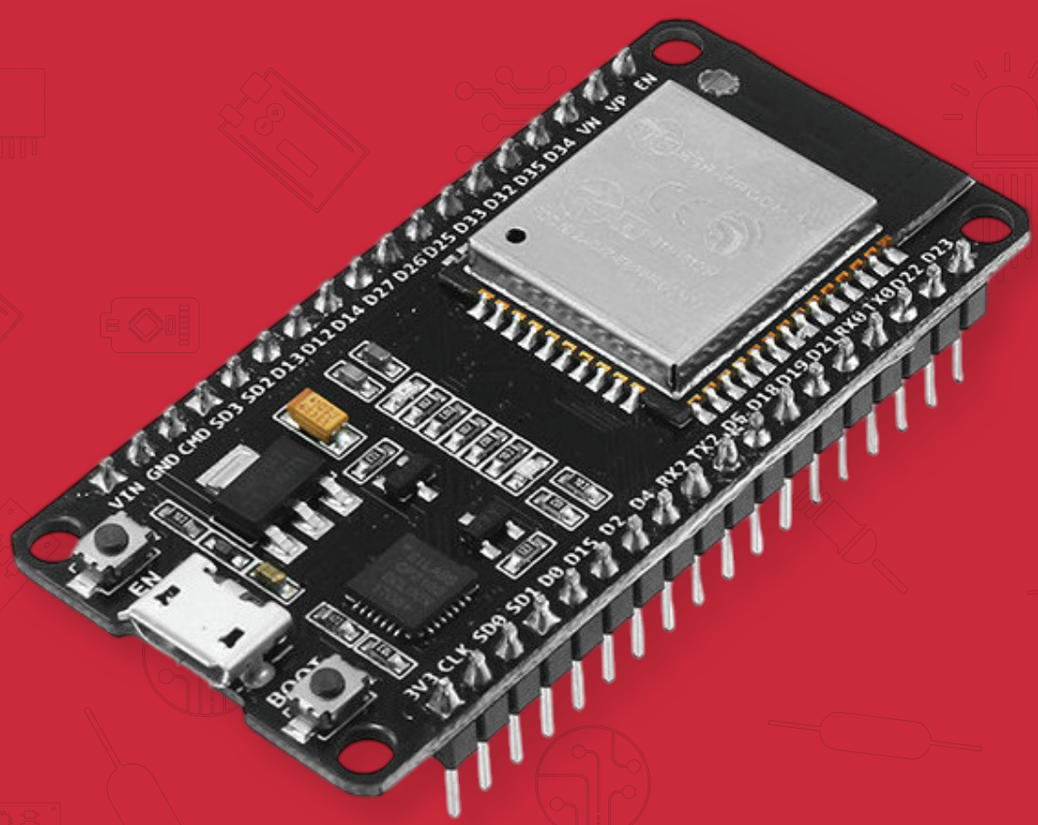
YoupiLab

DEMYSTIFYING ELECTRONICS

SERVEUR WEB ESP32



GUIDE DE PROJET ETAPE PAR ETAPE

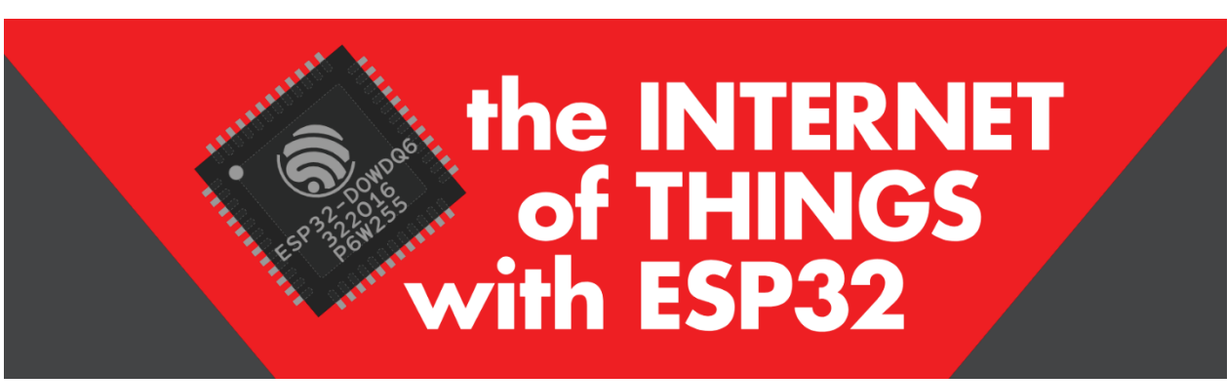
Bonjour et merci d'avoir téléchargé ce projet ebook!

Ce livre électronique rapide vous aidera à démarrer et à créer un serveur Web avec l'ESP32 à l'aide d'Arduino IDE.

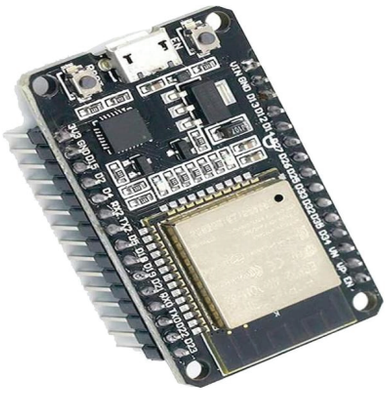
Présentation de la carte ESP32

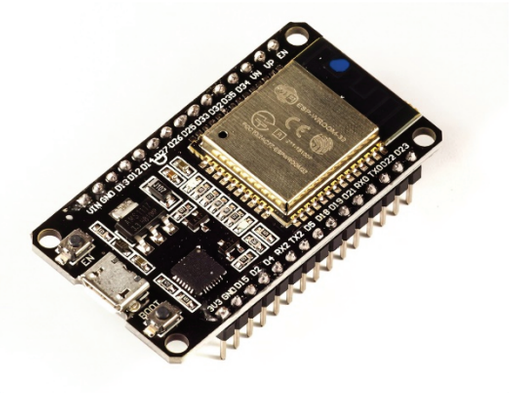
L'ESP32 est le successeur de l'ESP8266. Il est doté de nombreuses nouvelles fonctionnalités. Il combine désormais les capacités sans fil Wi-Fi et Bluetooth.

Il existe de nombreuses cartes de développement ESP32. Je vous encourage à visiter le site The Internet of Things with ESP32 où chaque puce ESP32 et carte de développement sont répertoriées. Vous pouvez comparer leurs différences et caractéristiques.

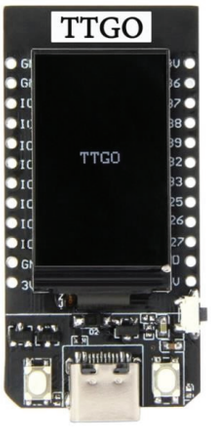


Dans cet ebook, nous utiliserons la carte ESP32 DEVKIT, mais n'importe quel autre ESP32 avec la puce ESP-WROOM-32 fonctionnera très bien.

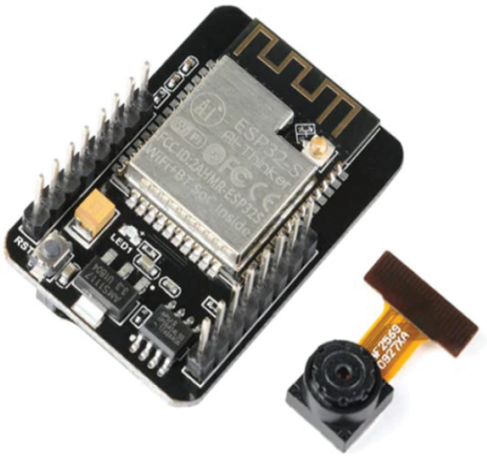




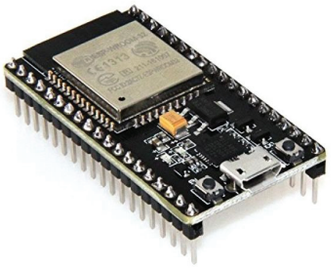
Voici quelques exemples de cartes très similaires et compatibles avec le projet de cet ebook.



Module TTGO Esp32



ESP32 CAM

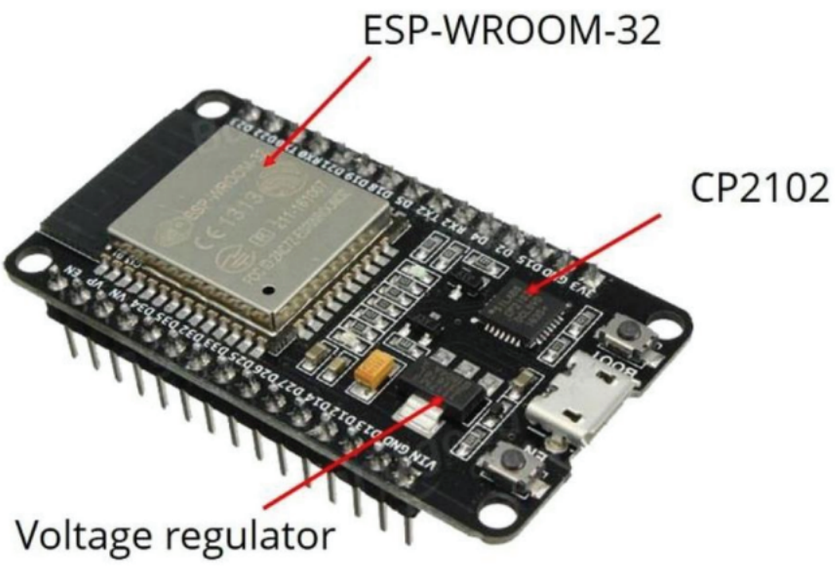


Module wifi NODE MCU ESP 32

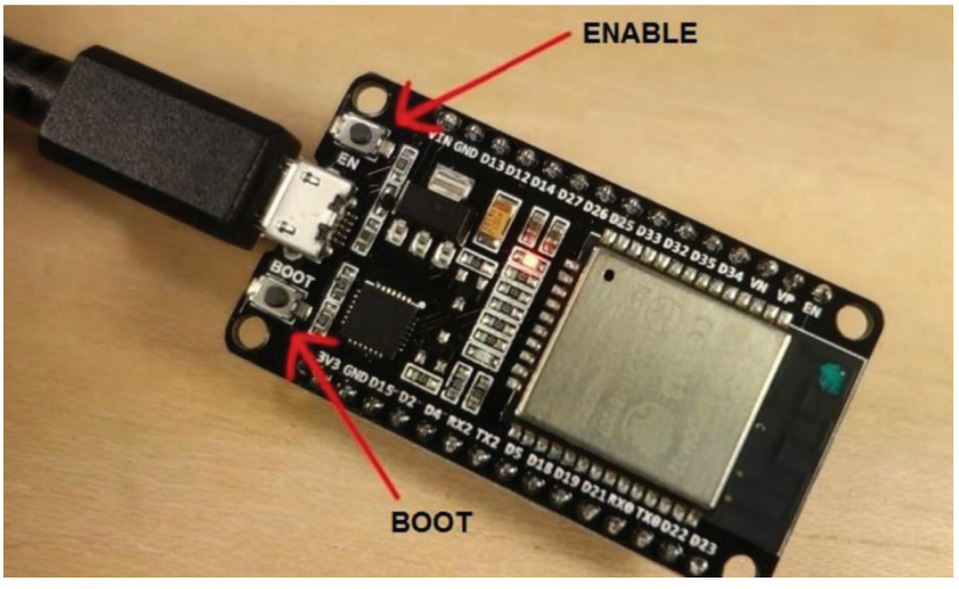
Vous pouvez trouver ces différentes cartes disponibles sur notre plateforme à travers ce lien : https://youpilab.com/components/searchbar/ESP32

Caractéristiques

L'ESP32 est livré avec la puce ESP-WROOM-32. Il dispose d'un régulateur de tension de 3,3 V qui fait chuter la tension d'entrée pour alimenter la puce ESP32. Il est également livré avec une puce CP2102 qui vous permet de brancher l'ESP32 sur votre ordinateur pourle programmer sans avoir besoin d'un FTDI programmeur.



