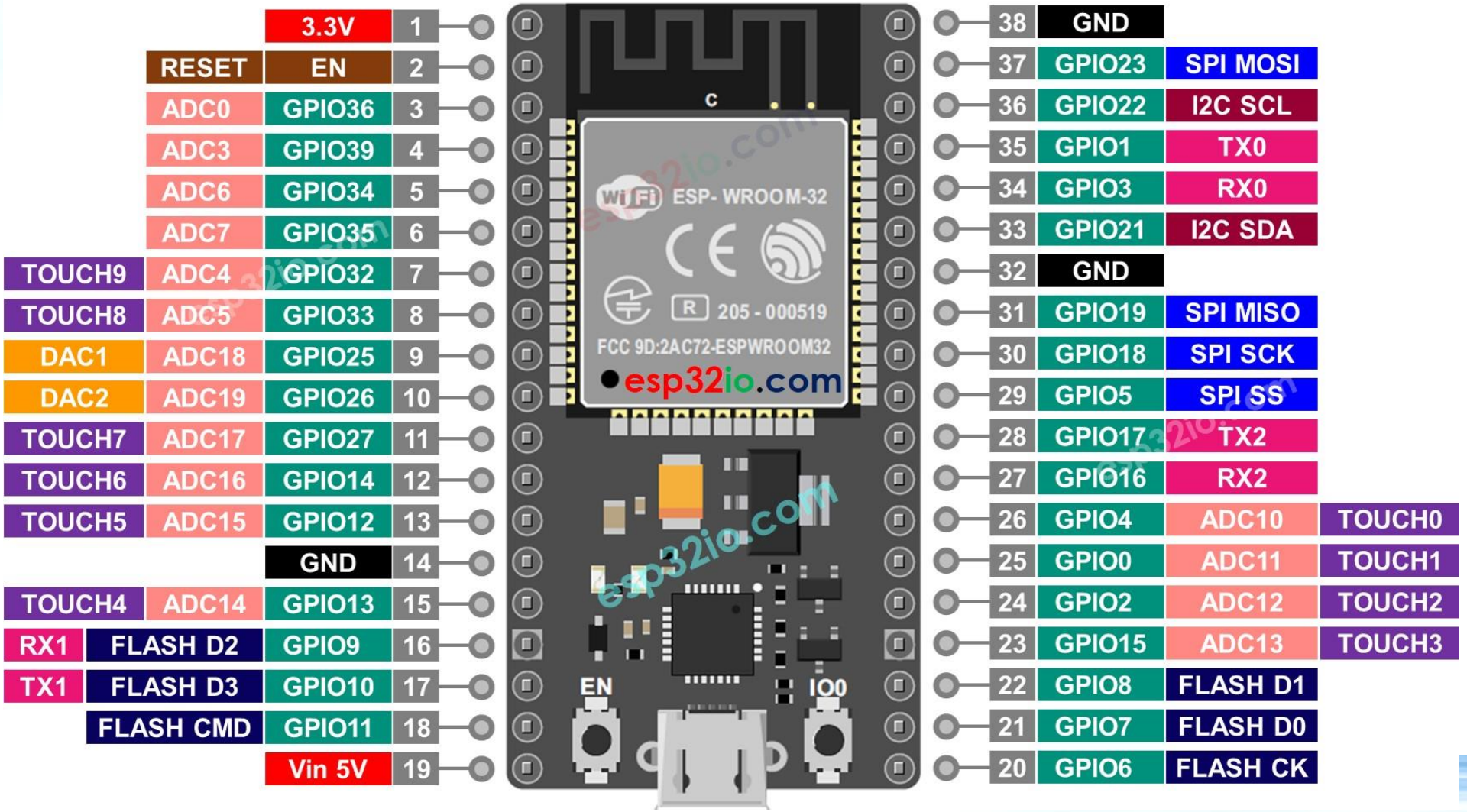
	ESP8266	ESP32
MCU	Xtensa Single-core 32-bit	Xtensa Dual-Core 32-bit
Wi-Fi	HT20 (802.11 b/g/n)	HT40 (802.11 b/g/n)
Bluetooth	Non	Oui (4.2 et BLE)
Frequence	80Mhz	160 a 240 Mhz
SRAM	64kB	520kB + 10 pour RTC
DRAM	96kB	328kB
Flash	4mB	4mB (suivant versions)
GPIO	17	36
Analog GPIO	1	18

Broches de la carte ESP8266 (NodeMCU):





Broches de la carte ESP WROOM-32:

