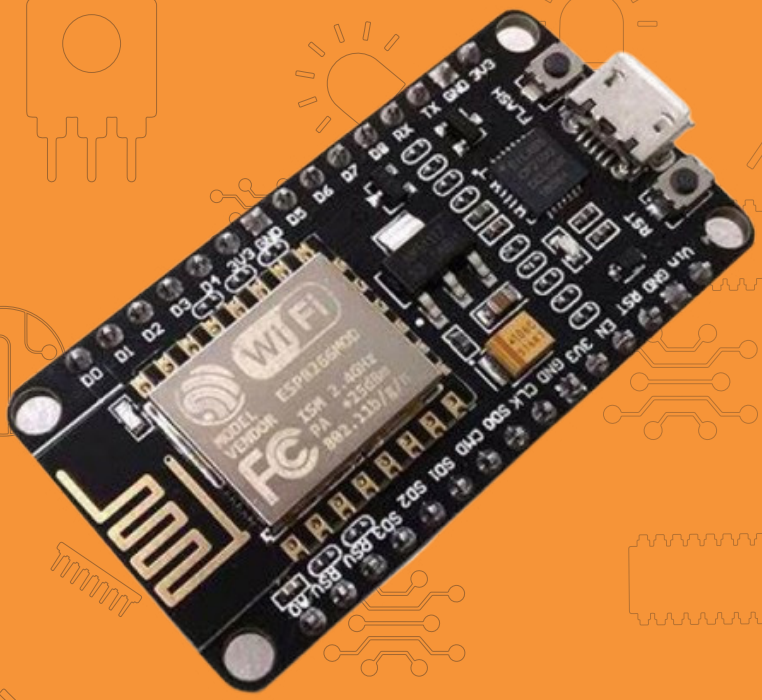
YoupiLab

DEMYSTIFYING ELECTRONICS

SERVEUR WEB ESP8266



GUIDE DE PROJET ÉTAPE PAR ÉTAPE

Serveur Web ESP8266 avec Arduino IDE

Bonjour et merci d'avoir téléchargé ce projet eBook !

Ce livre électronique rapide est notre guide étape par étape conçu pour vous aider à créer un site Web serveur avec le module Wifi appelé ESP8266.

À propos de l'ESP8266

L'ESP8266 est un module Wi-Fi à 1400 Fcfa (jusqu'à 5000 Fcfa). Il vous permet de contrôler les entrées et les sorties comme vous le feriez avec un Arduino, mais il est doté du Wi-Fi.

Il est donc idéal pour les applications de domotique et d'Internet des objets.

Alors, que pouvez-vous faire avec ce module à faible coût ?

Vous pouvez créer un serveur Web, envoyer des requêtes HTTP, contrôler les sorties, lire les entrées et les interruptions, envoyer des Courriels, publier des tweets, créer des gadgets IoT et bien plus encore.

Spécifications ESP8266

Les spécifications de l'ESP8266 décrivent les principales fonctionnalités et capacités de ce microcontrôleur, souvent utilisé pour des projets IoT (Internet des objets).

Voici une explication détaillée :

- **Protocole 802.11 b/g/n :**
  - L'ESP8266 prend en charge les normes Wi-Fi 802.11 b/g/n, qui sont des normes de communication sans fil utilisées dans les réseaux locaux sans fil (Wi-Fi).
  - 802.11b est une norme plus ancienne avec une vitesse maximale de 11 Mbps.
  - 802.11g offre jusqu'à 54 Mbps.

- 802.11n est plus rapide, avec des débits allant jusqu'à 600 Mbps, et fonctionne à la fois sur les bandes de 2,4 GHz et 5 GHz.

- **Wi-Fi Direct (P2P), point d'accès logiciel :**
  - Wi-Fi Direct permet une communication directe entre appareils sans passer par un routeur ou un point d'accès. Cela facilite la mise en réseau entre appareils sans configuration réseau complexe.
  - L'ESP8266 peut également être configuré comme un point d'accès logiciel (soft AP), c'est-à-dire qu'il peut créer un réseau Wi-Fi auquel d'autres appareils peuvent se connecter. Cela est utile pour des projets où l'ESP8266 doit être autonome et héberger son propre réseau.
- **Pile de protocoles TCP/IP intégrée :**
  - L'ESP8266 intègre une pile TCP/IP (Transmission Control Protocol / Internet Protocol), qui est un ensemble de protocoles permettant à l'appareil de communiquer sur des réseaux, notamment l'Internet. Cela inclut les protocoles TCP pour établir des connexions fiables, et IP pour adresser et acheminer les données. Grâce à cela, l'ESP8266 peut gérer directement des communications réseau sans avoir besoin d'un processeur externe.
- **Processeur 32 bits basse consommation intégré :**
  - L'ESP8266 dispose d'un processeur 32 bits, capable de traiter des instructions complexes plus rapidement qu'un processeur 8 ou 16 bits. Il est conçu pour être économe en énergie, ce qui le rend idéal pour des projets alimentés par batterie ou des dispositifs où l'efficacité énergétique est essentielle.
- **SDIO 2.0, SPI, UART :**
  - SDIO 2.0 (Secure Digital Input/Output) : Une interface utilisée pour connecter des cartes SD ou d'autres périphériques de stockage.
  - SPI (Serial Peripheral Interface) : Un protocole de communication rapide utilisé pour interagir avec des périphériques comme des capteurs, des écrans ou des mémoires.
  - UART (Universal Asynchronous Receiver-Transmitter) : Une interface de communication série couramment utilisée pour la communication avec d'autres microcontrôleurs ou pour le débogage.

Trouvez votre ESP8266

L'ESP8266 est disponible dans une grande variété de versions. L'ESP-12E ou souvent appelé ESP-12E NodeMCU Kit est actuellement la version la plus pratique et c'est le module que nous utiliserons le plus tout au long de ce projet. Vous pouvez trouver cette carte ESP8266 sur le lien suivant disponible dans notre boutique YoupiLab.

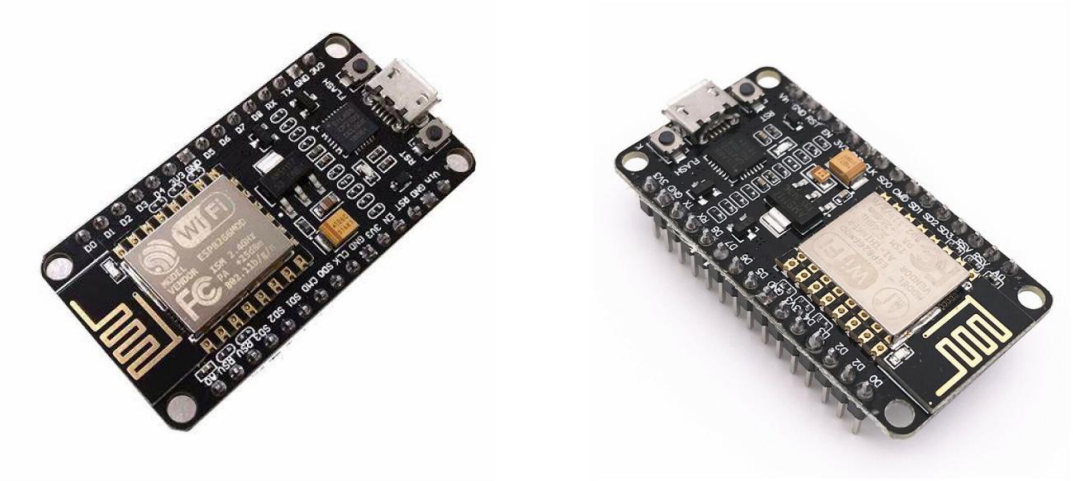
https://youpilab.com/components/product/module-wifi-node-mcu-esp8266-ch340g

Ce projet a été testé avec ESP-01, ESP-07, ESP-12 et ESP-12E. Donc vous pouvez suivre ce guide de projet avec n'importe laquelle de ces cartes.

Nous vous recommandons vivement d'utiliser le l'ESP8266-12E NodeMCU, celui qui dispose d'un programmateur intégré. Le programmateur intégré facilite le prototypage et le téléchargement de vos programmes.