La carte dispose de deux boutons intégrés : le bouton ENABLE et le bouton BOOT.



Si vous appuyez sur le bouton ENABLE, l'ESP32 se redémarre. Si vous maintenez le bouton BOOT enfoncé puis appuyez sur le bouton enable, l'ESP32 redémarre en mode programmation.

Caractéristiques

En ce qui concerne les spécifications de la puce ESP32, vous constaterez que :

• L'ESP32 est double cœur, ce qui signifie qu'il dispose de 2 processeurs.

• Il dispose du Wi-Fi et du Bluetooth intégrés.

• Il exécute des programmes 32 bits.

• La fréquence d'horloge peut aller jusqu'à 240 MHz et il dispose d'une RAM de 512 Ko.

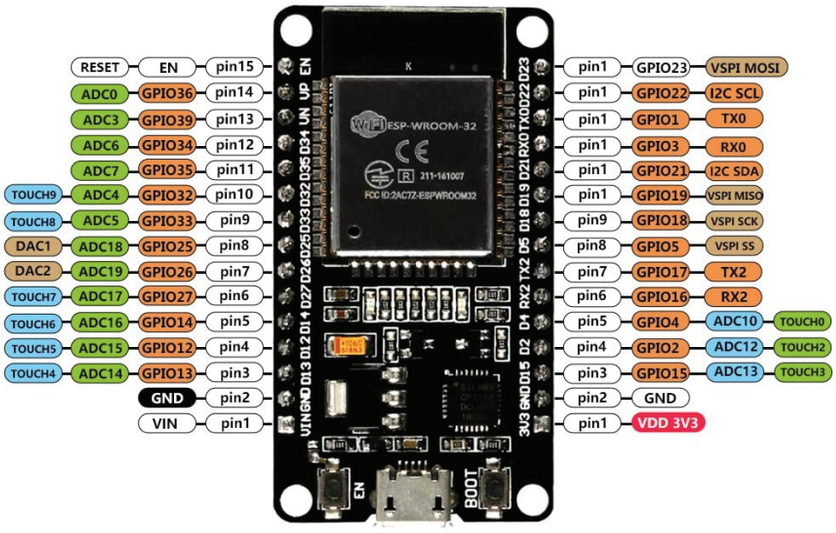
• Il dispose également d'une grande variété de périphériques disponibles, tels que : écran tactilecapacitif, ADC, DAC,UART, SPI, I2C et bien plus encore

Spécifications - ESP32 DEVKIT V1 DOIT	Caractéristiques
Nombre de cœurs	2 (Double cœur)
Wi-Fi	2,4 GHz jusqu'à 150 Mbit/s
Bluetooth	BLE (Bluetooth Low Energy) et Bluetooth classique
Architecture	32 bits
Fréquence d'horloge	Jusqu'à 240 MHz
RAM	512 Ko
Nombre de broches	30
Périphériques	Touche capacitive, ADC (convertisseur analogique-numérique), DAC (convertisseur numérique-analogique), I^2C (Circuit Inter-Intégré), UART (transmetteur/récepteur asynchrone universel), CAN 2.0 (réseau de zone de contrôle), SPI (Interface de périphériques série), I^2S (Son Inter-IC), RMII (Interface Médias Indépendante Réduite), PWM (modulation de largeur d'impulsion), et plus.

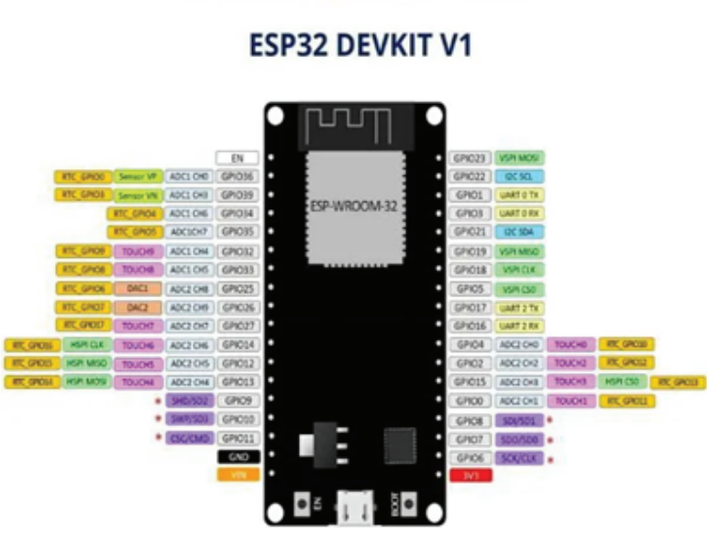
ESP32 Pinout:

Les figures suivantes décrivent clairement les GPIO de la carte et leurs fonctionnalités. Nous vous recommandons d'imprimer ce brochage pour référence ultérieure.

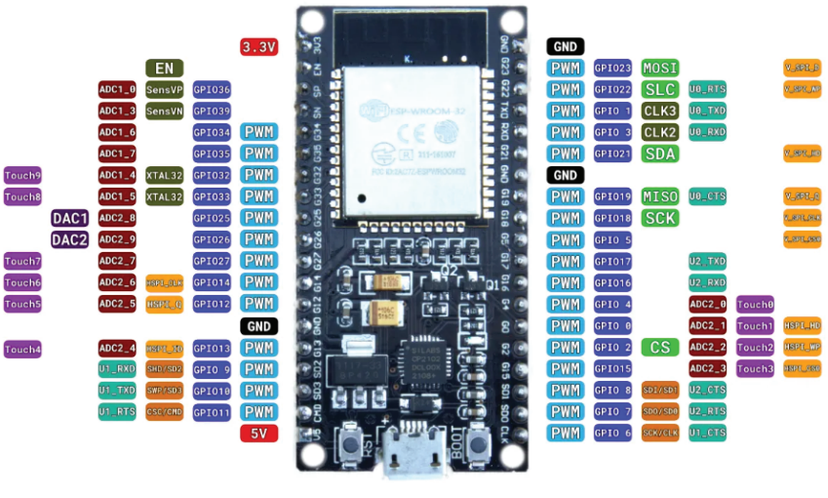
ESP32 DEVKIT V1 Version avec 30 GPIOS



ESP32 DEVKIT V1 Version avec 36 GPIOS



Nœud MCU ESP32 38 broches



ESP32 DEVKIT V1 Version avec 38 GPIOS

Installation de l'ESP32 dans l'IDE Arduino

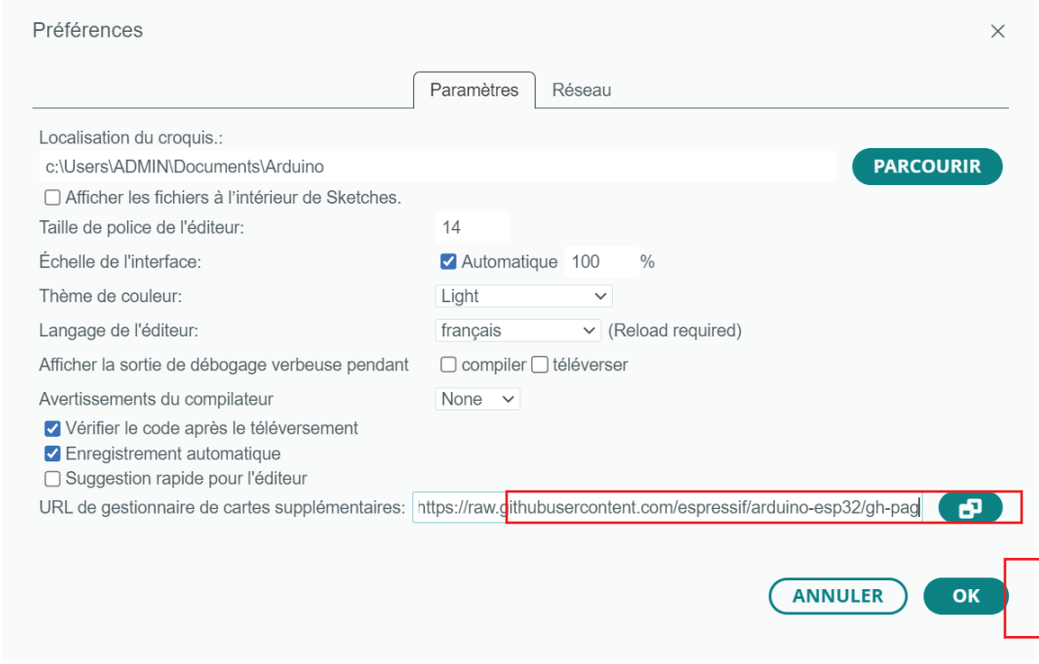
Important : avant de commencer cette procédure d'installation, assurez-vous que la dernière version de l'IDE Arduino est installée sur votre ordinateur. Si ce n'est pas le cas, désinstallez-la et réinstallez-la. Sinon, elle risque de ne pas fonctionner.

L'ESP32 est actuellement intégré à l'IDE Arduino, tout comme cela a été fait pour l'ESP8266. Ce module complémentaire pour l'IDE Arduino vous permet de programmer l'E SP32 à l'aide de l'IDE Arduino et de son langage de programmation.

1. Installation de la carte ESP32

Pour installer la carte ESP32 dans votre IDE Arduino, suivez ces instructions suivantes :

1) Ouvrez la fenêtre des préférences depuis l'IDE Arduino. Allez dans Fichier
2) Entrez https://dl.espressif.com/dl/package_esp32_index.json dans le champ « URL supplémentaires du gestionnaire de cartes » comme indiqué dans la figure ci-dessous. Cliquez ensuite sur le bouton « OK ».