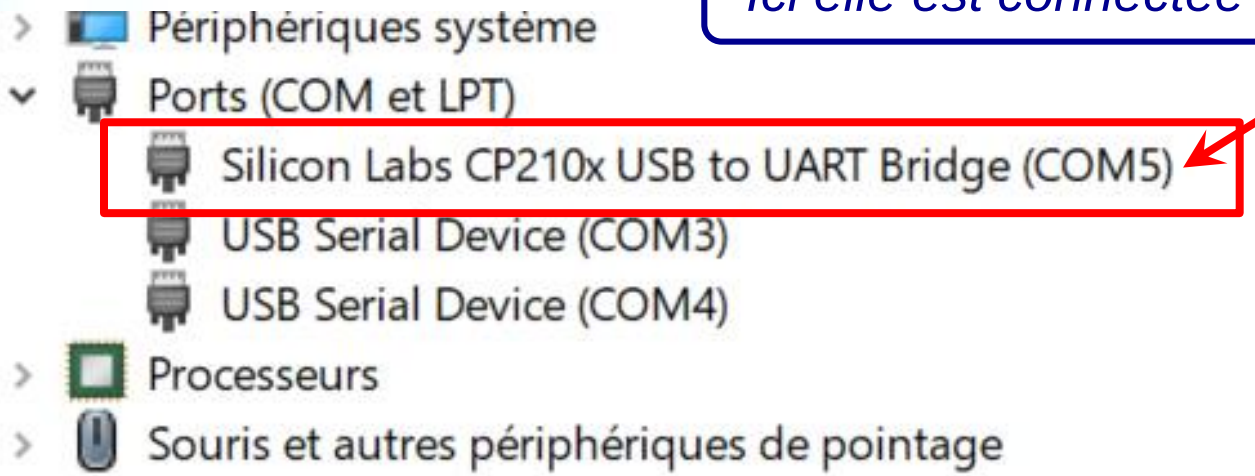


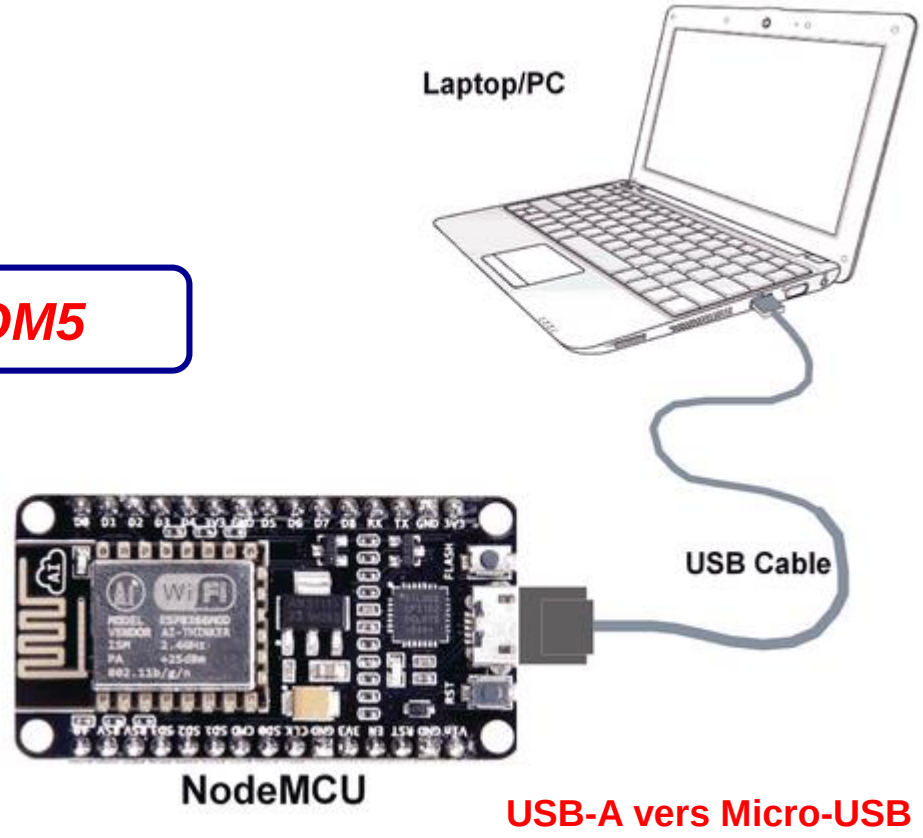
4.3. Etapes de développement de l'application:

1 **Connecter le NodeMCU avec le PC (vérifier que le carte est bien détectée. Sinon installer ses drivers):**

Ouvrir le gestionnaire de périphériques. Dans Ports (COM et LPT) vous devriez apercevoir la carte détectée.

*Ici elle est connectée sur le **COM5***





USB-A vers Micro-USB

Pour installer le driver CP210x USB to UART Bridge:

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

2 Préparation de l'environnement de développement (IDE Arduino):

a) Télécharger et Installer l'IDE Arduino sur votre ordinateur pour programmer le NodeMCU

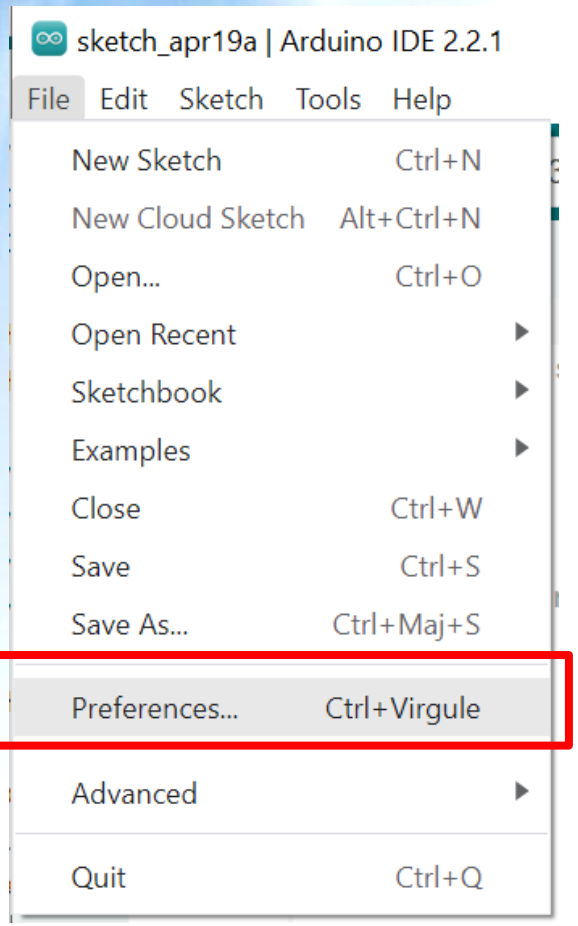
https://www.arduino.cc/.

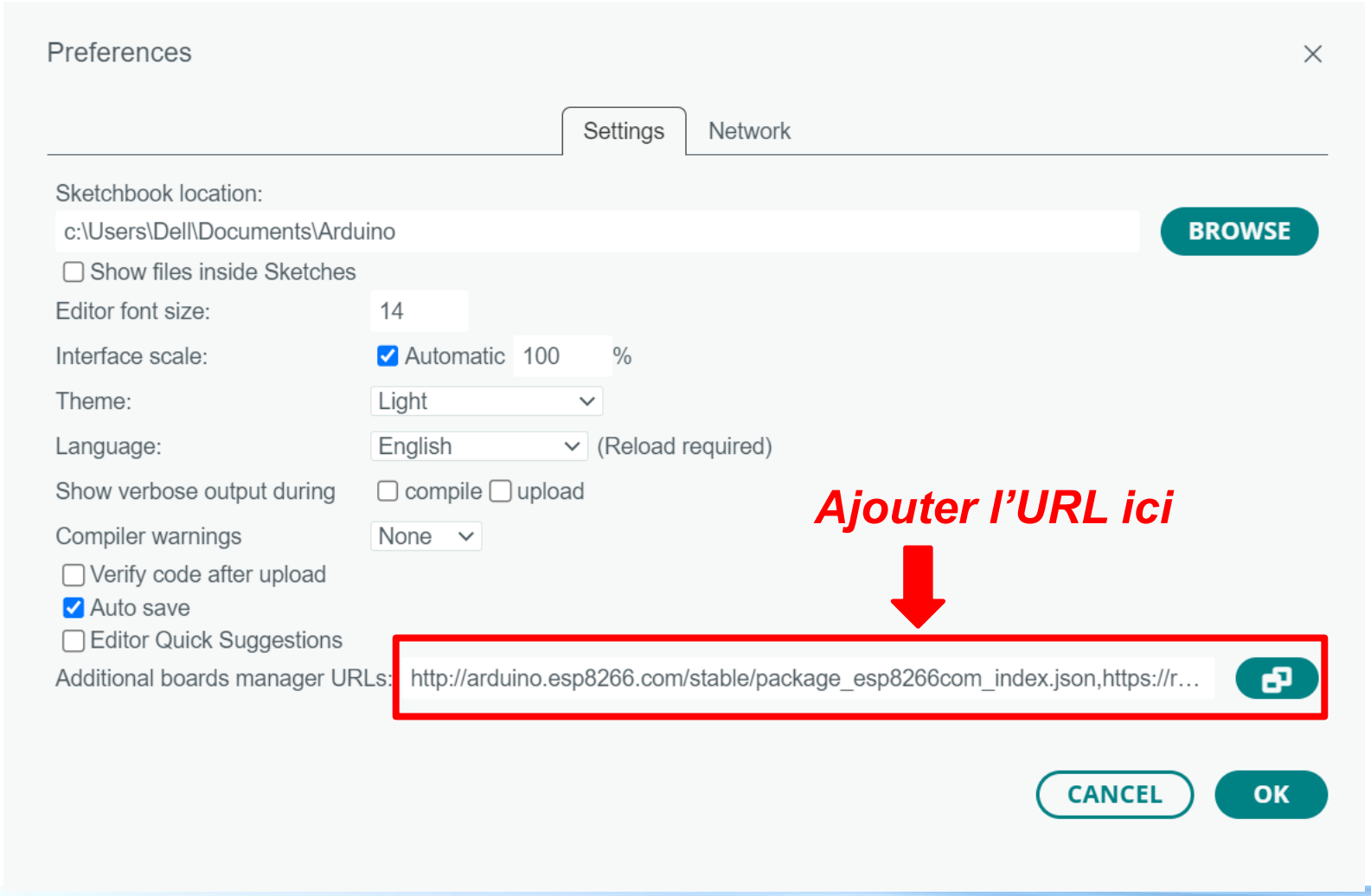
b) Ouvrir un nouveau fichier à partir de l'IDE d'Arduino.