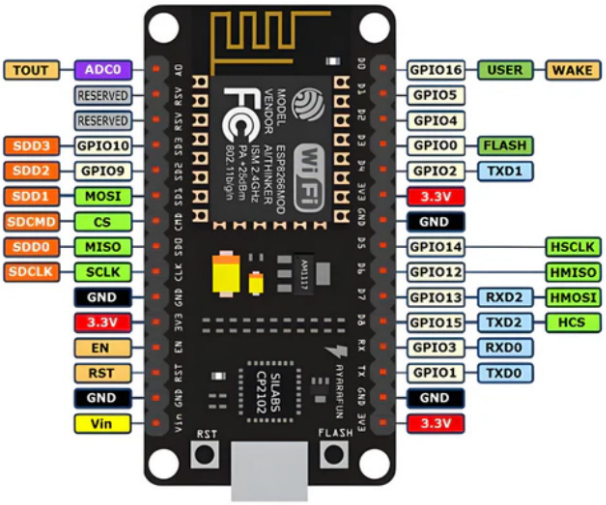
Vous pourriez également trouver utile de consulter l'article suivant :

https://youpilab.com/components/product/module-wifi-node-mcu-esp-8266-12



Brochage de l'ESP8266-12E NodeMCU

Voici un aperçu rapide du brochage du NodeMCU ESP-12E :



ESP8266 avec Arduino IDE

Dans cette section, vous allez télécharger, installer et préparer votre IDE Arduino pour fonctionner avec l'ESP8266. Vous pouvez programmer votre ESP8266 à l'aide du langage de programmation Arduino convivial.

Qu'est-ce que l'IDE Arduino ?

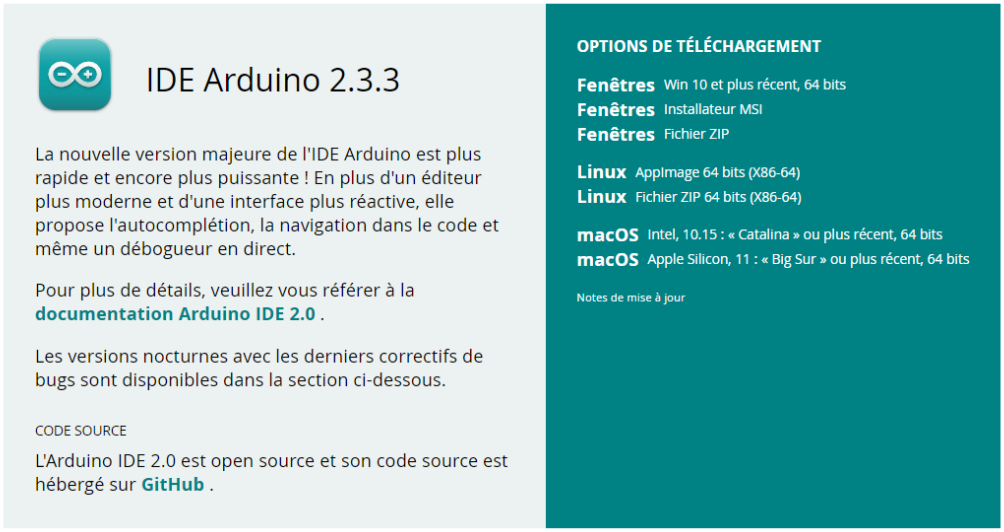
L'IDE Arduino est un logiciel open source qui facilite l'écriture de code et son téléchargement sur la carte Arduino.

L'IDE Arduino est un logiciel multiplateforme, ce qui signifie qu'il fonctionne sous Windows, Mac OS X ou Linux (il a été créé en JAVA).

Téléchargement de l'IDE Arduino

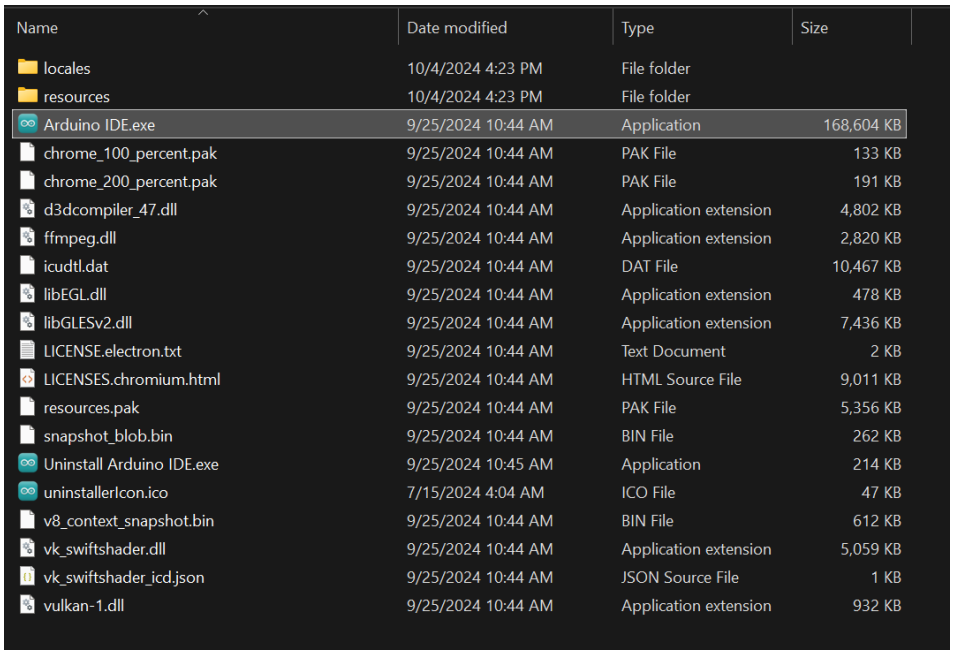
Pour télécharger l'IDE Arduino, visitez l'URL suivante : https://www.arduino.cc/en/software

Ensuite, sélectionnez votre système d'exploitation et téléchargez le logiciel (comme indiqué ci-dessous).



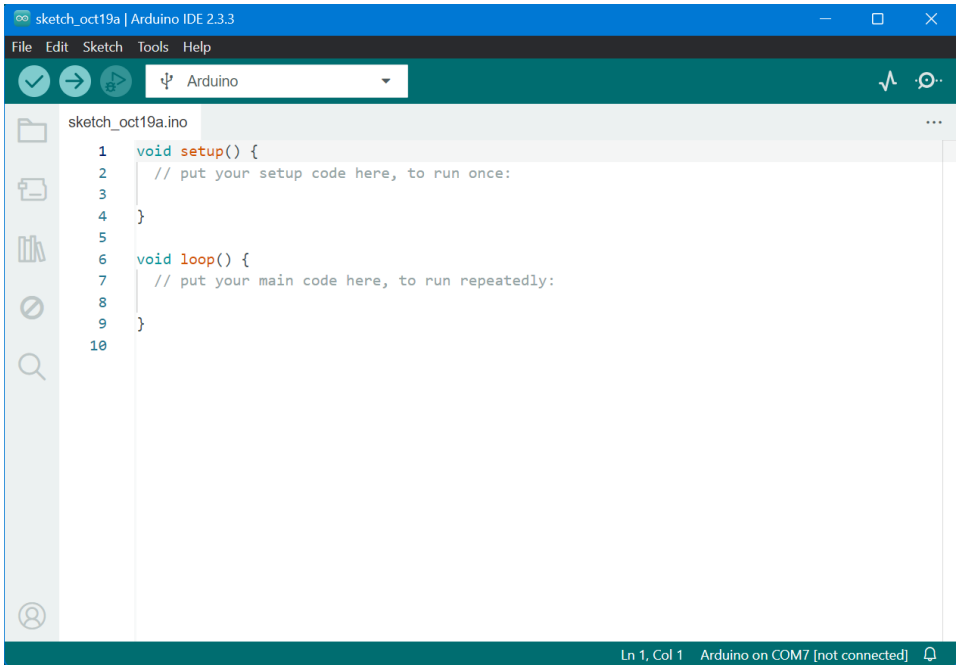
Installation de l'IDE Arduino

Récupérez le fichier que vous venez de télécharger et qui s'appelle « Arduino-(...).zip ». Exécutez ce fichier et suivez l'assistant d'installation qui s'affiche à l'écran. Ouvrez le fichier d'application Arduino IDE (voir la figure ci-dessous).