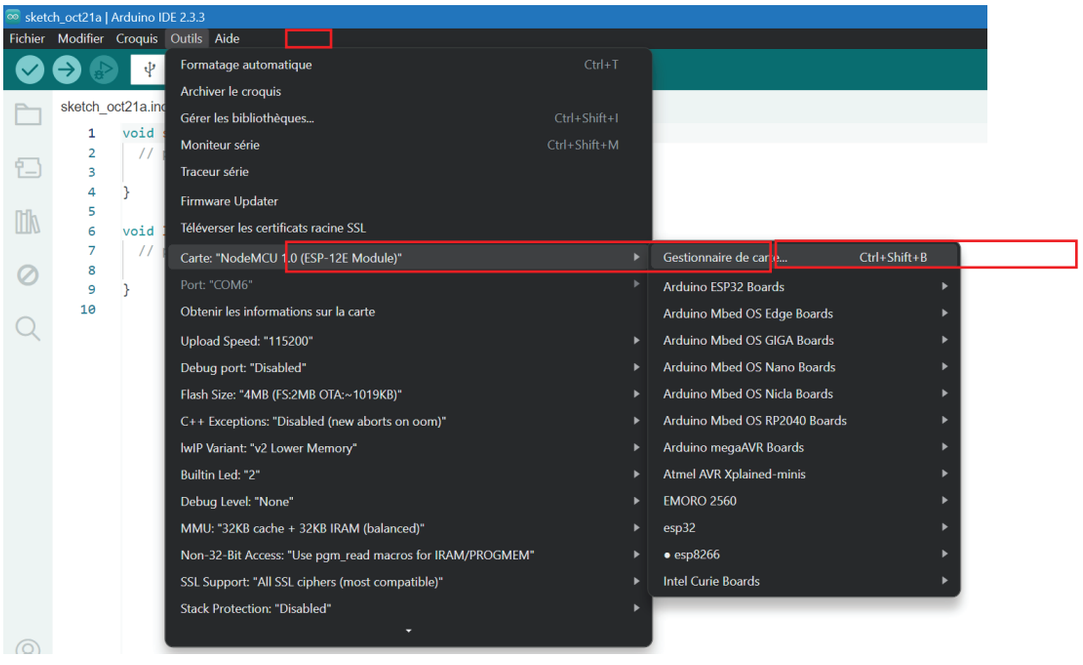


Remarque : :si vous disposez déjà de l'URL des cartes ESP8266, vous pouvez séparer les URL par une virgule comme suit :

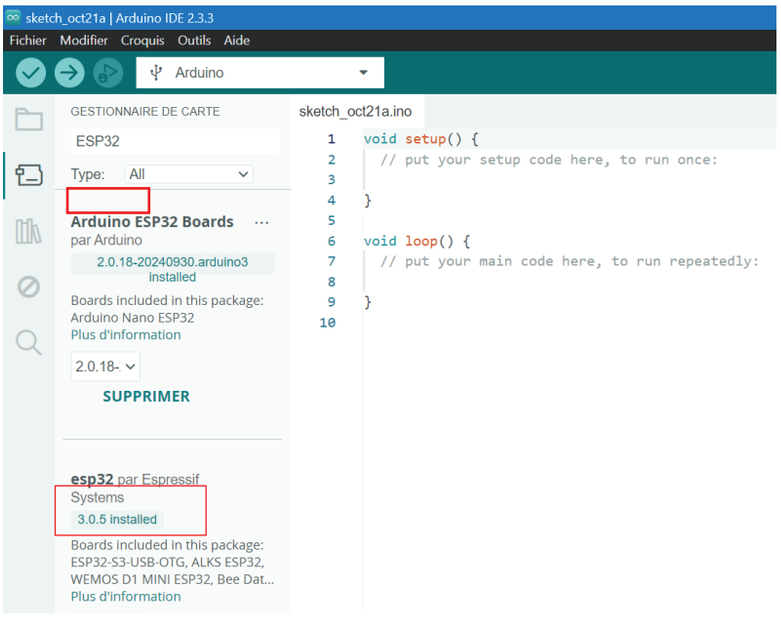
https://dl.espressif.com/dl/package_esp32_index.json,

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

3) Ouvrez votre IDE; allez dans Outils; Carte puis gestionnaire de carte



4) Recherchez ESP32 et appuyez sur le bouton d'installation de « ESP32 by Espressif Systems » :

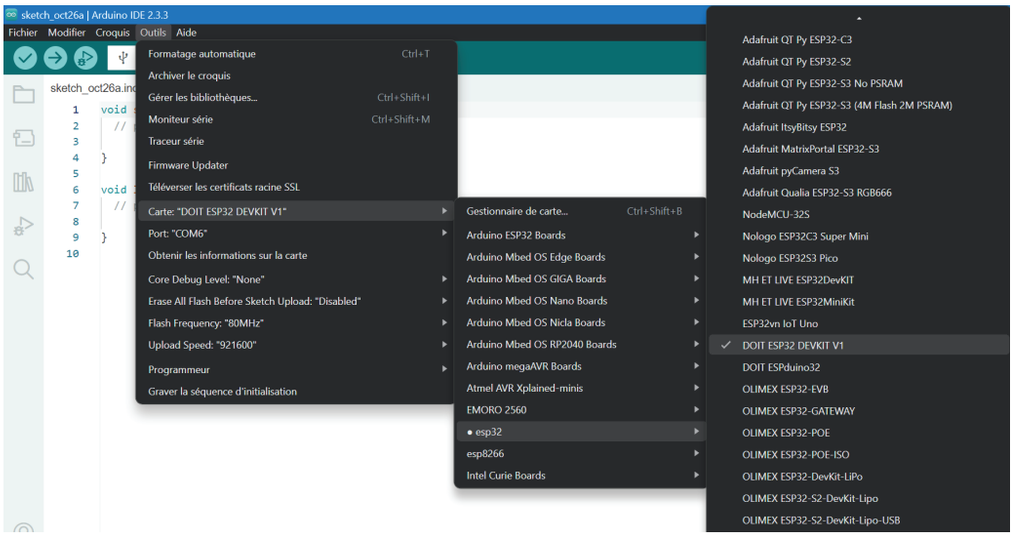


Test de l'installation

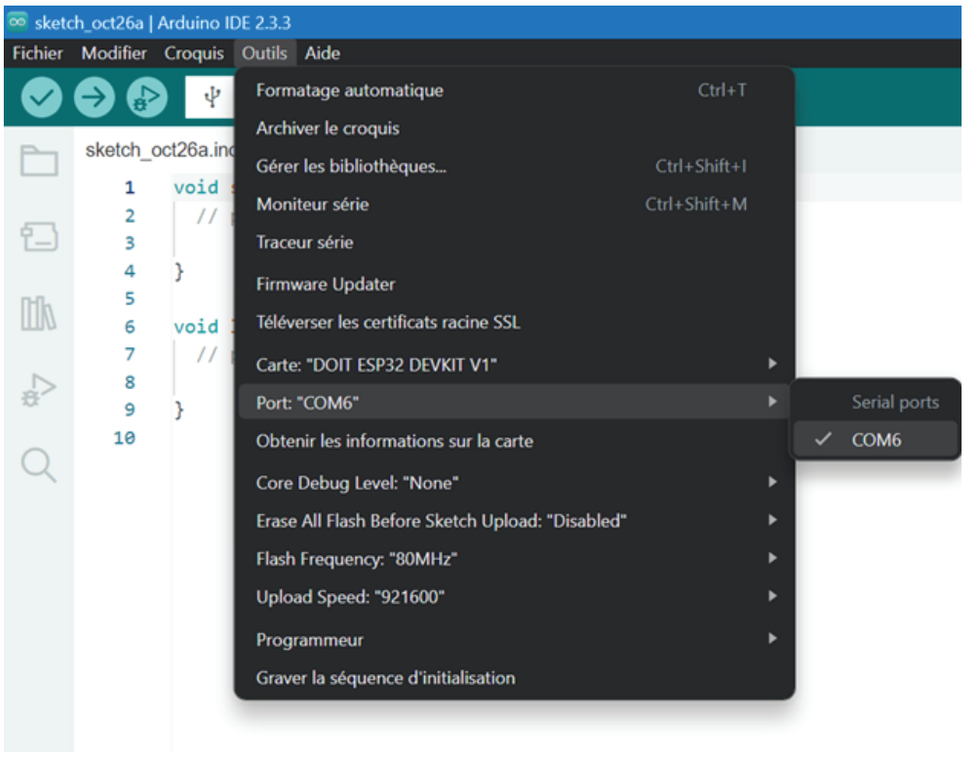
Branchez votre carte ESP32 DOIT DEVKIT V1 sur votre ordinateur. Ensuite, suivez ces étapes

1) Ouvrez l'IDE Arduino

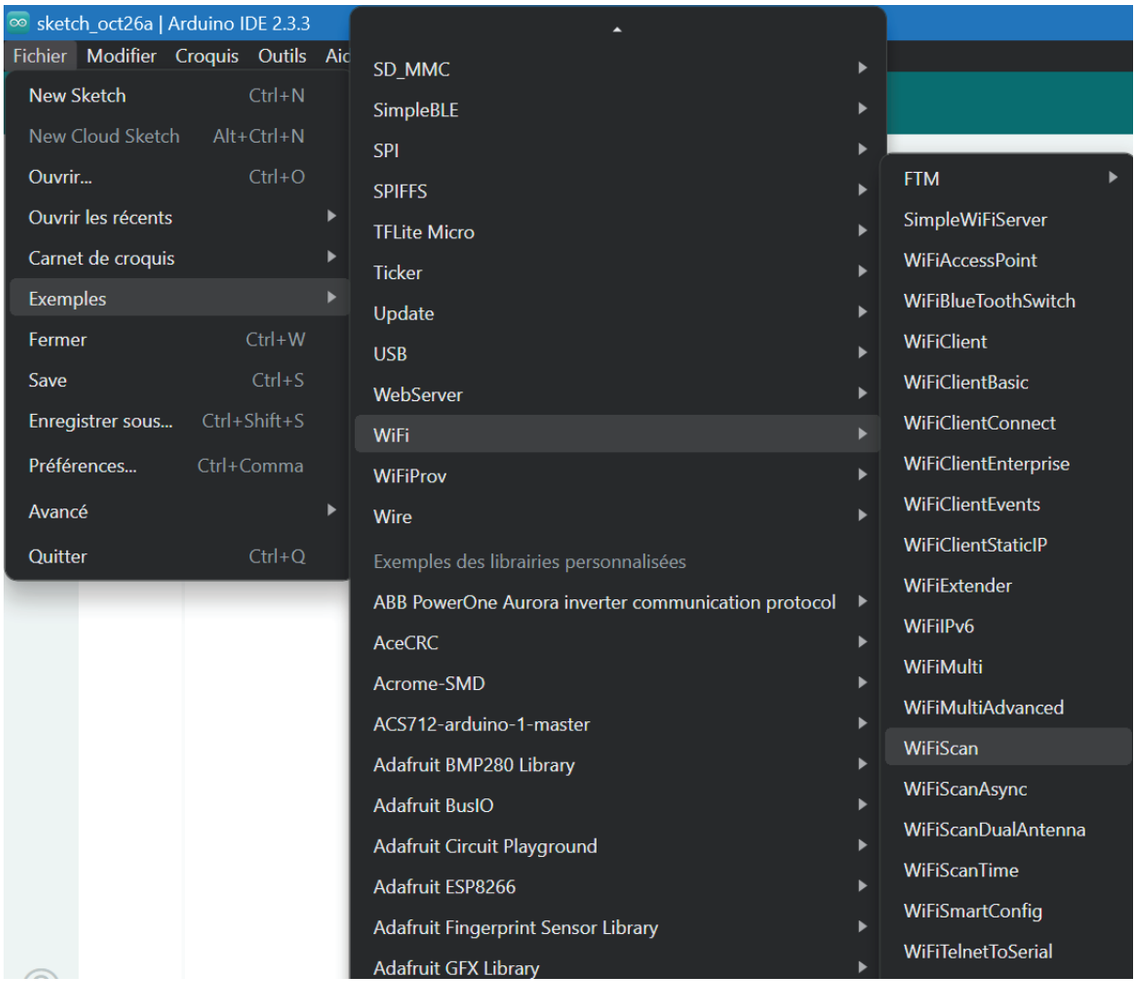
2) Sélectionnez votre carte dans le menu Outils; Carte (dans notre cas i s'agit du DOIT ESP32 DEVKIT V1)