c) Installer les packages de ESP8266: Dans préférences ajouter l'URL suivant:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

…Application N°2: Commande à Distance d'une LED / Etapes de développement



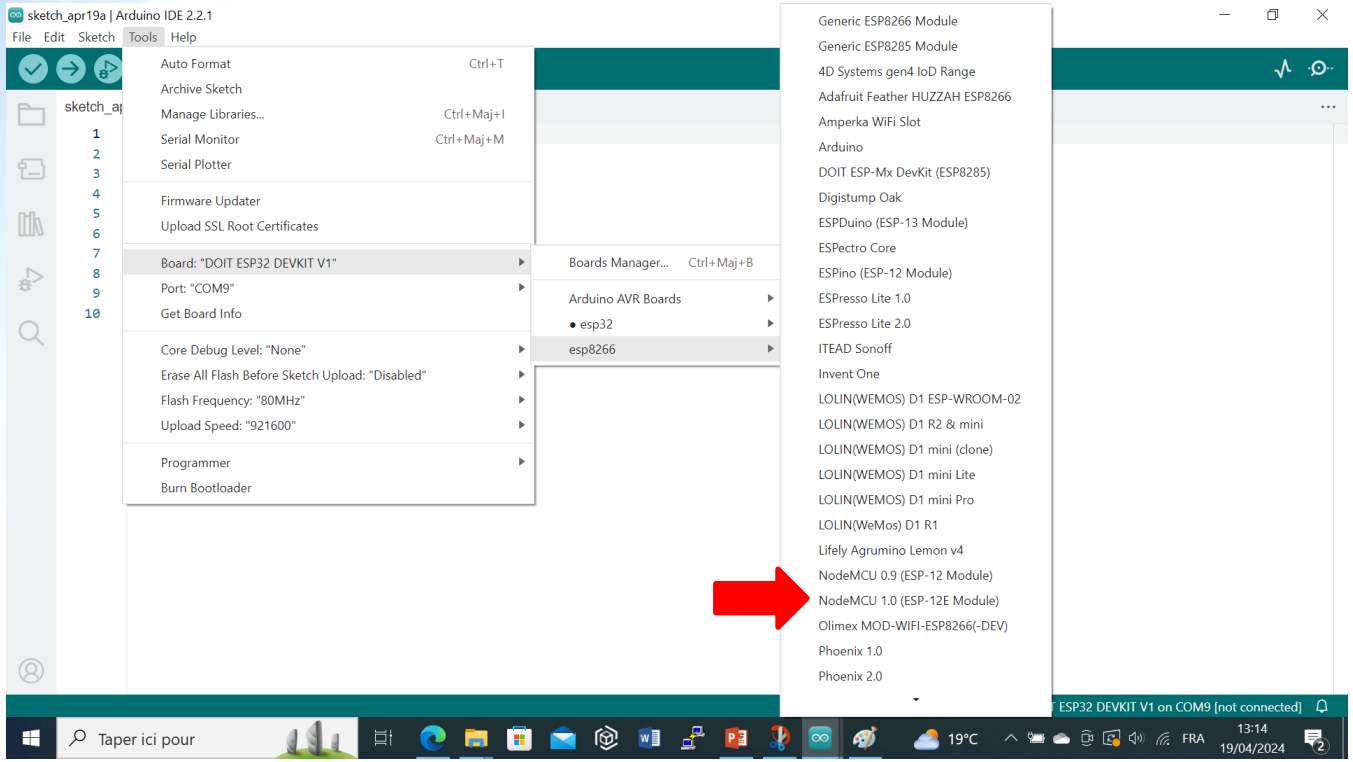


d) Dans "Tools", ouvrir "Boards Manager" :

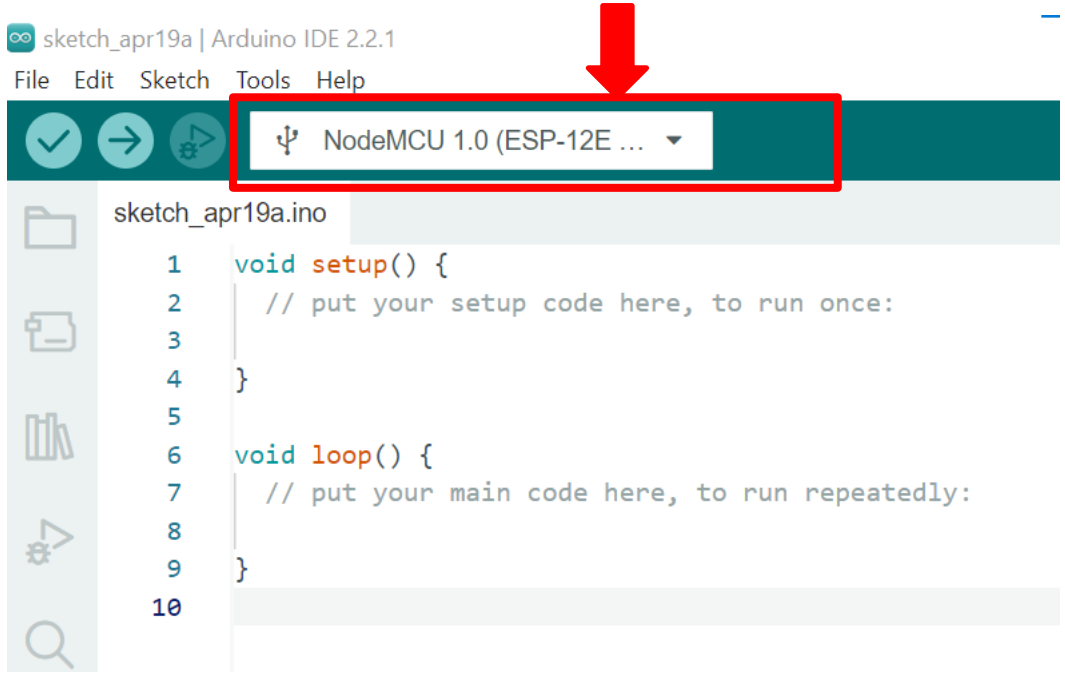
Dans le menu taper: esp8266, puis installer ses packages

e) Une fois l'installation terminée, selectionner le NodeMCU 1.0 parmi les cartes ESP8266:

Tools > Board > ESP8266 > NodeMCU 1.0.



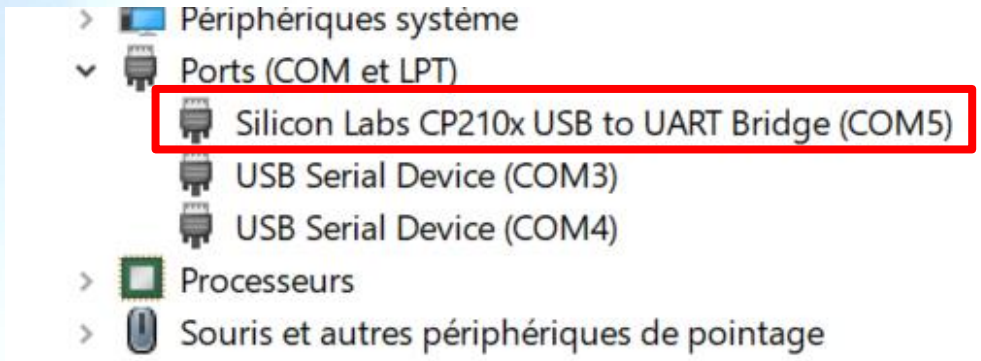
La carte de développement utilisée apparaît dans ce menu.