Name	Date modified	Type	Size
locales	10/4/2024 4:23 PM	File folder	
resources	10/4/2024 4:23 PM	File folder	
Arduino IDE.exe	9/25/2024 10:44 AM	Application	168,604 KB
chrome_100_percent.pak	9/25/2024 10:44 AM	PAK File	133 KB
chrome_200_percent.pak	9/25/2024 10:44 AM	PAK File	191 KB
d3dcompiler_47.dll	9/25/2024 10:44 AM	Application extension	4,802 KB
ffmpeg.dll	9/25/2024 10:44 AM	Application extension	2,820 KB
icudtl.dat	9/25/2024 10:44 AM	DAT File	10,467 KB
libEGL.dll	9/25/2024 10:44 AM	Application extension	478 KB
libGLESv2.dll	9/25/2024 10:44 AM	Application extension	7,436 KB
LICENSE.electron.txt	9/25/2024 10:44 AM	Text Document	2 KB
LICENSES.chromium.html	9/25/2024 10:44 AM	HTML Source File	9,011 KB
resources.pak	9/25/2024 10:44 AM	PAK File	5,356 KB
snapshot_blob.bin	9/25/2024 10:44 AM	BIN File	262 KB
Uninstall Arduino IDE.exe	9/25/2024 10:45 AM	Application	214 KB
uninstallerIcon.ico	7/15/2024 4:04 AM	ICO File	47 KB
v8_context_snapshot.bin	9/25/2024 10:44 AM	BIN File	612 KB
vk_swiftshader.dll	9/25/2024 10:44 AM	Application extension	5,059 KB
vk_swiftshader_icd.json	9/25/2024 10:44 AM	JSON Source File	1 KB
vulkan-1.dll	9/25/2024 10:44 AM	Application extension	932 KB

Lorsque l'IDE Arduino s'ouvre pour la première fois, voici ce que vous devriez voir :

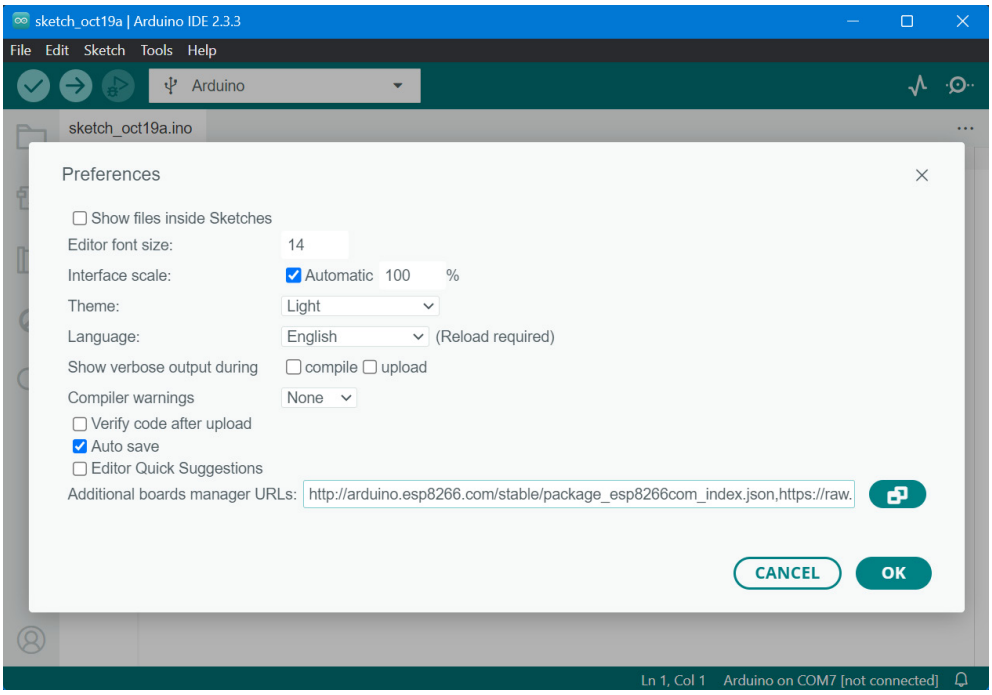


Installation de la carte ESP8266

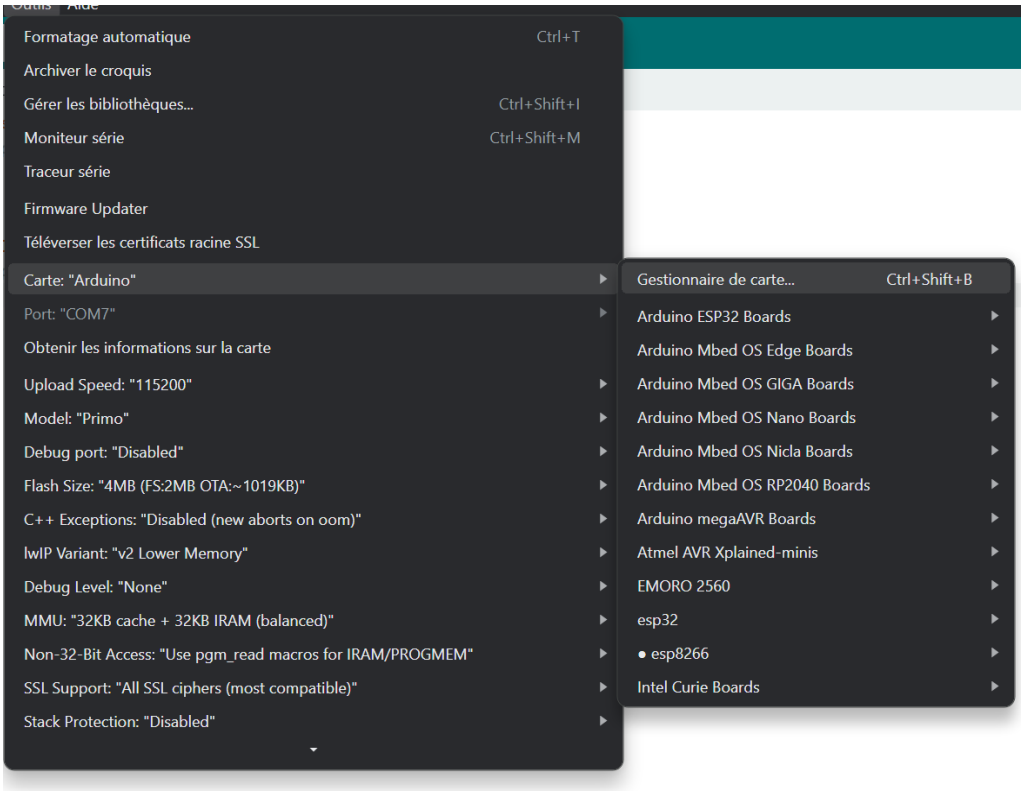
Pour installer la carte ESP8266 dans votre IDE Arduino, suivez les étapes suivantes