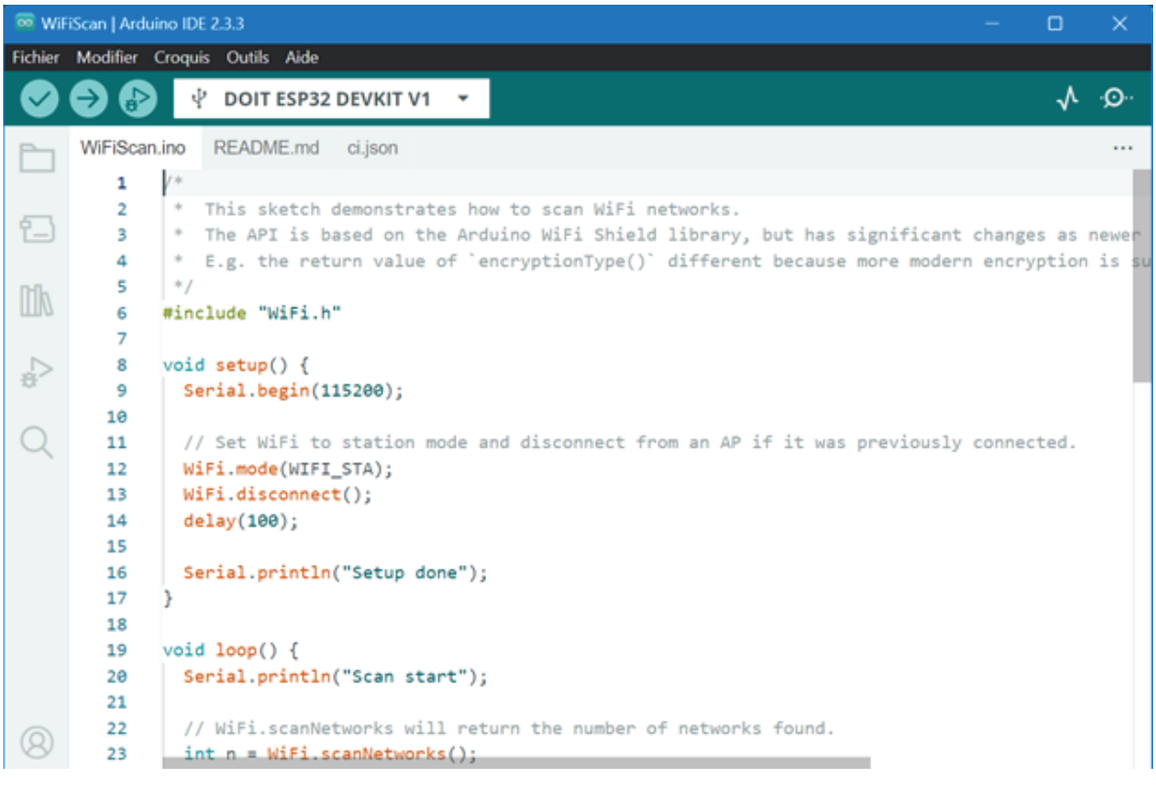
3.) Sélectionnez le po rt (si vous ne voyez pas le port COM dans votre IDE Arduino vous devez installer les pilotes à ce lien [CP210x USB to UART Bridge VCP Drivers - Silicon Labs])



4) Ouvrez l'exemple suivant sous fichier Exemples Wifi (ESP32) Analyse Wifi

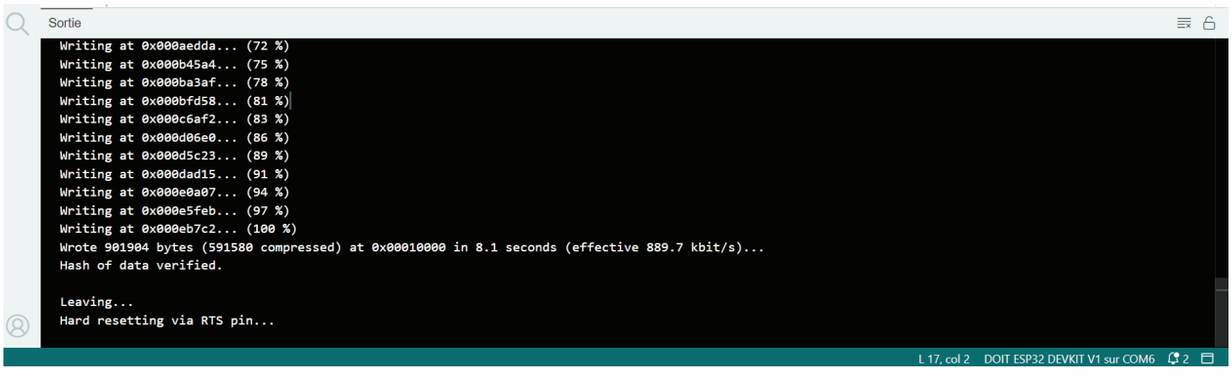


5) Un nouveau croquis s'ouvre



6) Appuyer sur le bouton téléverse dans l'IDE Arduino. Attendez quelques secondes pendant que le code se compile et se téléverse sur votre carte

7)Si tout s'est déroulé comme prévu vous devriez voir un message « Téléversement fait »



8)Ouvrez le moniteur série Arduino IDE à un débit de 115 200.

9)Appuyer sur le bouton intégré de l'ESP32 et vous devriez voir les réseaux disponibles à proximité de votre ESP32

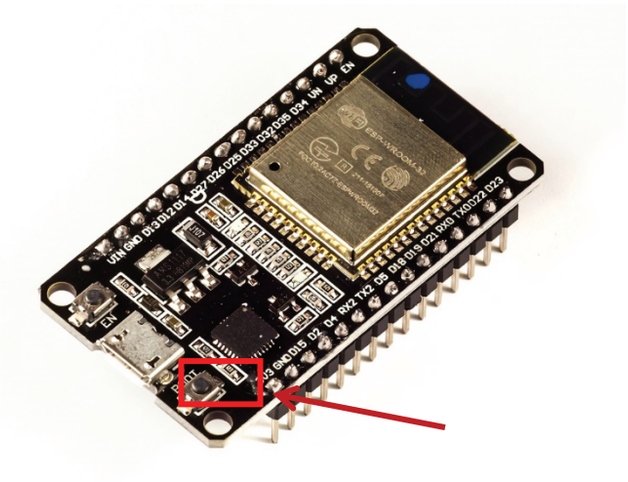
Il s'agit d'un tutoriel très basique qui illustre comment préparer votre IDE Arduino pour l'ESP32 sur votre ordinateur

Conseil de dépannage n°1 :

« Échec de la connexion à l'ESP32 : délai expiré… Connexion… »

Lorsque vous essayez de télécharger un nouveau croquis sur votre ESP32 et qu'il ne parvient pas à se connecter à votre carte, cela signifie que vot ESP32 n'est pas en mode flashage/téléchargement. Après avoir sélectionné le bon nom de carte et le port COM, suivez ces étapes :

• Maintenez enfoncé le bouton « BOOT » de votre carte ESP32.



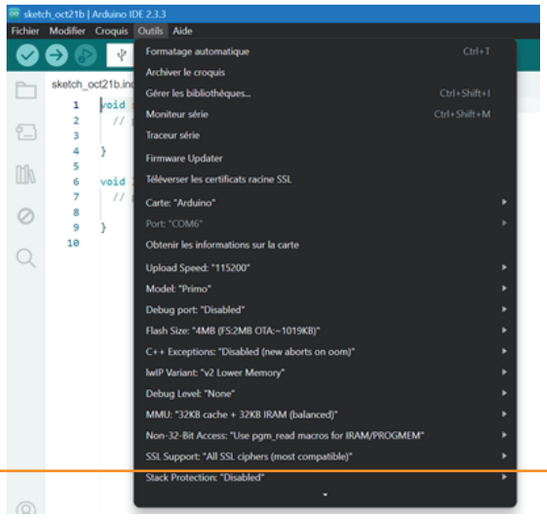
• Appuyez sur le bouton « Télécharger » dans l'IDE Arduino pour télécharger un nouveau croquis :

Après avoir vu le message « Connexion… » dans votre IDE Arduino, relâchez le doigt du bouton « BOOT ».