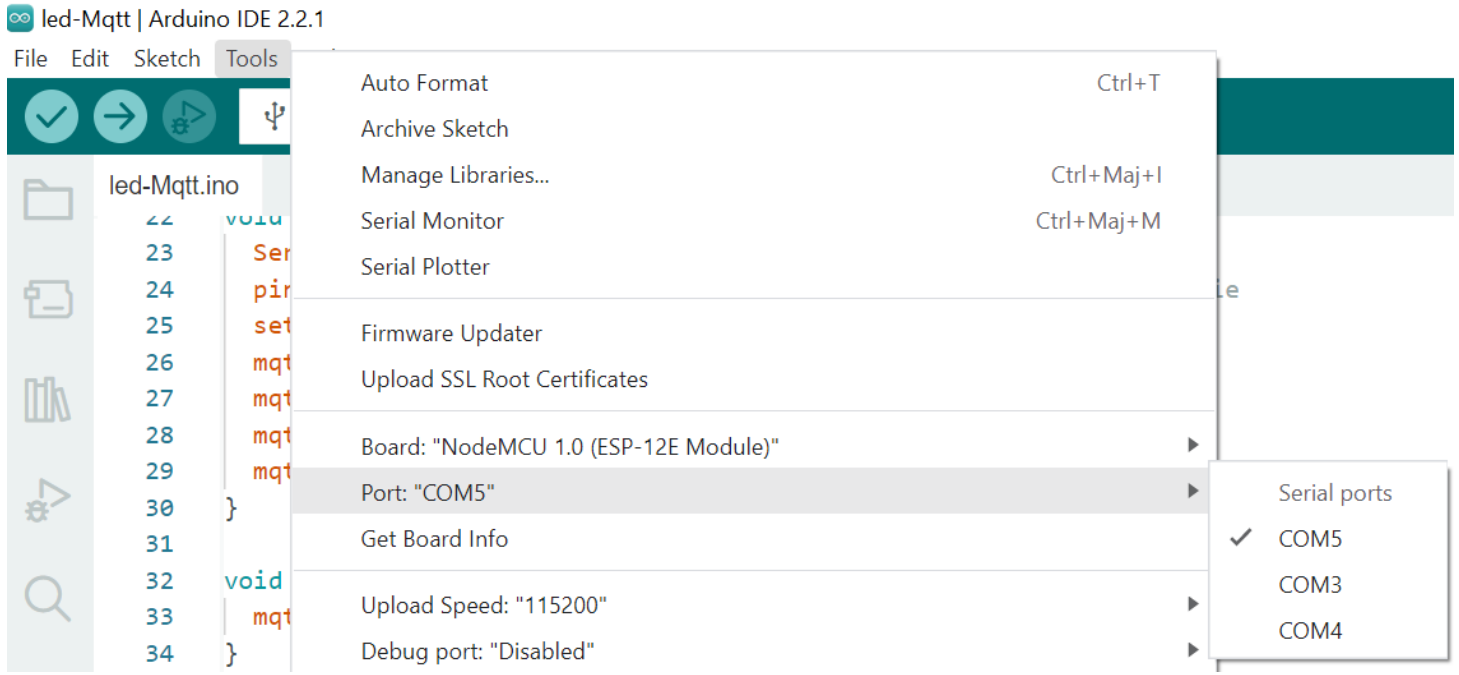


f) Selectionner le port sur lequel la carte est connectée.

Dans notre exemple il s'agit de CMOM5:

Tools > Port > COM5





g) Installer les bibliothèques suivantes:

- *Bibliothèque pour utiliser les fonctionnalités WiFi sur le NodeMCU (qui est basé sur l'ESP8266) :* ***ESP8266WiFi.h***
- *Bibliothèque de gestion de MQTT:* ***PubSubClient.h***

Exemple pour installer la bibliothèque ESP8266WiFi

1) *Allez dans le menu "Croquis" (**Sketch**) > "Inclure une bibliothèque" (**Include Library**) > "Gérer les bibliothèques" (**Manage Libraries**).*
2) *Dans la fenêtre de gestion des bibliothèques, recherchez "**ESP8266WiFi**".*
3) *Cliquez sur le résultat de la recherche ("ESP8266WiFi" par ESP8266 Community) pour ouvrir les détails de la bibliothèque.*
4) *Cliquez sur le bouton "Installer" (**Install**) pour installer la bibliothèque ESP8266WiFi dans votre IDE Arduino.*

3 Elaboration du programme en utilisant l'IDE Arduino

***a) Inclure les librairies pour se connecter au Wifi et au serveur MQTT**: ESP8266WiFi et PubSubClient*

b) Définir la broche et l'état de la LED:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const int LED = 2; // GPIO 2 correspondant à D4 sur la carte NodeMCU
bool ledState = false;
```

c) Initialiser la connexion Wi-Fi:

```
// Paramètres du WiFi
const char* ssid = « Nom-du-Wifi"; // Remplacer par le nom de votre Wifi
const char* password = « Mot-de-passe"; // Remplacer par son mot de passe
```

d) Configurer les paramètres de connexion du Broker MQTT:

```
// Paramètres du Broker MQTT
const char *mqtt_broker = "test.mosquitto.org";  // broker mosquitto
const char *mqtt_topic = « ISE/Etudiant/led1"; //topic : remplacer Etudiant par votre nom
const char *mqtt_username = ""; // Laissez vide si le broker n'exige pas d'authentification
const char *mqtt_password = ""; // Laissez vide si le broker n'exige pas d'authentification
const int mqtt_port = 1883;  // port MQTT (TCP)
```

e) Initialiser les clients Wi-Fi et MQTT:

```
WiFiClient espClient;
PubSubClient mqtt_client(espClient);
```

f) Se connecter au Wi-Fi:

```
void connectToWiFi() {
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to the WiFi network");
}
```

g) Se connecter au serveur MQTT et s'abonnez au topic :

Nous utilisons la méthode mqtt_client.connect() pour nous connecter au serveur MQTT, puis mqtt_client.subscribe() pour nous abonner à un topic et publier un message de test en cas de connexion réussie.

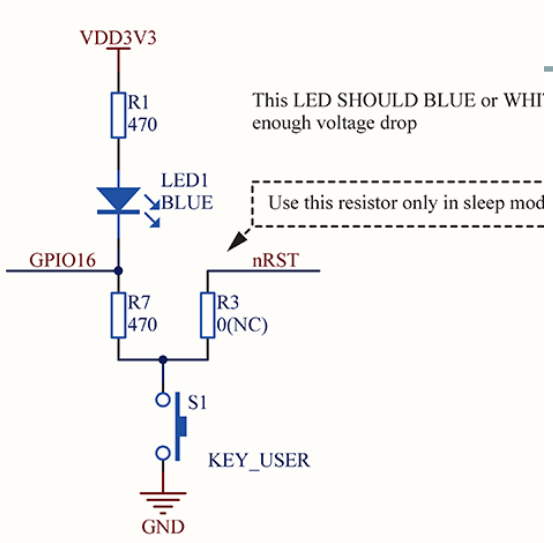
```
void connectToMQTTBroker() {
    while (!mqtt_client.connected()) {
        String client_id = "esp8266-client-" + String(WiFi.macAddress());
        Serial.printf("Connecting to MQTT Broker as %s.....\n", client_id.c_str());
        if (mqtt_client.connect(client_id.c_str(), mqtt_username, mqtt_password)) {
            Serial.println("Connected to MQTT broker");
            mqtt_client.subscribe(mqtt_topic);
            // Publish message upon successful connection
            mqtt_client.publish(mqtt_topic, "Hi, this is the Embedded Systems Master from ENSA");
        } else {
            Serial.print("Failed to connect to MQTT broker, rc=");
            Serial.print(mqtt_client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
```

h) Écrire la fonction de Callback:

Lorsque le client MQTT reçoit un message, nous devons allumer/Eteindre la LED.