1. Ouvrez la fenêtre des fichiers depuis l'IDE Arduino.
2. Allez dans Fichier instructions puis préférence

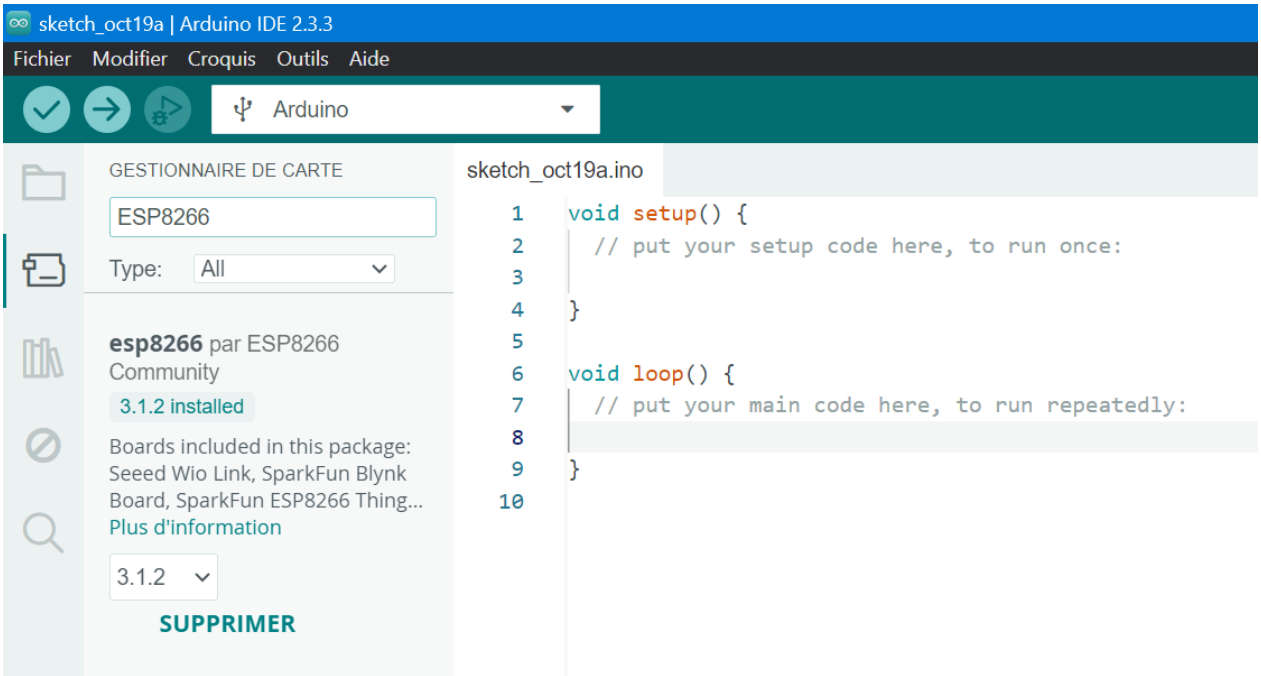
Entrez <<http://arduino.esp8266.com/stable/package_esp8266com_in Préférences dex.json>> dans le champ URL supplémentaires du gestionnaire de cartes et appuyez sur le bouton « OK » comme dans l'image ci-dessous :

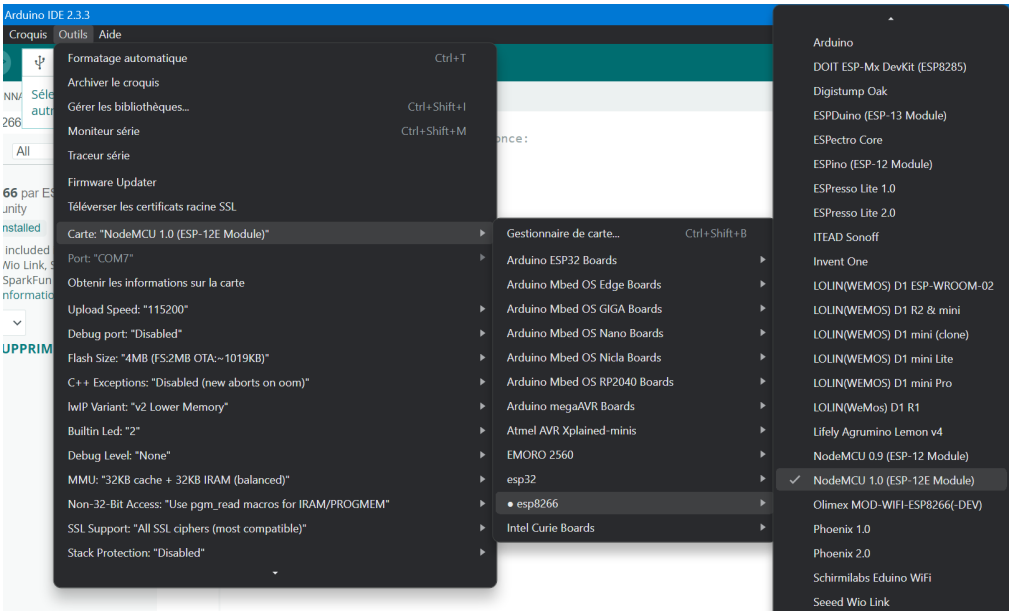


Allez ensuite dans Outils; carte et gestionnaire de carte



Sélectionnez le menu de la carte ESP8266 et installez « esp8266 by Community »





Allez dans Outils ; carte; ESP8266 puis NodeMCU 1.0(ESP-12E Module)

Enfin, rouvrez votre IDE Arduino pour vous assurer qu'il se lance avec les nouvelles cartes installées.

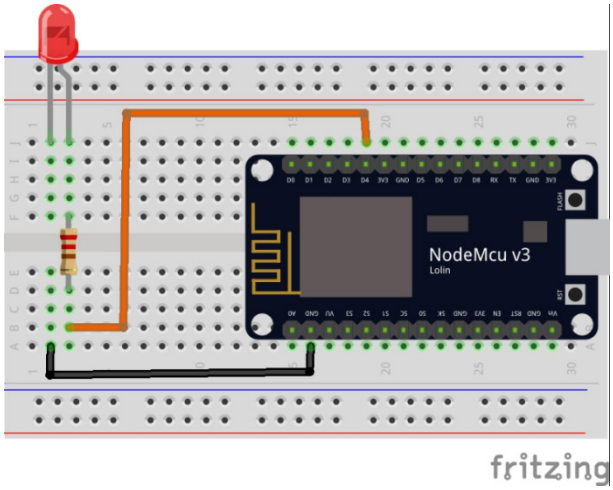
Faire clignoter une LED avec Arduino IDE

Dans cette section, vous allez concevoir un circuit simple pour faire clignoter une LED avec l'ESP8266 utilisant Arduino IDE. Pourquoi fait-on toujours clignoter une LED en premier ? C'est une excellente question ! Si vous pouvez faire clignoter une LED, vous pouvez pratiquement dire que vous pouvez allumer n'importe quel appareil électronique (allumé ou éteint.)

Rédiger votre croquis Arduino

Le schéma pour faire clignoter une LED est très simple.

Connectez une LED et une résistance de 220 Ohm à votre ESP8266 D4 (GPIO 2).