Conseil de dépannage n° 2 :

Port COM non trouvé/non disponible

Si vous branchez votre carte ESP32 à votre ordinateur, mais que vous ne trouvez pas le port ESP32 disponible dans votre IDE Arduino (il est grisé) :

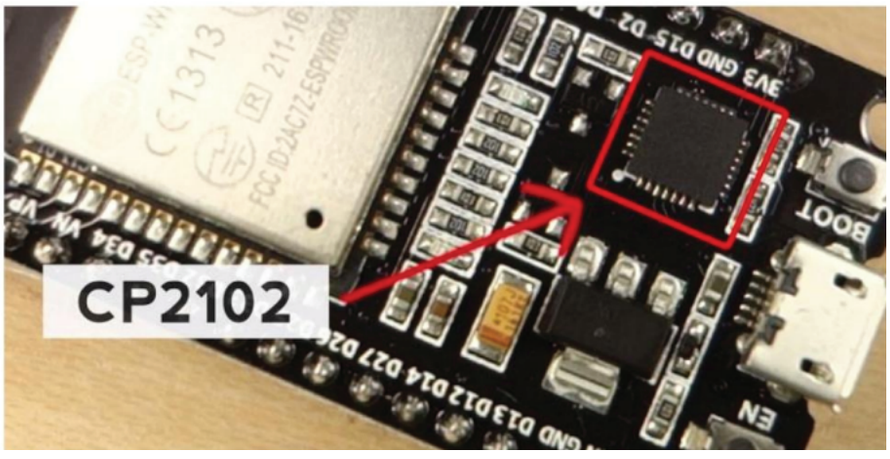


Il peut s'agir de l'un de ces deux problèmes :

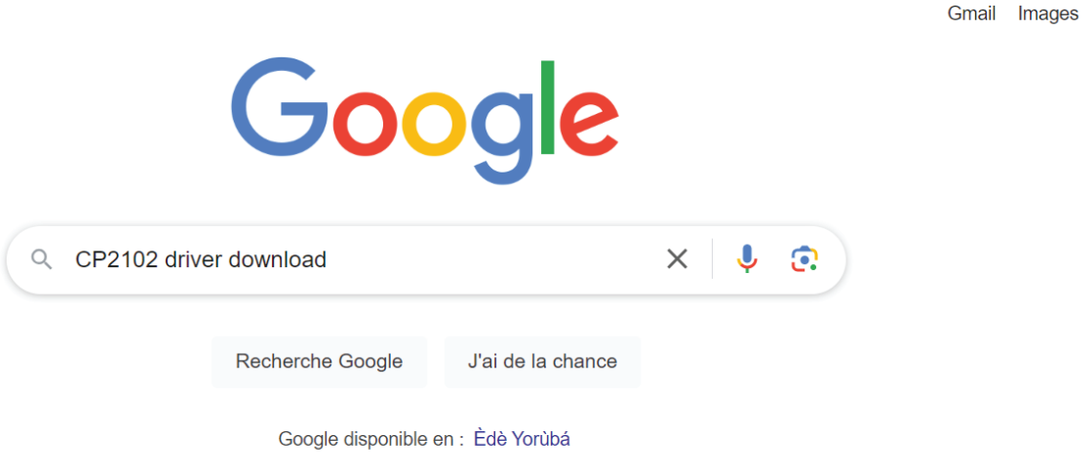
Pilotes USB manquants ou câble USB sans fils de données.

1. Si vous ne voyez pas le port COM de votre ESP disponible, cela signifie souvent que vous n'avez pas installé les pilotes USB. Examinez de plus près la puce à côté du régulateur de tension intégré et vérifiez son nom.

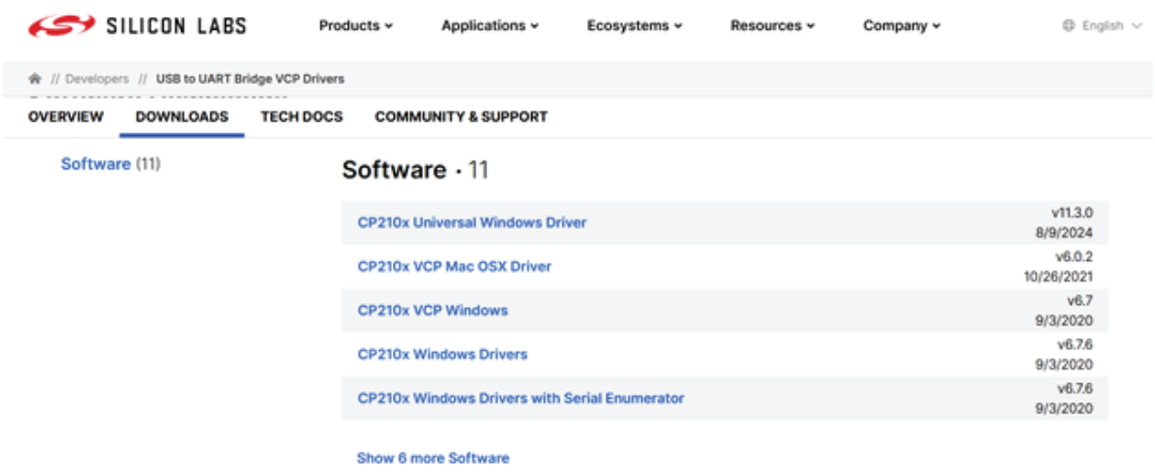
La carte ESP32 DEVKIT V1 utilise la puce CP2102.



Accédez à Google et recherchez votre puce particulière pour trouver les pilotes et les installer pour votre système d'exploitation.



Vous pouvez télécharger les pilotes CP2102 sur [CP210x USB to UART Bridge VCP Drivers - Silicon Labs (silabs.com)]



Une fois installés, redémarrez l'IDE Arduino et vous devriez voir le port COM dans le menu

1. Si vous avez installé les pilotes, mais que vous ne voyez pas votre appareil, vérifiez que vous utilisez un câble USB avec des câbles de données. Les câbles USB des power Banks n'ont souvent pas de câbles de données (ils servent uniquement à la char ge). Ainsi, votre ordinateur n'établira jamais de communication série avec votre ESP32. L'utilisation d'un câble USB approprié devrait résoudre votre problème.

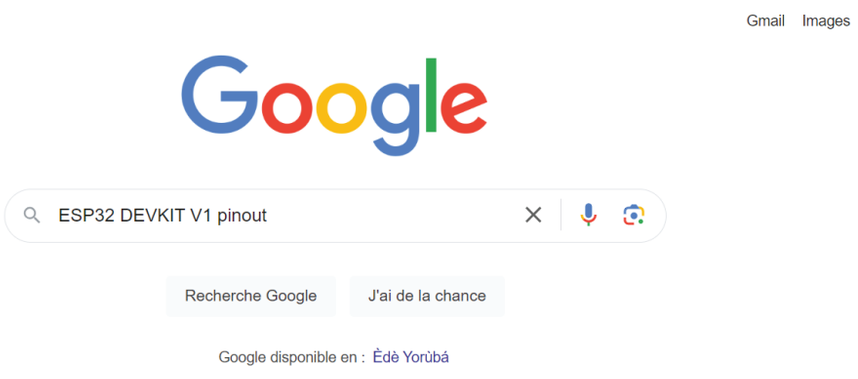
Nom de la carte ESP32

Après avoir lu la page produit de votre ESP32, vous devriez connaître le nom de votre carte. Mais si vous avez encore des doutes, vous pouvez jeter un œil au dos de la carte. Le nom est généralement imprimé avec une sérigraphie. Dans notre cas, vous pouvez clairement voir que c'est la carte ESP32 DEVKIT V1

Si vous possédez un NodeMCU ESP32, la figure suivante montre à quoi il ressemble (il est indiqué NodeMCU ESP-32S).

Brochage ESP32

Connaître le nom de votre carte est très important pour pouvoir recherche son brochage. Vous pouvez maintenant accéder à Google et rechercher le brochage de votre carte de développement. Recherchez le nom de votre carte ESP32 et ajoutez le mot-clé « pinout » à la fin.



Ensuite, recherchez une image qui a le même brochage que votre ESP32.

Je vous recommande également de vister le site The Internet of Things with ESP32 site Web car il fournit une liste complète avec des noms et des chiffr pour toutes les cartes de développement ESP32 connues.

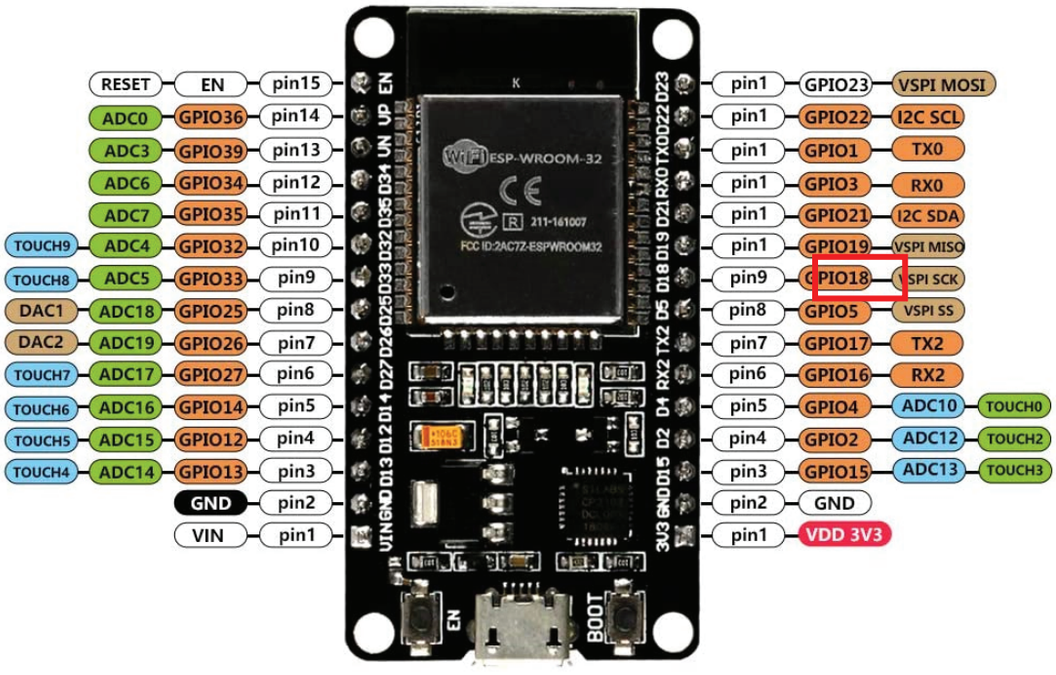
sketch_oct22a.ino

```
1   // Constante définissant la broche à laquelle la LED est connectée
2   const int ledPin = 18;
3
4   void setup() {
5     // Configure la broche ledPin comme une sortie numérique
6     pinMode(ledPin, OUTPUT);
7   }
8
9   void loop() {
10    // Allume la LED
11    digitalWrite(ledPin, HIGH);
12    // Attend 1 seconde
13    delay(1000);
14    // Éteint la LED
15    digitalWrite(ledPin, LOW);
16    // Attend 1 seconde
17    delay(1000);
18  }
```

Comme vous pouvez le voir, vous devez connecter une LED à la broche 8 qui fait référence au GPIO 18.

```
const int ledPin = 18;
```

Si vous utilisez la carte ESP32 DEVKIT V1 DOIT, vous devez connecter votre LED à la broche se trouvant en position 7 sur le côté supérieur droit.