```
void mqttCallback(char *topic, byte *payload, unsigned int length) {
    Serial.print("Message received on topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    String message;
    for (int i = 0; i < length; i++) {
        message += (char) payload[i];  // Convert *byte to string
    }
    Serial.println(message); // afficher le message MQTT reçu

    if (message == "on") {
      digitalWrite(LED, LOW); // logique inversée: la LED intégrée à la carte s'allume quand le GPIO est à l'état bas
      Serial.println("LED est allumée");
    }
    if (message == "off") {
      digitalWrite(LED, HIGH); // Eteindre la LED
      Serial.println("LED est Eteinte");
    }
    Serial.println();
    Serial.println("-----------------------");
}
```



Ici, nous avons utilisé la LED Bleue intégrée à la carte (voir schéma en haut). Vous pouvez utiliser une LED externe en série avec une résistance de 220Ω. Dans ce cas, inverser la logique de commande du GPIO

4 Programme complet ……..1/5:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const int LED = 2; // GPIO 2 correspondant à D4 sur la carte NodeMCU
bool ledState = false;

// Paramètres du WiFi

const char* ssid = "AUTOMATIC_LAB_2G"; // Remplacer par le nom du Wifi
const char* password = "ACET-LAB"; // Remplacer par le mot de passe du Wifi

// Paramètres du Broker MQTT
const char *mqtt_broker = "test.mosquitto.org";  // broker mosquitto
const char *mqtt_topic = "ISE/elfadil/led1";     // topic MQTT:
const char *mqtt_username = "";  // Laissez vide si le broker Mosquitto de test n'exige pas d'authentification
const char *mqtt_password = "";  // Laissez vide si le broker Mosquitto de test n'exige pas d'authentification
const int mqtt_port = 1883;  // port MQTT (TCP)
```

Utiliser le Wifi disponible!

Remplacer par votre topic. Chaque Etudiant veillera à utiliser un topic différent des autres!

4 Programme complet….2/5:

```
WiFiClient espClient;
PubSubClient mqtt_client(espClient);
void connectToWiFi();
void connectToMQTTBroker();
void mqttCallback(char *topic, byte *payload, unsigned int length);
void setup() {
    Serial.begin(9600);
    pinMode(LED, OUTPUT); // Définir le GPIO de la LED comme sortie
    connectToWiFi();
    mqtt_client.setServer(mqtt_broker, mqtt_port);
    mqtt_client.setCallback(mqttCallback);
    connectToMQTTBroker();
}
void connectToWiFi() {
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to the WiFi network");
}
```

4 Programme complet….3/5:

```
void connectToMQTTBroker() {
    while (!mqtt_client.connected()) {
        String client_id = "esp8266-client-" + String(WiFi.macAddress());
        Serial.printf("Connecting to MQTT Broker as %s.....\n", client_id.c_str());
        if (mqtt_client.connect(client_id.c_str(), mqtt_username, mqtt_password)) {
            Serial.println("Connected to MQTT broker");
            mqtt_client.subscribe(mqtt_topic);
            // Publish message upon successful connection
            mqtt_client.publish(mqtt_topic, "Hi, this is the Embedded Systems Master from ENSA");
        } else {
            Serial.print("Failed to connect to MQTT broker, rc=");
            Serial.print(mqtt_client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
```

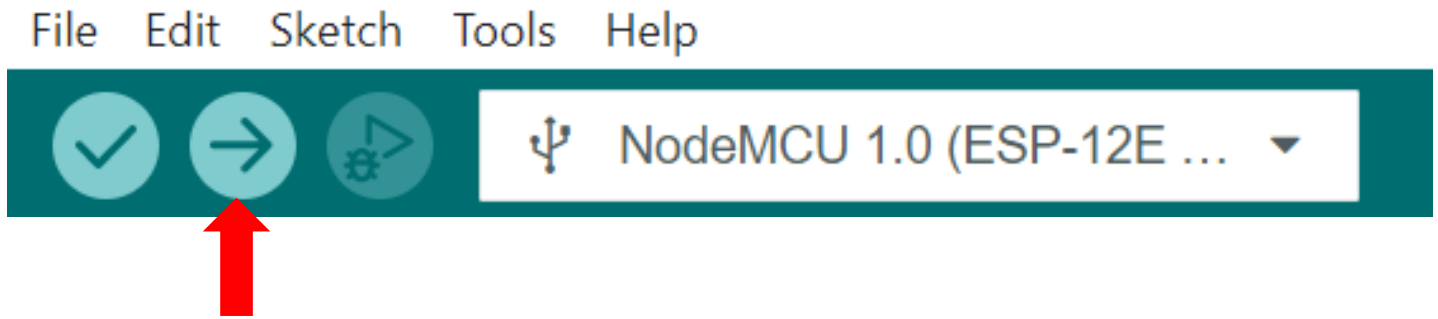
4 Programme complet….4/5:

```
void mqttCallback(char *topic, byte *payload, unsigned int length) {
    Serial.print("Message received on topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    String message;
    for (int i = 0; i < length; i++) {
        message += (char) payload[i];  // Convert *byte to string
    }
    Serial.println(message); // afficher le message MQTT reçu
        if (message == "on") {
      digitalWrite(LED, LOW); // logique inversée: la LED intégrée à la carte s'allume quand le GPIO est à l'état bas
      Serial.println("LED est allumée");
    }
    if (message == "off") {
      digitalWrite(LED, HIGH); // Eteindre la LED
      Serial.println("LED est Eteinte");
    }
    Serial.println();
    Serial.println("-----------------------");
}
```

4 Programme complet….5/5:

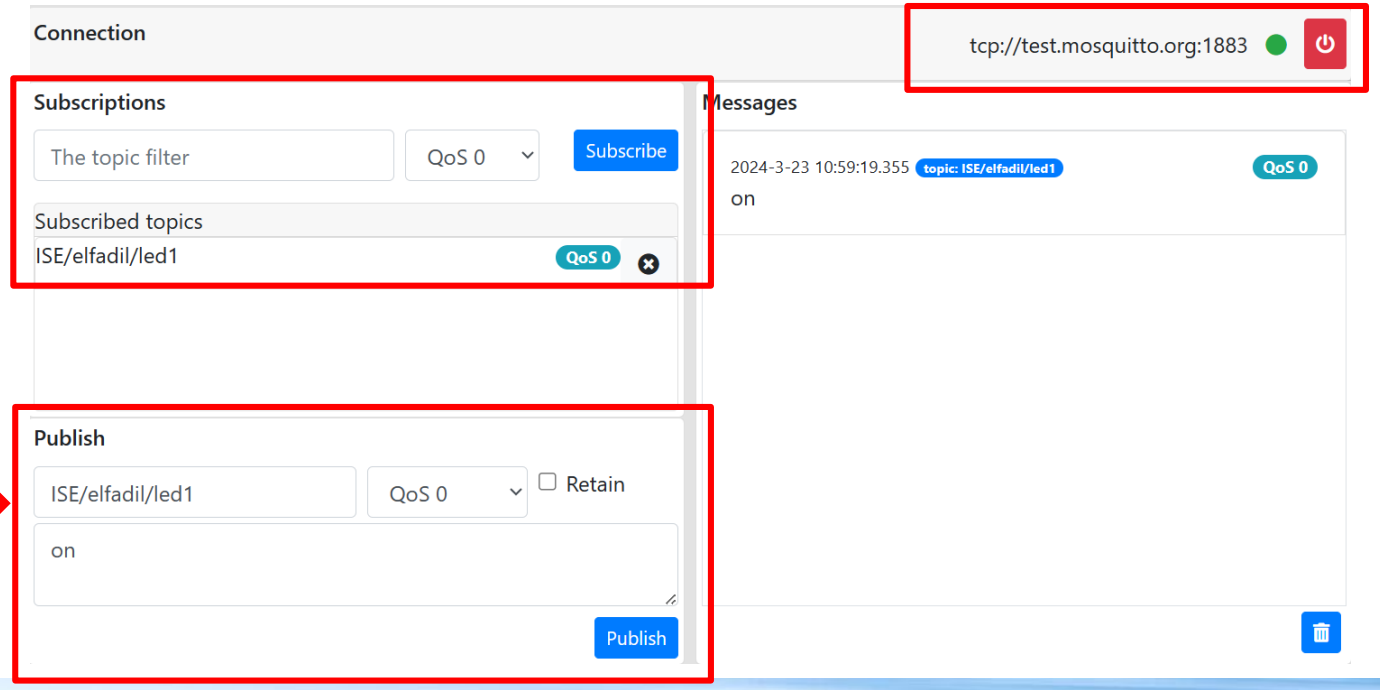
```
void loop() {
    if (!mqtt_client.connected()) {
        connectToMQTTBroker();
    }
    mqtt_client.loop();
}
```