Code

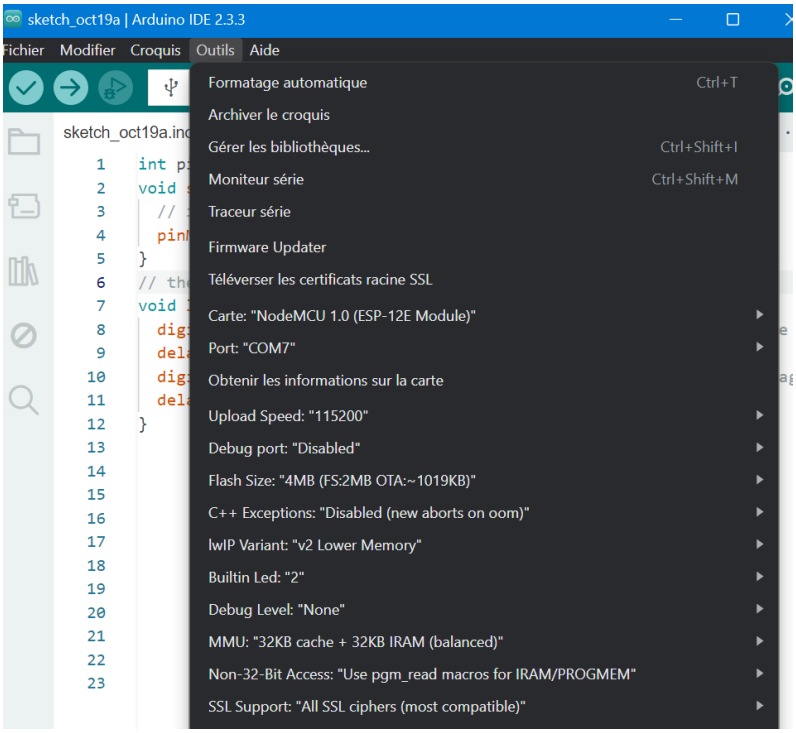
```
int pin = 2;
void setup() {
  // initialize GPIO 2 as an output
  pinMode(pin, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(pin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(pin, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Téléchargement du code sur ESP8266

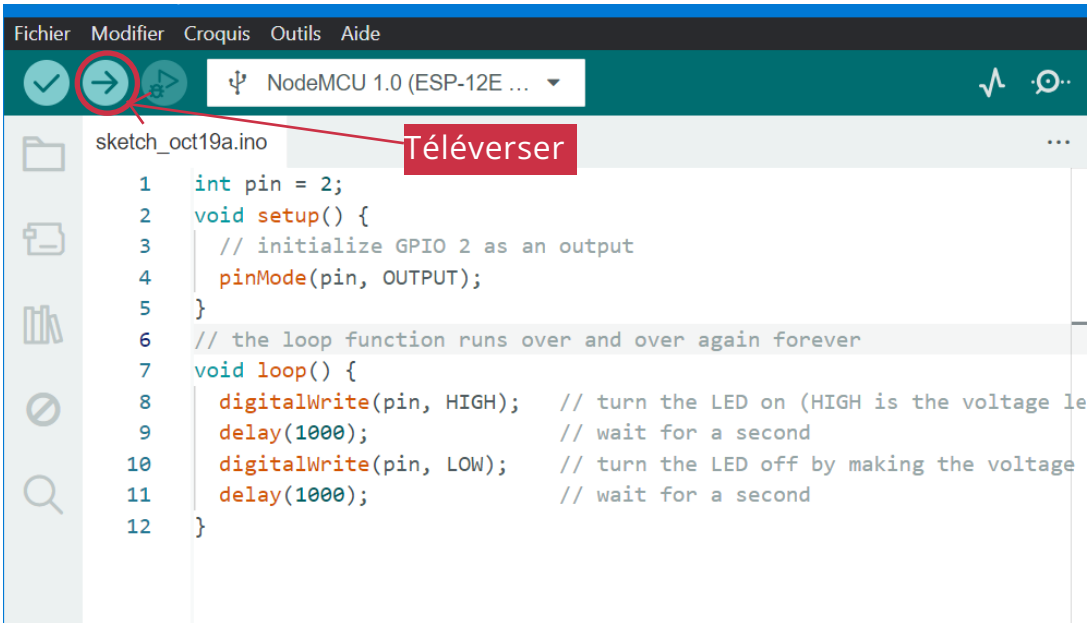
Le téléchargement de code sur votre kit ESP-12E NodeMCU est très simple, car il dispose d'un programmateur intégré. Vous branchez votre carte à votre ordinateur et vous n'avez pas besoin d'effectuer de connexions supplémentaires.

Regardez le menu Outils, sélectionnez Carte « NodeMCU 1.0 (module ESP-12E) » et toutes les configurations, par défaut, devraient ressembler à ceci :

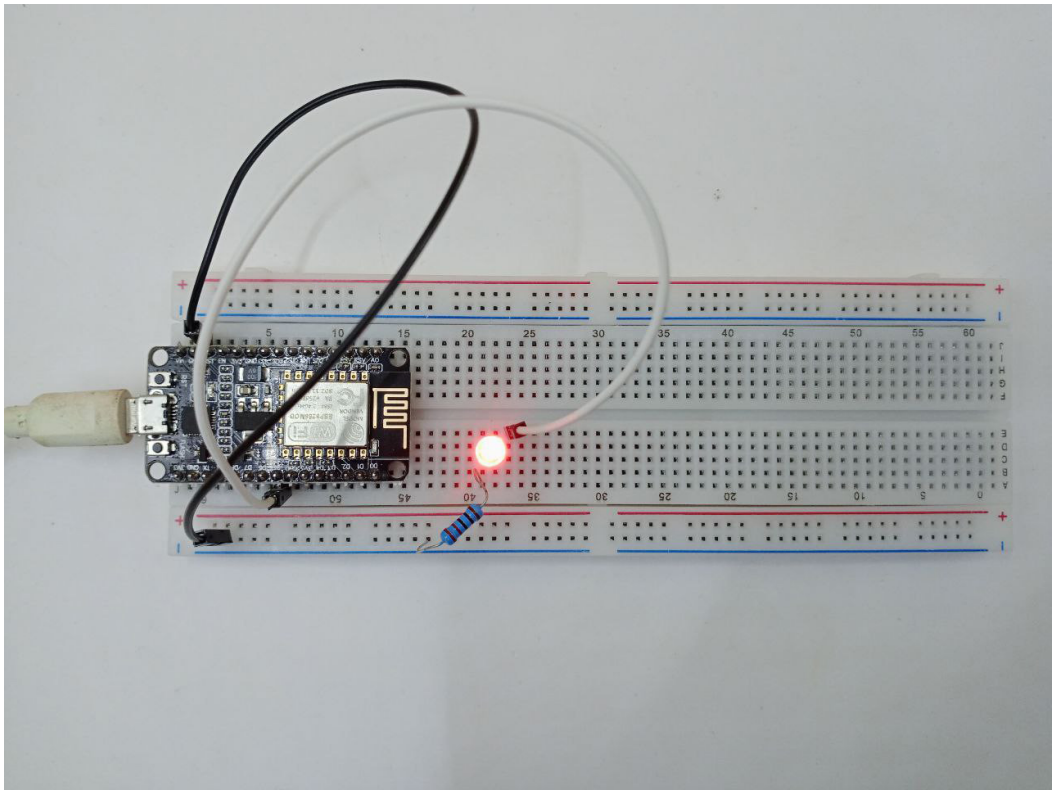
Important : il est fort probable que votre port COM soit différent de celui de la capture d'écran précédente (Port : « COM7 »). Ce n'est pas un problème, car cela n'interfère avec rien. En revanche, toutes les autres configurations devraient ressembler exactement à comme le mien.



Après avoir vérifié les configurations, cliquez sur le bouton « Téléverser » dans l'IDE Arduino et attendez quelques secondes jusqu'à ce que vous voyiez le message « Téléversement fait. » dans le coin inférieur droit.

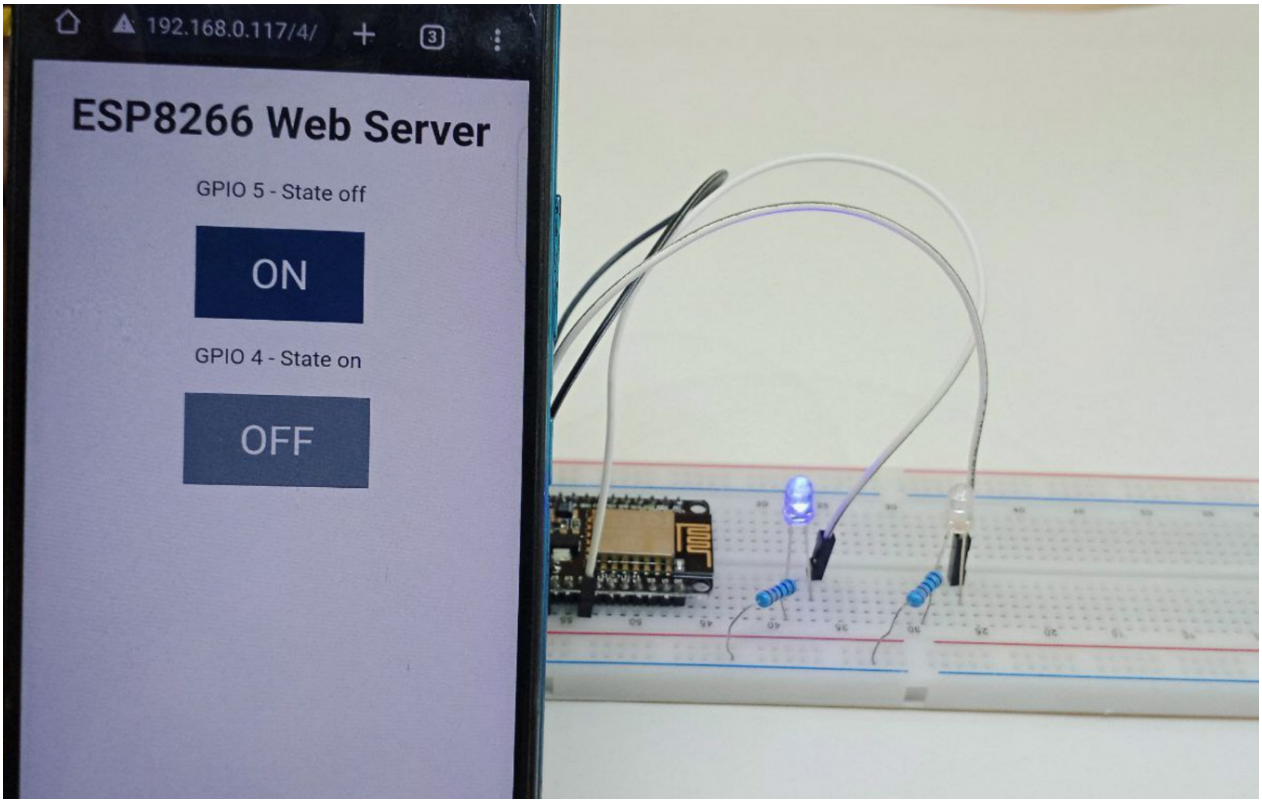


Redémarrez votre ESP8266. Félicitations, vous avez réussi ! Votre LED doit clignoter toutes les 1 seconde !



Serveur Web ESP8266

Ce tutoriel est un guide étape par étape qui vous montre comment créer un serveur Web ESP8266 autonome qui contrôle deux sorties (deux LED). Vous pouvez ensuite remplacez ces LED par d'autres appareils électroniques.



Ce serveur Web ESP8266 est adapté aux appareils mobiles et peut être consulté à partir de n'importe quel appareil doté d'un navigateur sur votre réseau local. Le code de ce projet est réalisé à l'aide d'Arduino IDE.

Code

Copiez le code ci-dessous dans votre IDE Arduino, mais ne le téléchargez pas encore. Vous devez apporter quelques modifications pour le faire fonctionner.

Le code en entier ici :

sketch_oct19b.ino