Important : vérifiez toujours le brochage de votre carte spécifique, avant de construire un circuit

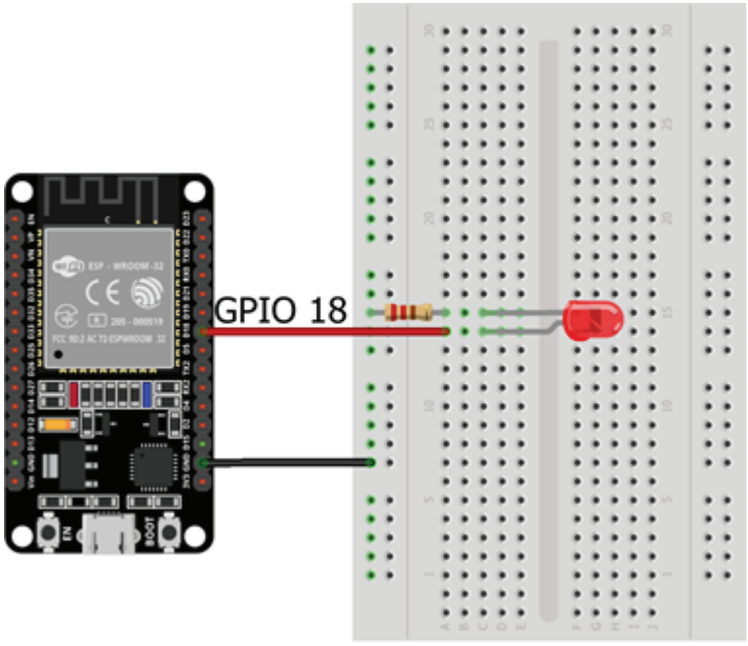
Schématique

Voici une liste des pièces dont vous avez besoin pour assembler le circuit :

- ESP32 DOIT DEVKIT V1
- LED 5mm
- Resistance 220 ohms
- Jumper
- Breadboard

Retrouvez tous ces composants disponibles sur notre plateforme: https://youpilab.com/components/

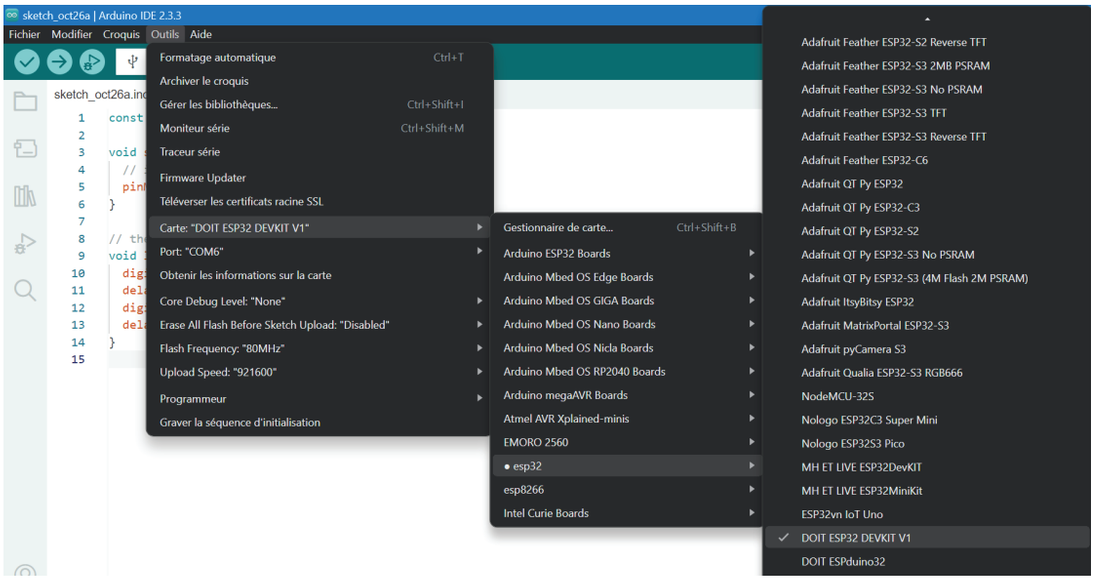
Suivez le schéma suivant pour connecter la LED à la carte ESP32 DEVKIT (vérifiez également où se trouve GND sur votre carte).



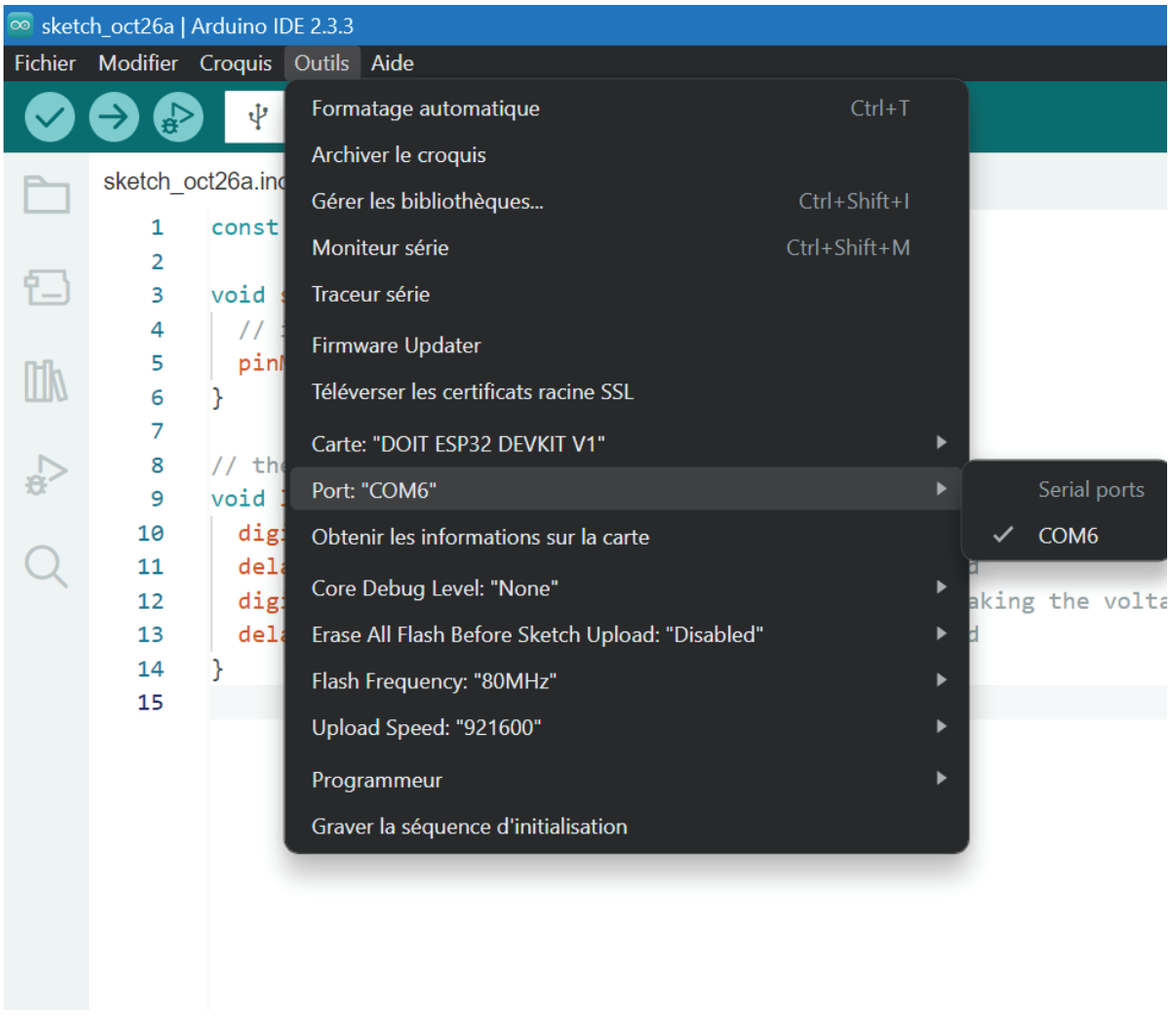
Ce schéma utilise la version du module ESP32 DEVKIT V1 avec 30 GPIO. Si vous utilisez un autre modèle, veuillez vérifier le brochage de la carte que vous utilisez.

Préparation de l'IDE Arduino

Après avoir connecté une LED au GPIO 18 de la carte ESP32, vous devez aller dans outils; Carte, faire défiler jusqu'à la section ESP32 et sélectionner le nom de votre carte ESP32 que vous avez trouvé précédemment. Dans mon cas, il s'agit de la carte DOIT ESP32 DEVKIT V1.



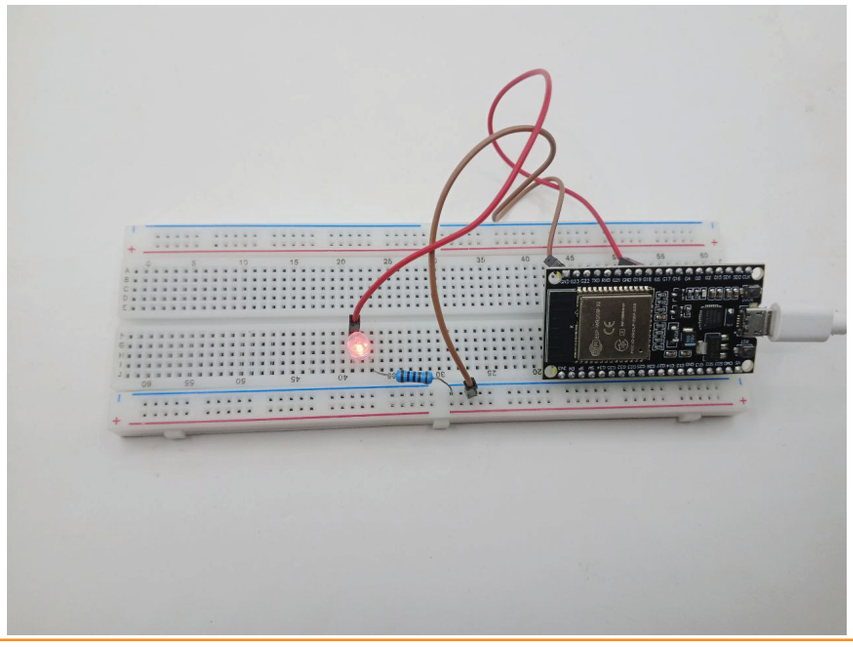
Tout en ayant l'ESP32 branché à votre ordinateur. Allez dans Outils; Port ; COM disponible



Téléchargement du croquis

Revenez à l'IDE Arduino, appuyez sur le bouton de téléchargement de l'IDE Arduino et attendez quelques secondes pendant qu'il compile et télécharge votre croquis.

La LED connectée au GPIO 18 doit clignoter toutes les secondes



Avant de passer directement à la construction du serveur Web, il est important de décrire ce que notre serveur Web fera, afin qu'il soit plus faci de suivre les étapes ultérieures

- Le serveur Web que vous allez créer contrôle deux LED connectées aux GPIO 26, et 27.
- Vous pouvez accéder au serveur Web ESP32 en saisissant l'adresse IP de l'ESP32 dans un navigateur dans le réseau local.
- En cliquant sur les boutons de votre serveur Web, vous pouvez instantanément modifier l'état de chaque LED.