5 La carte étant connectée au port USB du PC, Téléverser, alors, le programme (UPLOAD):



6 Tester l'application IoT réalisée:

- Utilisez le client MQTT pour vous connecter au serveur MQTT. **Exemple**: **« Test Client »: https://testclient-cloud.mqtt.cool/**
- Abonnez-vous à votre topic (Subscription): Exemple dans ce cas: **ISE/elfadil/led1**
- Publiez un message « on » pour allumer la LED et un message « off » pour l'éteindre.



Vérifier que lorsque vous publiez un message « on » la LED bleue intégrée à la carte doit s'allumer et lorsque vous publiez un message « off » la LED doit s'éteindre! ENJOY!!!

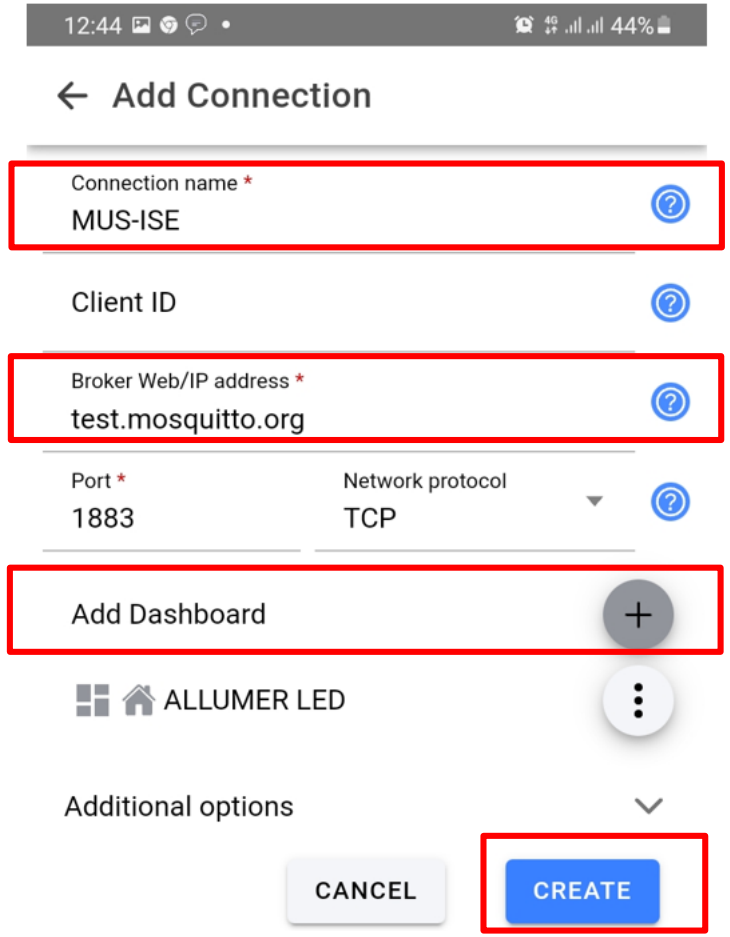
Si vous utilisez une LED externe, celle-ci s'allumera lorsque vous publiez un message « off »

4.4. Monitoring à l'aide de l'application Android « IoT MQTT Panel »:

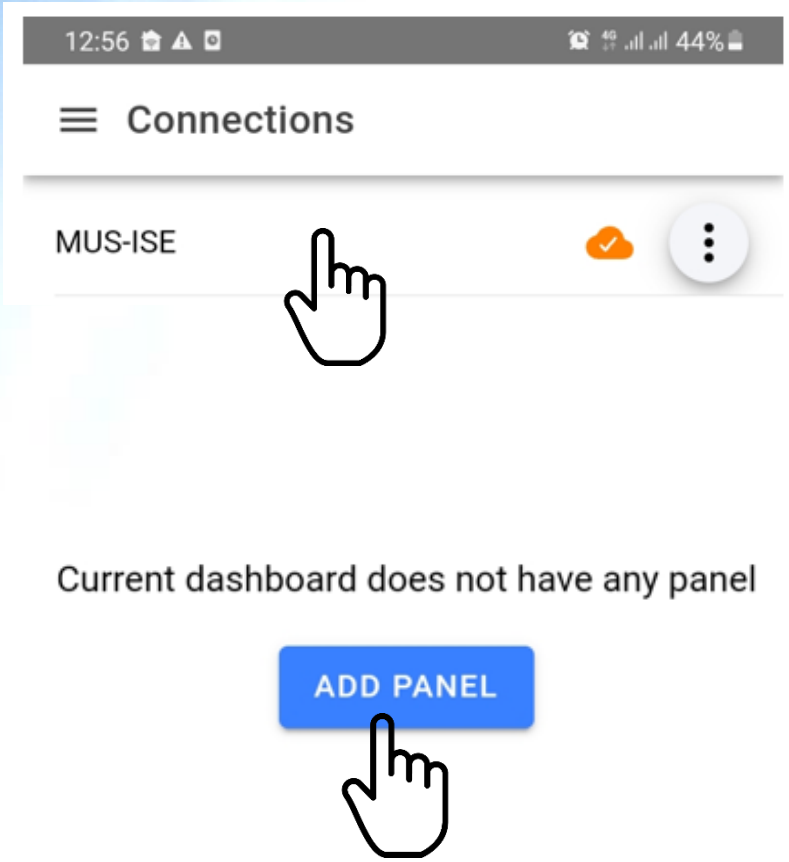
a) **Installer l'application « *IoT MQTT Panel* »: sur votre Smart phone**