```
#include <ESP8266WiFi.h>
// Replace with your network credentials
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output5State = "off";
String output4State = "off";

// Assign output variables to GPIO pins
const int output5 = 5;
const int output4 = 4;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
```

Vous devez entification réseau, afin r.

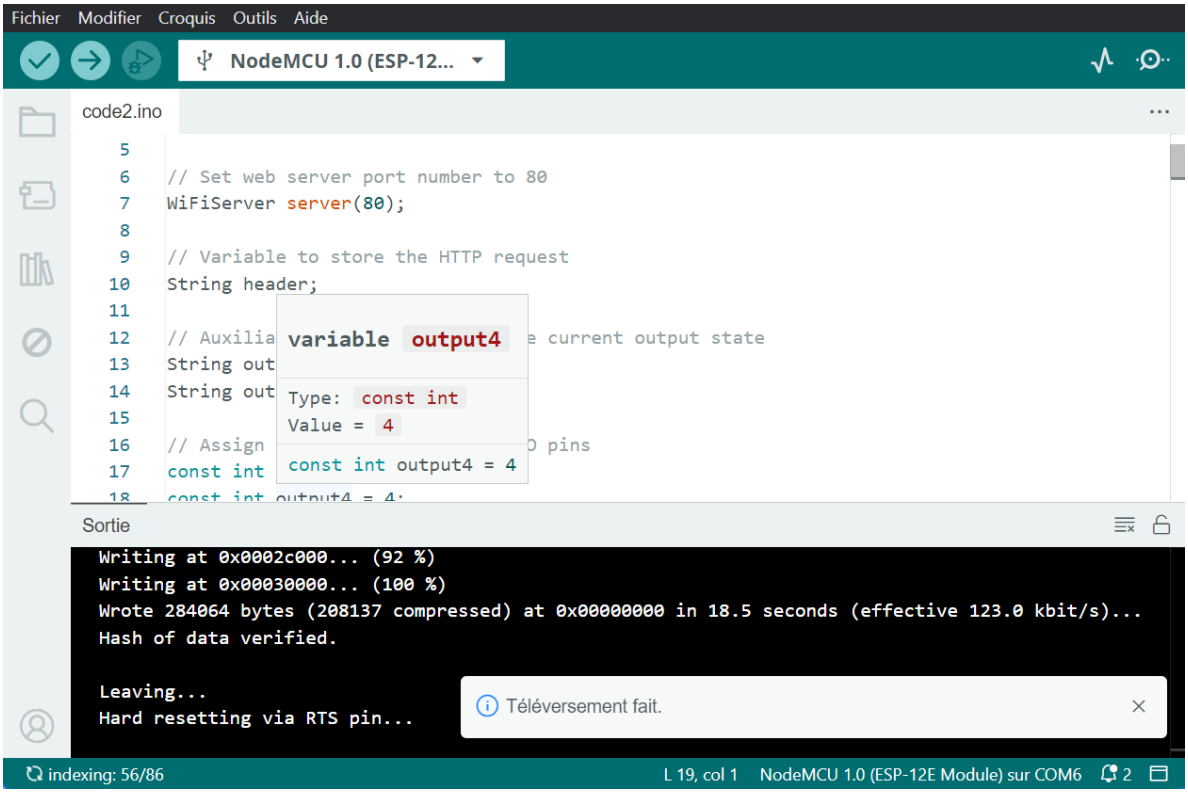
```
// Replace with your network credentials
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

```

Téléchargement du croquis

Téléchargement du croquis sur l'ESP-12E Si vous utilisez un kit ESP-12E NodeMCU, le téléchargement du croquis est très simple, car il dispose d'un programmateur intégré. Branchez votre carte à votre ordinateur. Assurez-vous d'avoir sélectionné la bonne carte et le bon port COM.

Ensuite, cliquez sur le bouton « Téléverser » dans l'IDE Arduino et attendez quelques secondes

jusqu'à ce que vous voyiez le message « Téléversement fait. » dans le coin inférieur droit.

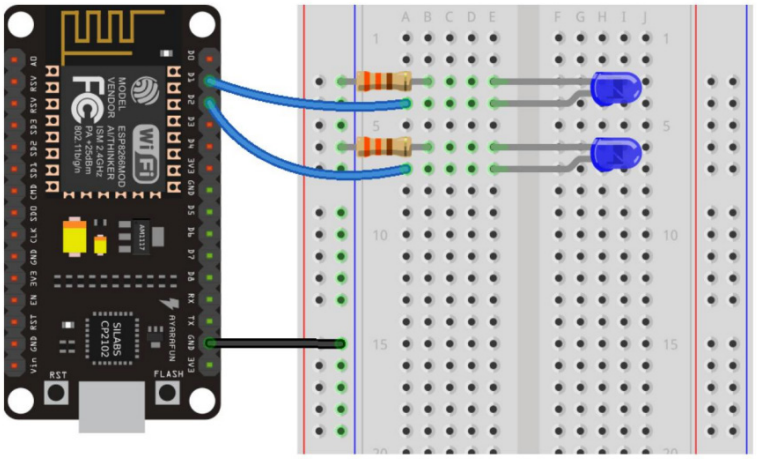


Schémas

Pour construire le circuit, vous avez besoin des pièces suivantes :

- ESP8266
- 2x LEDS
- 2xRésistances de 220 ou 330 ohms
- Breadboard
- Jumpers

Connectez deux LED à votre ESP8266 comme indiqué dans le schéma suivant – avec une LED connectée au GPIO 4 et une autre au GPIO 5.