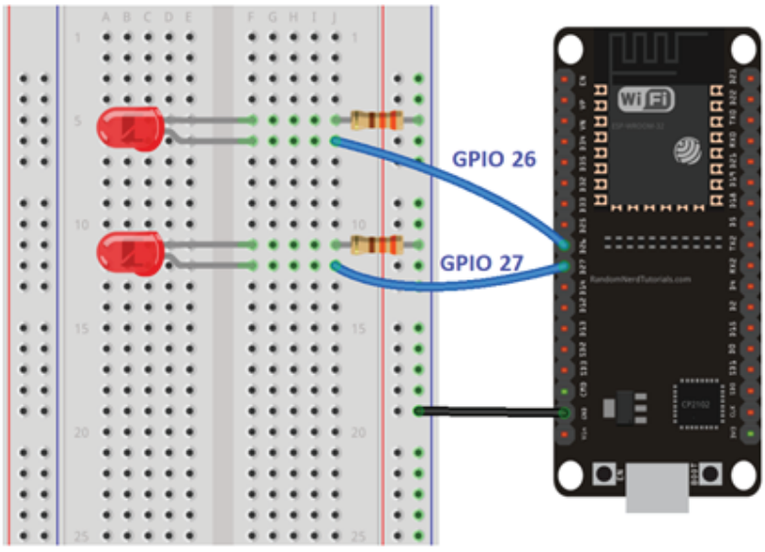
Ceci est juste un exemple simple pour illustrer comment construire un serveur Web qui contrôle les sorties, l'idée est de remplacer ces LED par un relais, ou tout autre composant électronique que vous voulez.

Schématique

Commencez par construire le circuit. Connectez deux LED à votre ESP32 comme indiqué dans le schéma suivant, avec une LED connectée au GPIO 26 et une autre au GPIO 27.

Voici une liste des pièces dont vous avez besoin pour assembler le circuit

- ESP32 DOIT DEVKIT V1
- 2x LED
- 2xResistances de 220 Ohms
- Breadboard
- Jumper



Ce schéma utilise la version du module ESP32 DEVKIT V1 avec 36 GPIO. Si vou utilisez un autre modèle, veuillez vérifier le brochage de la carte que vous utilisez.

Création du serveur Web

Après avoir câblé le circuit, l'étape suivante consiste à téléverser le code sur votre ESP32. Copiez le code ci-dessous sur votre IDE Arduino, mais ne le téléverser pas encore. Vous devez apporter quelques modifications pour que cela fonctionne pour vous.

Trouvez l'intégralité du code sur ce lien: https://docs.google.com/document/d/1gF4MGXGUCgXxRABjqucne9m3GH6Uv-FQQtyP2x95tlk/edit?usp=drive_link

Serveur_Web_ESP32.ino

```
// Inclusion de la bibliothèque Wi-Fi
#include <WiFi.h>

// Remplacement par vos identifiants réseau
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Configuration du serveur web sur le port 80
WiFiServer server(80);

// Variable pour stocker la requête HTTP entrante
String header;

// Variables pour stocker l'état actuel des sorties
String output26State = "off";
String output27State = "off";

// Assignation des broches GPIO aux sorties
const int output26 = 26;
const int output27 = 27;

// Temps actuel et précédent pour gérer les délais
unsigned long currentTime = millis();
unsigned long previousTime = 0;
// Délai d'attente maximal pour une requête (en millisecondes)
const long timeoutTime = 2000;

void setup() {
```

Définition de vos identifiants réseau

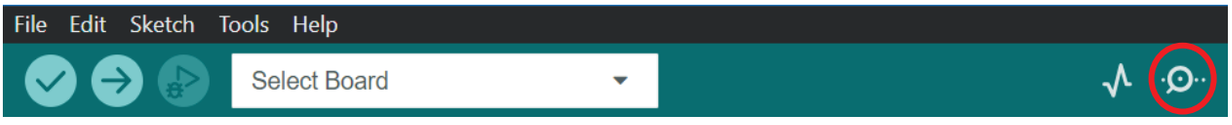
Vous devez modifier les lignes suivantes avec vos informations d'identification réseau : SSID et mot de passe.

Le code est bien commenté sur les endroits où vous devez apporter les modifications.

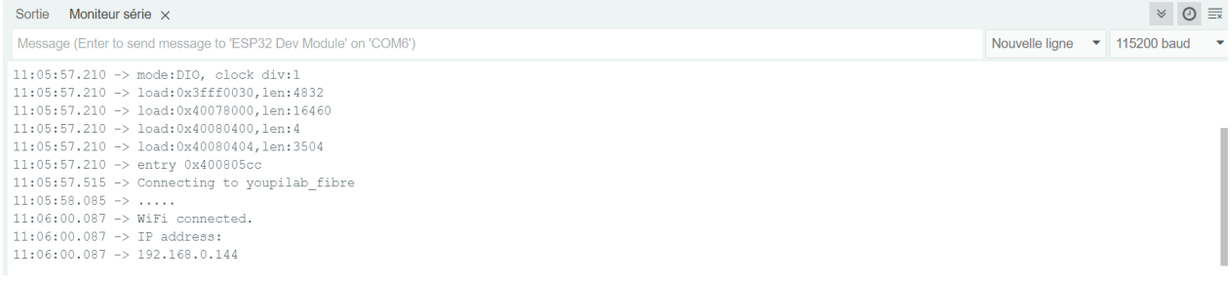
```
4   // Remplacement par vos identifiants réseau
5   const char* ssid = "REPLACE_WITH_YOUR_SSID";
6   const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Trouver l'adresse IP de l'ESP32

Vous pouvez maintenant télécharger le code et il fonctionnera immédiatement. N'oubliez pas de vérifier si vous avez sélectionné la bonne carte et le bon port COM, sinon vous obtiendrez une erreur lors de la tentative de téléchargement. Ouvrez le moniteur série à un débit en bauds de 115 200.



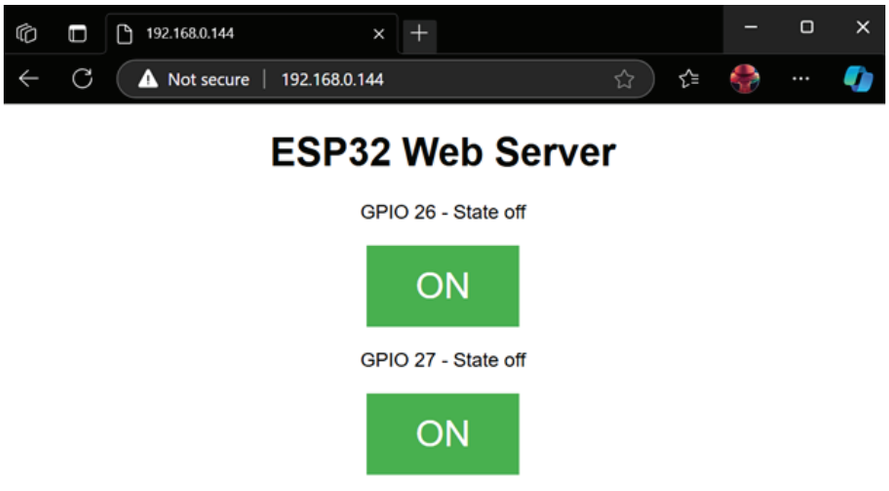
L'ESP32 se connecte au Wi-Fi et affiche l'adresse IP de l'ESP sur le moniteur série. Copiez cette adresse IP, car vous en avez besoin pour accéder au serveur Web ESP32.



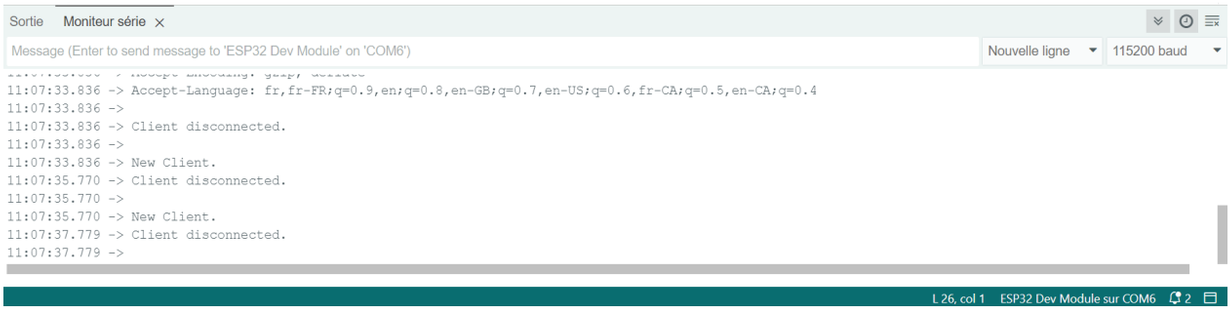
Remarque : si rien n'apparaît dans le moniteur série, appuyez sur le bouton « EN » de l'ESP32 (bouton d'activation à côté du port micro USB).

Accéder au serveur Web

Ouvrez votre navigateur, collez l'adresse IP ESP32 et vous verrez la page suivante.



Si vous jetez un œil au moniteur série, vous pouvez voir ce qui se passe en arrière-plan. L'ESP32 reçoit une requête HTTP d'un nouveau client (dans ce cas, votre navigateur). Vous pouvez également voir d'autres informations sur la requête HTTP, les champs d'entête HTTP qui définissent les paramètres de fonctionnement d'une transaction HTTP.

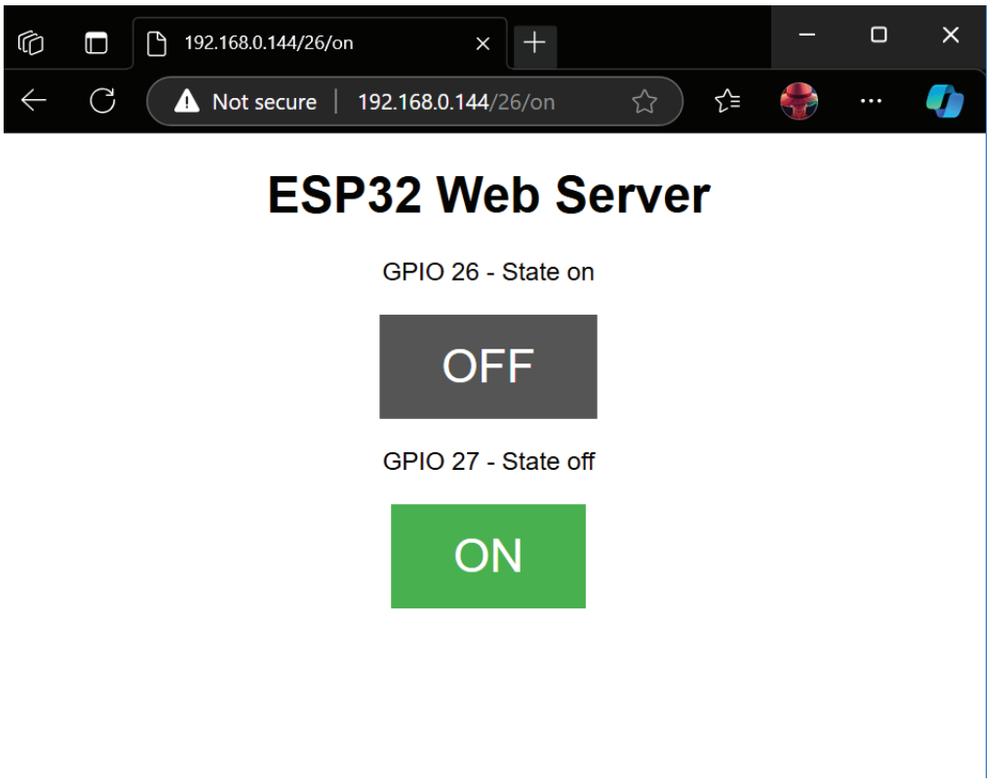


Tester le serveur Web

Testons le serveur Web. Cliquez sur le bouton pour activer GPIO26.Vous pouvez voir sur le moniteur série.

Surveillez que l'ESP32 reçoit une requête sur l'URL/26/ON.

Lorsque l'ESP32 reçoit cette demande; il allume la LED rattachée au GPIO 26 et son état est également mis à jour sur la page Web.



Testez le bouton pour le GPIO27 et voyez qu'il fonctionne de la même manière.

Comment fonctionne le code?

Maintenant, examinons de plus près le code pour voir comment il fonctionne, afin que vous puissiez le modifier pour répondre à vos besoins.

La première chose à faire est d'inclure la bibliothèque Wifi. Il s'agit de la même bibliothèque utilisée pour créer un serveur Web avec l'Arduino à l'aide du Shield Ethernet.