b) **Ajouter une connexion « Add Connection »:**

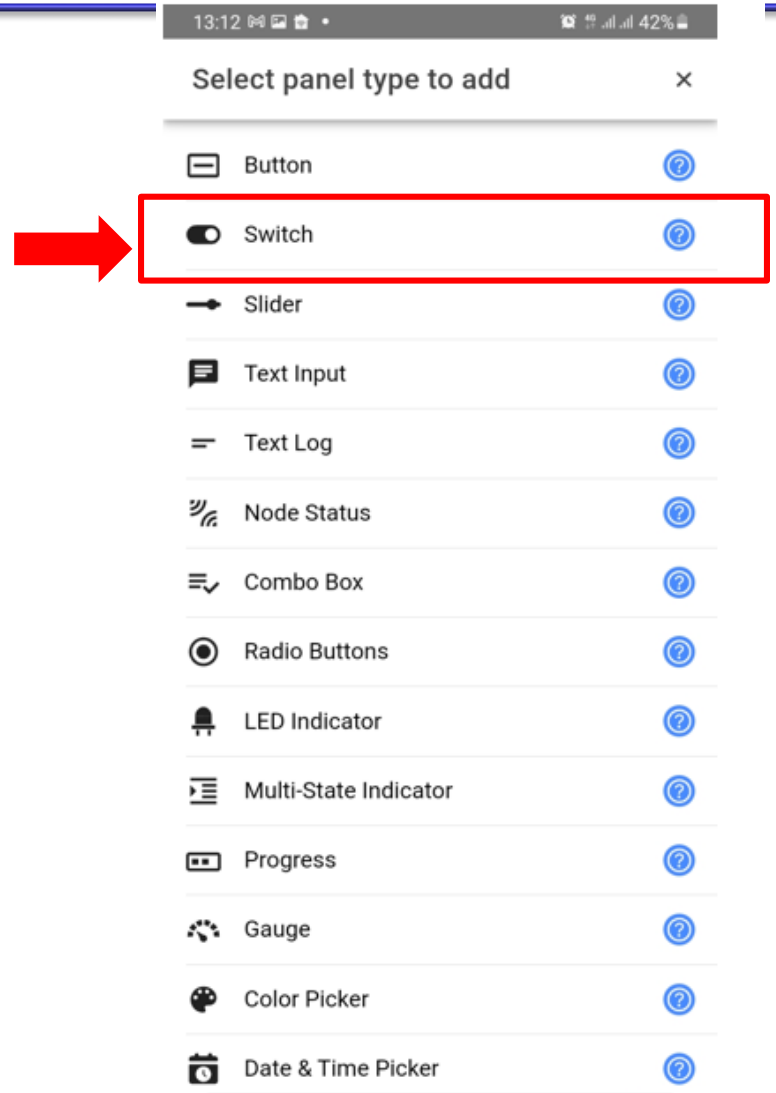
- Saisir un nom. Exemple: MUS-ISE
- Définir le broker: test.mosquitto.org
- Ajouter un dashboard: Exemple: ALUMER LED
- Appuyer sur: CREATE



c) Appuyer sur MUS-ISE et ajouter un panel (ADD PANEL)



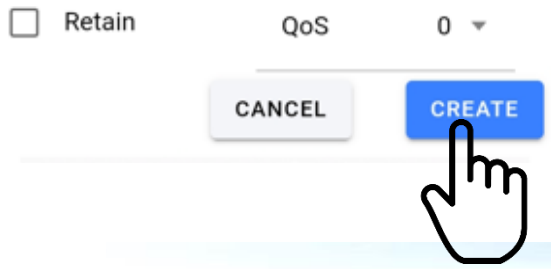
d) Sélectionner « Switch »

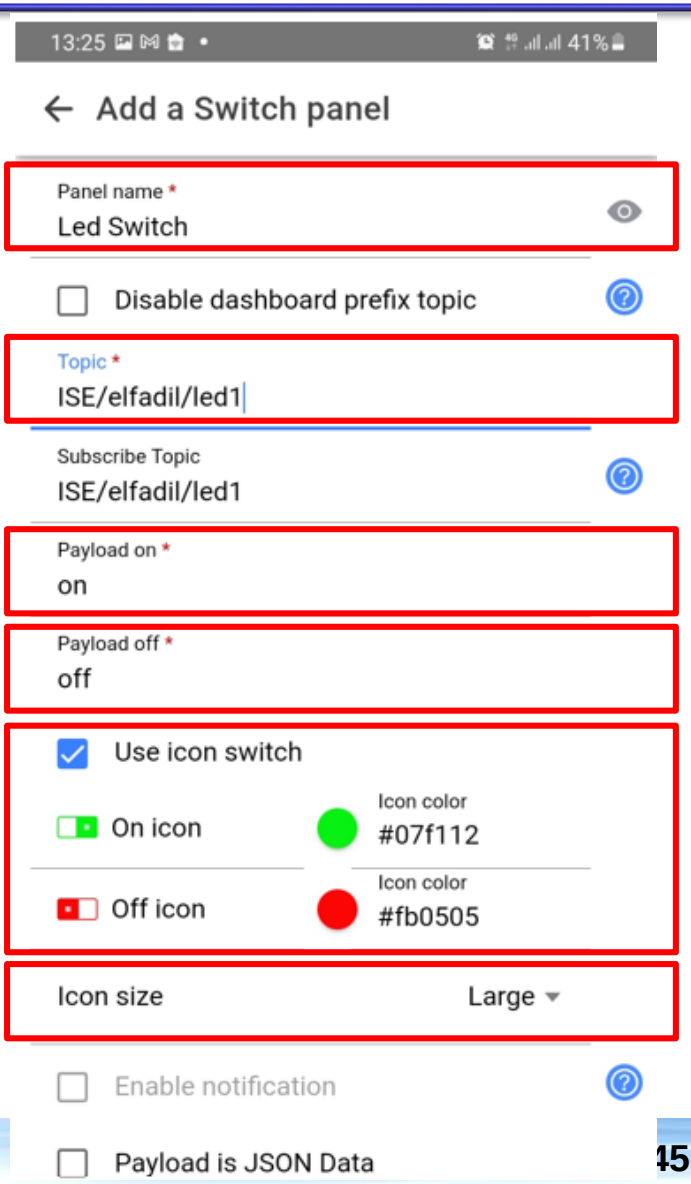


e) Configurer les paramètres du Switch:

- Nom. Exemple: **Led Switch**
- Topic. Exemple: **ISE/elfadil/led1**
- Payload on: **on**
- Payload off: **off**
- Vous pouvez utiliser une icône pour le switch. Exemple: le vert quand c'est ON et le rouge quand c'est OFF.
- Choisir la taille de l'icône. Exemple: **Large**.

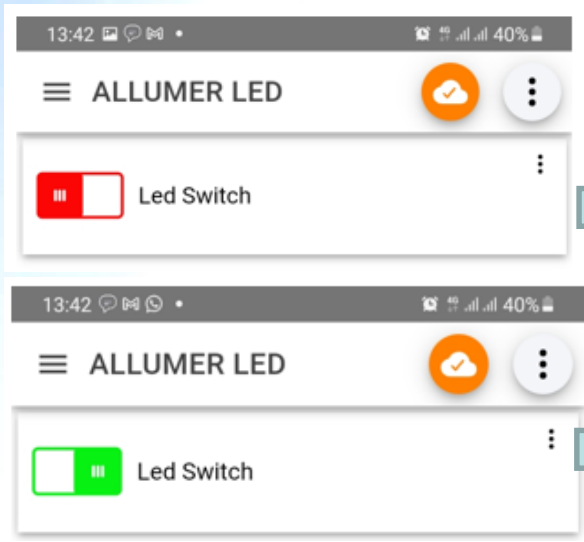
f) N'oublier pas d'appuyer sur CREATE:



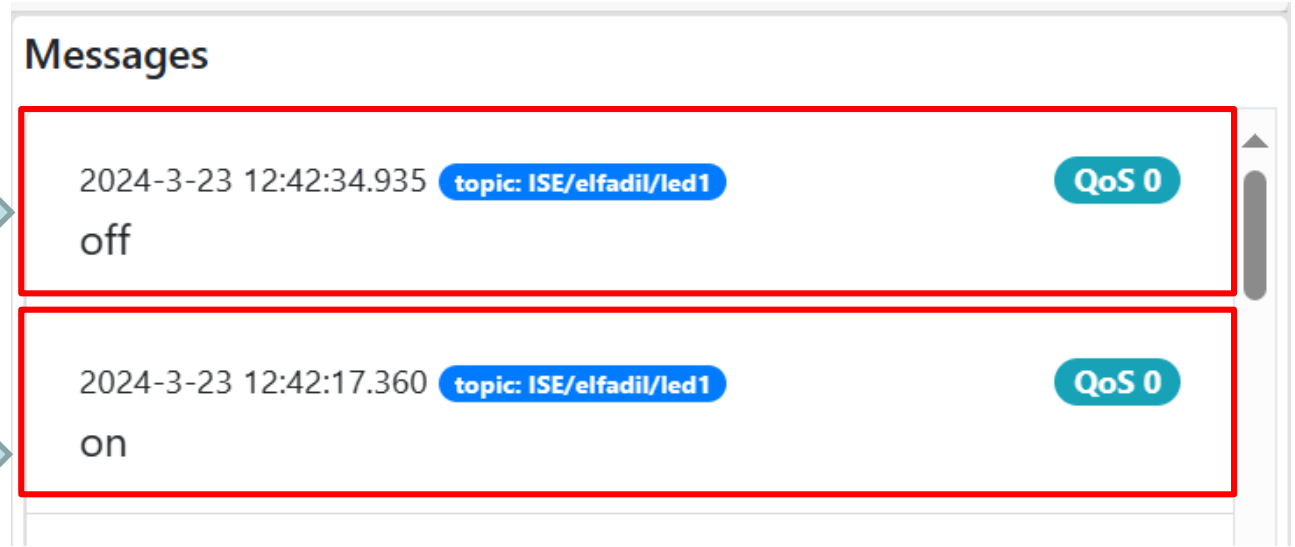


f) Vérifier le fonctionnement: Test et validation

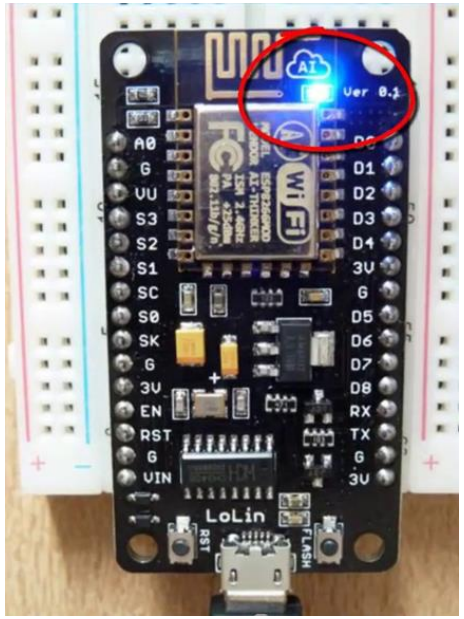
Application Android: IoT MQTT Panel



Application: Test Client



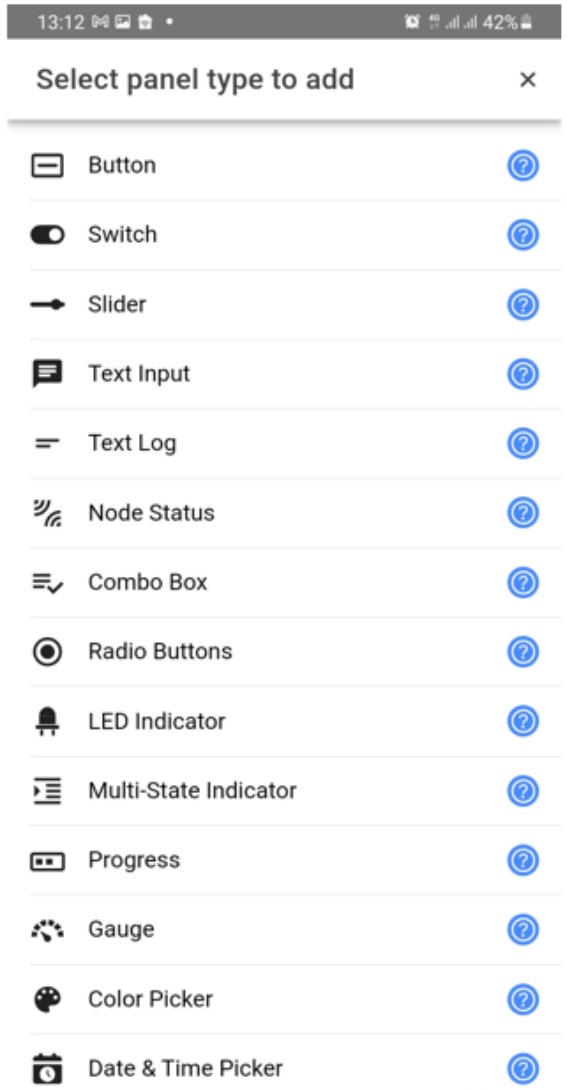
LED Bleue de la carte NodeMCU

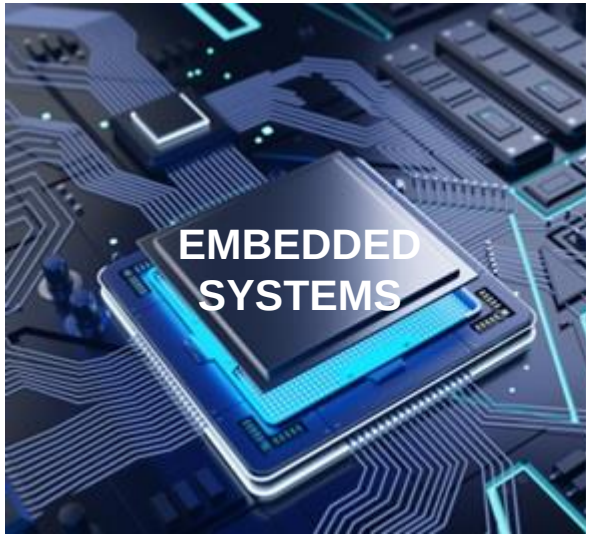


ENJOY !!!

g) Amusez-vous à utiliser les autres panels de l'application :

- **Button (bouton)** : Permet d'envoyer des commandes ou des messages à un appareil ou un système IoT lorsqu'il est activé.
- **Slider (curseur)** : Utilisé pour ajuster des valeurs numériques telles que la luminosité, la température, etc., en les faisant glisser entre un min et un max.
- **Text Input (entrée de texte)** : Permet à l'utilisateur d'entrer des données textuelles, comme des instructions ou des paramètres.
- **Combobox (liste déroulante)** : Offre une sélection de choix à l'utilisateur, souvent utilisée pour des options prédéfinies.
- **LED Indicator (indicateur LED)** : Indique l'état d'un dispositif ou d'un système à l'aide d'une LED virtuelle, souvent pour signaler un état actif ou inactif.
- **Gauge (jauge)** : Affiche visuellement une valeur numérique sur une échelle, souvent utilisée pour des mesures telles que la pression, la vitesse, etc.
- **Color Picker (sélecteur de couleur)** : Permet à l'utilisateur de choisir une couleur, utile pour des applications de contrôle visuel (mode HEX ou RGB).
- **Date and Time Picker (sélecteur de date et d'heure)** : Facilite la sélection de dates et d'heures spécifiques pour des actions programmées ou des paramètres temporels.
- **…**,





FIN !
Questions ?