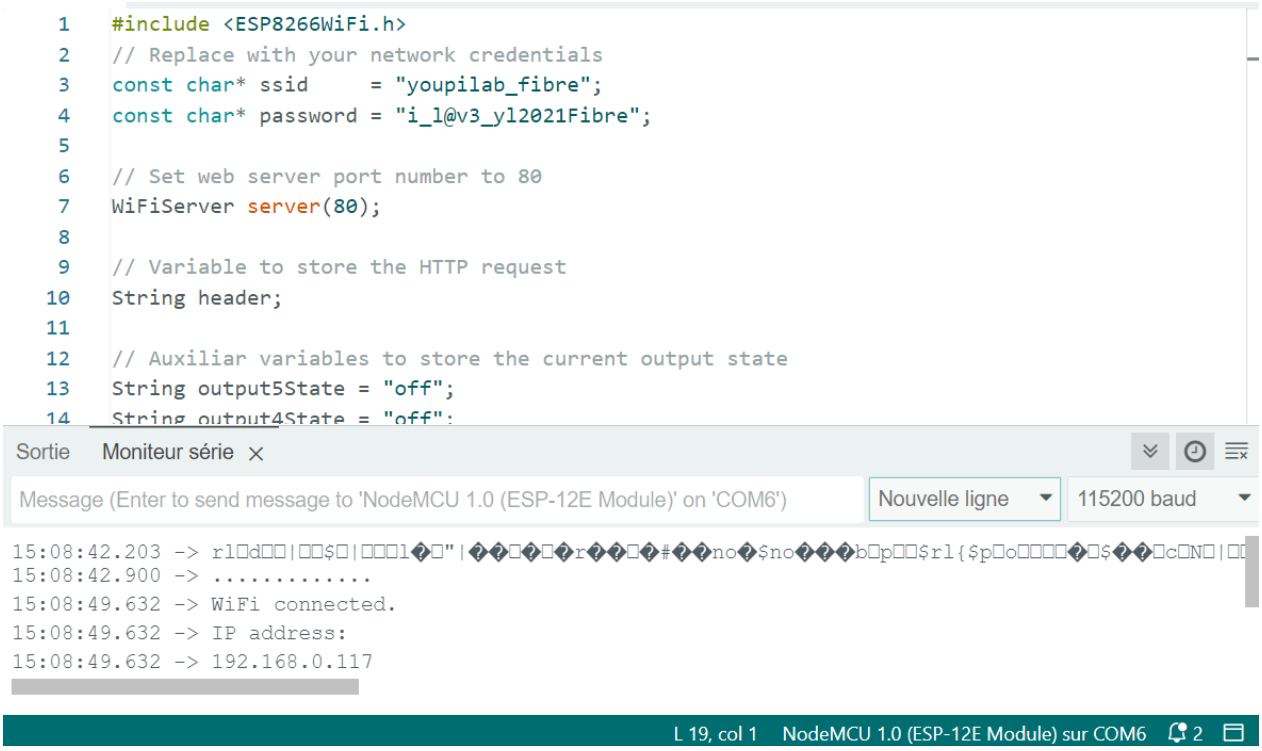
Tester le serveur Web

Maintenant, vous pouvez 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 115200.

Trouver l'adresse IP de l'ESP8266

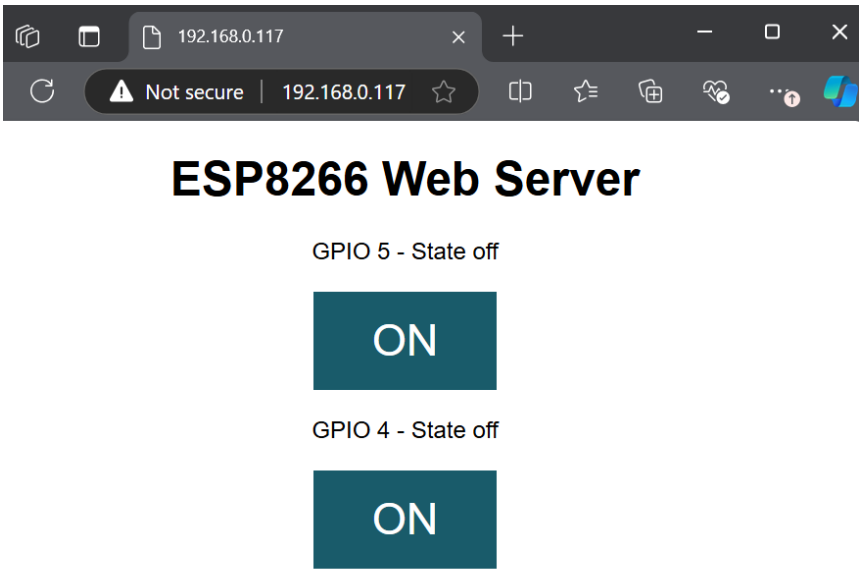
Appuyez sur le bouton RESET de l'ESP8266 et il affichera l'adresse IP de l'ESP sur le moniteur série



Copiez cette adresse IP, car vous en avez besoin pour accéder au serveur Web.

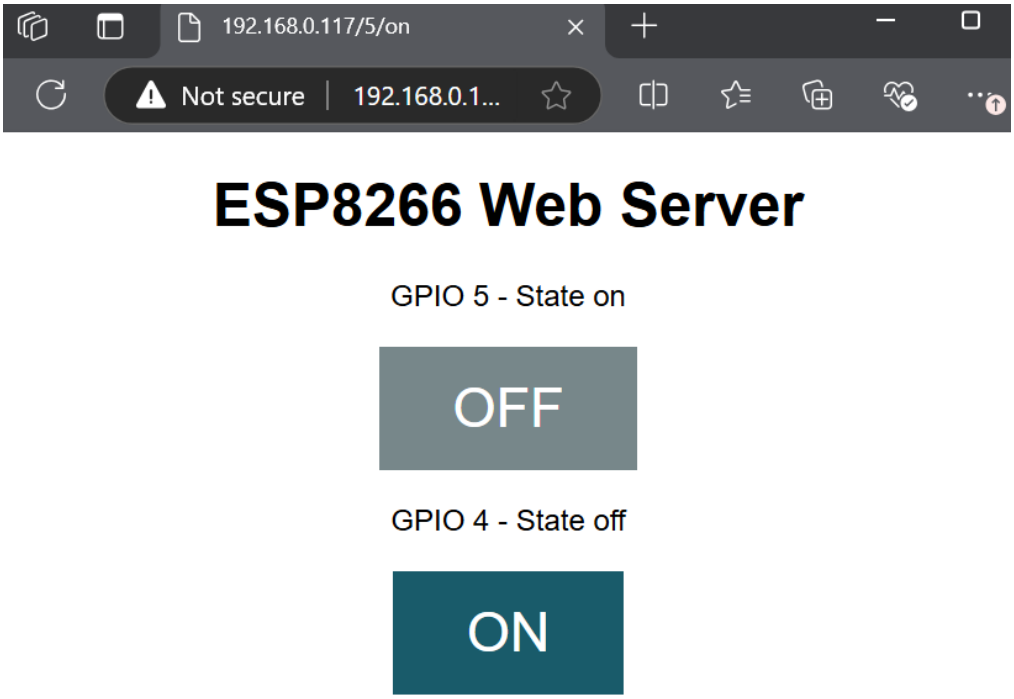
Accéder au serveur Web

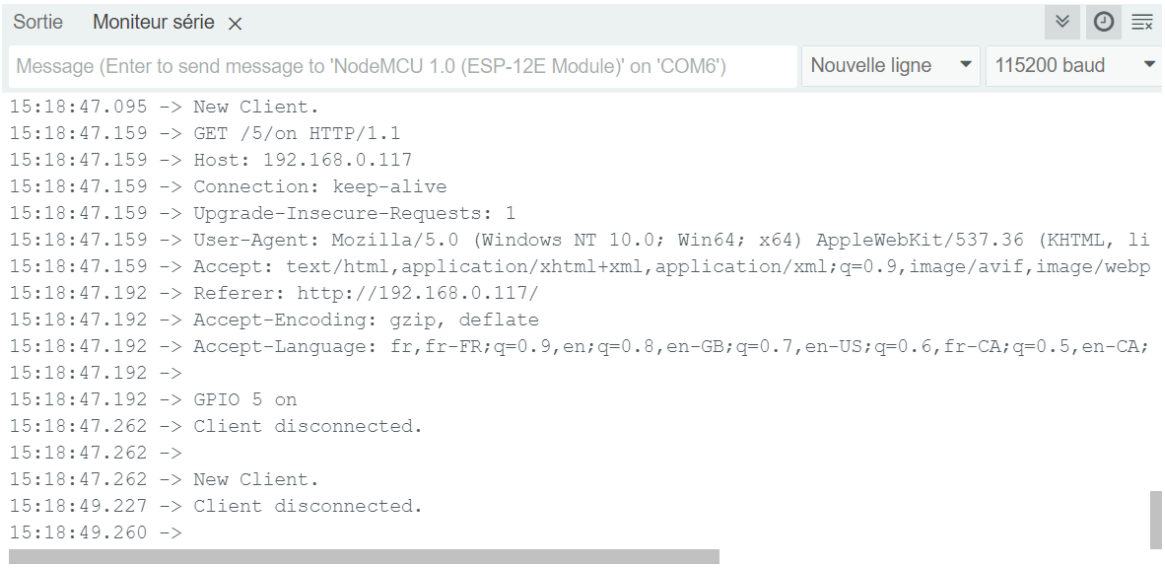
Ouvrez votre navigateur, saisissez l'adresse IP de l'ESP et vous verrez la page suivante. Cette page est envoyée par l'ESP8266 lorsque vous effectuez une demande sur l'ESP le moniteur série Adresse IP.



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

L'état de la LED est également mis à jour sur la page Web.





Testez le bouton GPIO 4 et vérifiez qu'il fonctionne de manière similaire.

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 que vous devez faire est d'inclure la bibliothèque ESP8266WiFi.

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

Comme mentionné précédemment, vous devez insérer votre SSID et votre mot de passe dans les lignes suivantes entre guillemets.