```
1   // Inclusion de la bibliothèque Wi-Fi
2   #include <WiFi.h>
```

Comme mentionné précédemment, vous devez insérer votre SSID et votre mot de passe dans les lignes suivantes à l'intérieur des guillemets.

```
4   // Remplacement par vos identifiants réseau
5   const char* ssid = "REPLACE_WITH_YOUR_SSID";
6   const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Ensuite, vous définissez votre serveur Websur le port 80.

```
8   // Configuration du serveur web sur le port 80
9   WiFiServer server(80);
```

La ligne suivante crée une variable pour stocker l'en-tête de la requête http

```
11  // Variable pour stocker la requête HTTP entrante
12  String header;
```

Ensuite, vous créez des variables auxiliaires pour stocker l'état actuel de vos sorties. Si vous souhaitez ajouter d'autres sorties et enregistrer leur état, vous devez créer davantage de variables.

```
14  // Variables pour stocker l'état actuel des sorties
15  String output26State = "off";
16  String output27State = "off";
```

Vous devez également attribuer un GPIO à chacune de vos sorties. Ici, nous utilisons le GPIO 26 et le GPIO 27. Vous pouvez utiliser tout autre GPIO approprié.

```
18  // Assignation des broches GPIO aux sorties
19  const int output26 = 26;
20  const int output27 = 27;
```

setup()

Passons maintenant à la fonction setup(). La fonction setup() ne s'exécute qu'une seule fois lorsque votre ESP est activé pour la première fois. Tout d'abord, nous démarrons une communication série à un débit en bauds de 115 200 à des fins de débogage.

```
29  // Initialisation de la communication série pour le débogage
30  Serial.begin(115200);
```

Vous définissez également vos GPIO comme SORTIES et les réglez sur LOW

```
32  // Configuration des broches GPIO comme sorties
33  pinMode(output26, OUTPUT);
34  pinMode(output27, OUTPUT);
35
36  // Initialisation des sorties à l'état bas
37  digitalWrite(output26, LOW);
38  digitalWrite(output27, LOW);
39
```

```
  // Connexion au réseau Wi-Fi
  Serial.print("Connexion à ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Affichage de l'adresse IP locale et démarrage du serveur web
  Serial.println("");
  Serial.println("WiFi connecté.");
  Serial.println("Adresse IP: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

loop()

Dans la fonction loop(), nous programmons ce qui se passe lorsqu'un nouveau client établit une connexion avec le serveur web.

L'ESP est toujours à l'écoute des clients entrants avec cette ligne :

```
  // Attente d'une nouvelle connexion client
  WiFiClient client = server.available();
```

Lorsqu'une requête est reçue d'un client, nous enregistrons les données entrantes. La boucle while qui suit s'exécutera tant que le client restera connecté. Nous vous déconseillons de modifier la partie suivante du code à moins que vous ne sachiez exactement ce que vous faites.

```
if (client) { // Si un nouveau client se connecte
  // Initialisation des variables de temps pour gérer le timeout
  currentTime = millis();
  previousTime = currentTime;
  Serial.println("Nouveau client connecté.");

  // Boucle tant que le client est connecté et que le délai n'est pas dépassé
  while (client.connected() && currentTime - previousTime <= timeoutTime) {
    currentTime = millis();

    // Lecture des données envoyées par le client
    if (client.available()) {
      char c = client.read();
      Serial.write(c); // Affichage des données reçues sur le moniteur série
      header += c; // Ajout du caractère à la requête HTTP complète

      // Fin de la requête HTTP détectée
      if (c == '\n' && currentLine.length() == 0) {
        // Envoi de la réponse HTTP
        client.println("HTTP/1.1 200 OK");
        client.println("Content-type:text/html");
        client.println("Connection: close");
        client.println();
```

La section suivante des instructions if et else vérifie quel bouton a été appuyé sur votre page Web et contrôle les sorties en conséquence. Comm nous l'avons vu précédemment, nous faisons une requête sur différentes URL en fonction du bouton sur lequel nous appuyons.

```
85          // Analyse de la requête pour contrôler les GPIO
86          if (header.indexOf("GET /26/on") >= 0) {
87            Serial.println("Activation du GPIO 26");
88            output26State = "on";
89            digitalWrite(output26, HIGH);
90          } else if (header.indexOf("GET /26/off") >= 0) {
91            Serial.println("Désactivation du GPIO 26");
92            output26State = "off";
93            digitalWrite(output26, LOW);
94          } // ... (même chose pour le GPIO 27)
```

Par exemple, si vous avez appuyé sur le bouton ON du GPIO 26, l'ESP reçoit une requête sur l'URL /26/ON et nous recevons ces informations sur l'en-tête HTTP. Nous pouvons donc vérifier si l'en -tête contient l'expression GET /26/on. Si c'est le cas, il affichera un message sur le moniteur série, il changera la variable output26state sur ON et allumera la LED. Et ce qui suit est utilisé pour empêcher les requêtes sur le favicon. Vous n'avez pas à vous soucier de cette ligne. Cela fonctionne de la même manière pour les autres boutons. Ainsi, si vous souhaitez ajouter d'autres sorties, vous devez modifier cette partie du code pour les inclure. Affichage de la page Web HTML client.println(""); La prochaine étape consiste à créer la page Web. L'ESP32 enverra une réponse à votre navigateur avec du code HTML pour créer la page Web. Styliser la page Web La page Web est envoyée au clien à l'aide de cette expression client.println(). Vous devez entrer ce que vous souhaitez envoyer au client comme argument. Remarque : dans les unités 3 et 4, vous découvrirez les bases d u HTML et du CSS, afin de pouvoir facilement modifier la page Web pour répondre à vos besoins. Ensuite, nou avons du texte CSS pour styliser les boutons et l'apparence de la page Wel Nous choisissons la police Helvetica, définissons le contenu à afficher sous forme de bloc et l'alignons au La première chose que nous devons envoyer est toujours la ligne suivante qui indique que nous envoyons du HTML.

```
96          // Construction et envoi de la page HTML
97          client.println("<!DOCTYPE html><html>");
```

Ensuite, la ligne suivante rend la page Web réactive dans n'importe quel navigateur Web.

```
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
```

Et ce qui suit est utilisé pour empêcher les requêtes sur le favicon. – Vous n'avez pas à vous soucier de cette ligne.

```
client.println("<link rel=\"icon\" href=\"data:,\">");
```

Styliser la page Web. La page Web est envoyée au client à l'aide de cette expression client.println(). Vous devez entrer ce que vous souhaitez envoyer au client comme argument. Remarque : dans les unités 3 et 4, vous découvrirez les bases du HTML et du CSS, afin de pouvoir facilement modifier la page Web pour répondre à vos besoins. Ensuite, nous avons du texte CSS pour styliser les boutons et l'apparen ce de la page Web. Nous choisissons la police Helvetica, définissons le contenu à afficher sous forme de bloc et l'alignons au centre.

```
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
```

Nous stylisons nos boutons avec la couleur #4CAF50, sans bordure, le texte en couleur blanche et avec ce remplissage : 16px 40px. Nous définissons également la décoration du texte sur aucune, définissons la taille de la police, la marge et le curseur sur un pointeur.

```
client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
```

Nous définissons également le style d'un deuxième bouton, avec toutes les propriétés du bouton quenous avons défini précédemment, mais avec une couleur différente. Ce sera le style du bouton d'arrêt.

```
client.println(".button2 {background-color: #555555;}</style></head>");
```

Définition du premier titre de la page Web.

Dans la ligne suivante, vous pouvez définir le premier titre de votre page Web. Ici, nous avons « Serveur Web ESP32 », mais vous pouvez modifier ce texte comme vous le souhaitez.

```
// En-tête de la page web
client.println("<body><h1>Serveur Web ESP32</h1>");
```

Affichage des boutons et de l'état correspondant

Ensuite, vous écrivez un paragraphe pour afficher l'état actuel du GPIO 26. Comme vous pouvez le voir, nous utilisons la variable output26State, de sorte que l'état se met à jour instantanément lorsque cette variable change.

```
// Afficher l'état actuel et les boutons ON/OFF pour GPIO 26
client.println("<p>GPIO 26 - État " + output26State + "</p>");
```

Ensuite, nous affichons le bouton marche ou arrêt, en fonction de l'état actue du GPIO. Si l'état actuel du GPIO est éteint, nous affichons le bouton ONsinon, nous affichons le bouton OFF.

```
// Afficher l'état actuel et les boutons ON/OFF pour GPIO 26
client.println("<p>GPIO 26 - État " + output26State + "</p>");
// Si l'état de GPIO 26 est off, afficher le bouton ON
if (output26State=="off") {
  client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");
} else {
  client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");
}
```

Nous utilisons la même procédure pour GPIO 27.

Fermeture de la connexion

Enfin, lorsque la réponse se termine, nous effaçons la variable d'en -tête et arrêtons la connexion avec le client avec client.stop().

```
// Réinitialiser la variable header
header = "";
// Fermer la connexion
client.stop();
```

Pour conclure

Maintenant que vous savez comment fonctionne le code, vous pouvez le modifier pour ajouter d'autres sorties ou modifier votre page Web. Pour modifier votre page Web, vous devrez peut-être connaître quelques notions de base de HTML et de CSS.