```
// Remplacez par vos identifiants réseau
const char* ssid     = "REMPLACER_PAR_VOTRE_SSID"";
const char* password = ""REMPLACER_PAR_VOTRE_MOT_DE_PASSE";
```

Ensuite, vous définissez votre serveur Web sur le port 80.

```
// Définir le numéro de port du serveur Web sur 80
WiFiServer server(80);
```

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

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

```
String output5State = "off";
String output4State = "off";
```

Vous devez également attribuer un GPIO à chacune de vos sorties. Ici, nous utilisons le GPIO 5 et le GPIO 4. Vous pouvez utiliser tout autre GPIO approprié.

```
const int output5 = 5;
const int output4 = 4;
```

void setup (){}

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

```
Serial.begin(115200);
```

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

```
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
// Définir les sorties sur LOW
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);
```

Les lignes suivantes démarrent la connexion Wi-Fi avec WiFi.begin(ssid, password), attendent une connexion réussie et impriment l'adresse IP ESP dans le moniteur série.

```
// Connect to Wi-Fi network with SSID and password
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
```

void loop ()

Dans la fonction void 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 :

```
WiFiClient client = server.available();   // Écouter les clients entrants
```

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, sauf si vous sachez exactement ce que vous faites.

```
if (client) {                             // Si un nouveau client se connecte,
  Serial.println("Nouveau client.");      // afficher un message sur le port série
  String currentLine = "";                // créer une chaîne pour stocker les données entrantes du client
  currentTime = millis();
  previousTime = currentTime;
  while (client.connected() && currentTime - previousTime <= timeoutTime) { // boucle tant que le client est connecté
    currentTime = millis();
    if (client.available()) {             // si des octets sont à lire du client,
      char c = client.read();             // lire un octet, puis
      Serial.write(c);                    // l'afficher dans le moniteur série
      header += c;
      if (c == '\n') {                    // si l'octet est un caractère de nouvelle ligne
        // si la ligne actuelle est vide, vous avez deux caractères de nouvelle ligne à la suite.
        // c'est la fin de la requête HTTP du client, donc envoyez une réponse :
        if (currentLine.length() == 0) {
          // Les en-têtes HTTP commencent toujours par un code de réponse (par exemple, HTTP/1.1 200 OK)
          // et un type de contenu pour que le client sache ce qui arrive, puis une ligne vide :
          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.

Comme nous l'avons vu précédemment, nous faisons une requête sur différentes URL en fonction du bouton sur lequel on appuie.

```
// allumer et éteindre les GPIO
if (header.indexOf("GET /5/on") >= 0) {
  Serial.println("GPIO 5 allumé");
  output5State = "on";
  digitalWrite(output5, HIGH);
} else if (header.indexOf("GET /5/off") >= 0) {
  Serial.println("GPIO 5 éteint");
  output5State = "off";
  digitalWrite(output5, LOW);
} else if (header.indexOf("GET /4/on") >= 0) {
  Serial.println("GPIO 4 allumé");
  output4State = "on";
  digitalWrite(output4, HIGH);
} else if (header.indexOf("GET /4/off") >= 0) {
  Serial.println("GPIO 4 éteint");
  output4State = "off";
  digitalWrite(output4, LOW);
}
```

Par exemple, si vous avez appuyé sur le bouton GPIO 5 ON, l'URL change pour l'adresse IP ESP suivie de /5/ON, et nous recevons ces informations sur l'en-tête HTTP. Nous pouvons donc vérifier si l'en-tête contient l'expression GET /5/on.

S'il contient, le code imprime un message sur le moniteur série, modifie la variable output5State sur on et allume la LED.

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

La prochaine étape consiste à générer la page Web. L'ESP8266 enverra une réponse à votre navigateur avec du texte HTML pour afficher la page Web.

La page Web est envoyée au client à l'aide de la fonction client.println(). Vous devez saisir comme argument ce que vous souhaitez envoyer au client.

Le premier texte que vous devez toujours envoyer est la ligne suivante qui indique que nous envoyons du HTML.

```
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\">");
```

La suivante est utilisée pour empêcher les requêtes liées au favicon. Vous n'avez pas à vous soucier de cette ligne.

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

Styliser la page Web

Ensuite, nous avons du CSS pour styliser les boutons et l'apparence de la page Web. Ensuite, la ligne suivante rend la page Web réactive dans n'importe quel navigateur Web. Nous choisissons la police Helvetica, définissons le contenu à afficher sous forme de bloc et aligné au centre.

```
// CSS pour styliser les boutons on/off
// N'hésitez pas à modifier les attributs background-color et font-size selon vos préférences
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 40px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
client.println(".button2 {background-color: #77878A;}</style></head>");
```

Nous stylisons nos boutons avec certaines propriétés pour définir la couleur, la taille, la bordure

Ensuite, nous définissons le style d'un deuxième bouton, avec toutes les propriétés du bouton que nous avons défini précédemment, mais avec une couleur différente. Ce sera le style pour le bouton off.

Définition du premier titre de la page Web

Dans la ligne suivante, vous définissez le premier titre de votre page Web, vous pouvez modifier ce texte comme vous le souhaitez.

```
// En-tête de la page Web
client.println("<body><h1>Serveur Web ESP8266</h1>");
```

Affichage des boutons et de l'état correspondant

Ensuite, vous écrivez un paragraphe pour afficher l'état actuel du GPIO 5. Comme vous pouvez le voir, nous utilisons la variable output5State , 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 5
client.println("<p>GPIO 5 - État " + output5State + "</p>");
```

Ensuite, nous affichons le bouton marche ou arrêt, en fonction de l'état actuel du GPIO 5 ici.

```
// Si l'état de output5 est "off", afficher le bouton ON
if (output5State == "off") {
  client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
} else {
  client.println("<p><a href=\"/5/off\"><button class=\"button button2\">OFF</button></a></p>");
}
```

Nous utilisons la même procédure pour GPIO 4.

```
// Afficher l'état actuel et les boutons ON/OFF pour GPIO 4
client.println("<p>GPIO 4 - État " + output4State + "</p>");
// Si l'état de output4 est "off", afficher le bouton ON
if (output4State == "off") {
  client.println("<p><a href=\"/4/on\"><button class=\"button\">ON</button></a></p>");
} else {
  client.println("<p><a href=\"/4/off\"><button class=\"button button2\">OFF</button></a></p>");
}
```

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().

```
// Effacer la variable d'en-tête
header = "";
// Fermer la connexion
client.stop();
```

Allez plus loin

Maintenant que vous savez comment fonctionne le code, vous pouvez le modifier pour ajouter plus de 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. Au lieu de contrôler deux LED, vous pouvez contrôler un relais pour contrôler pratiquement n'importe quel appareil électronique.