la communication sans fil des microcontrôleurs **devient flexible**

l'EEPROM ouvre des perspectives de mise en réseau pour les microcontrôleurs sans fil

Gamal Labib (Egypte)

Lorsque vous connectez un microcontrôleur à un réseau wifi à l'aide de l'ESP8266, vous souhaiteriez une approche plus souple que des identifiants WLAN fixes codés en dur. Dans cet article, je présenterai une solution avec une liste de réseaux AP préférés à choisir de manière interactive. En outre, nous exploiterons la fonction *Wi-Fi Protected Setup* (WPS) prise en charge par de nombreux points d'accès et routeurs.

L'ESP8266 est un module microcontrôleur (MCU) populaire qui prend en charge les communications sans fil et sert à diverses applications dans l'Internet des objets (IdO). Les cartes basées sur ce module ont généralement des identifiants de point d'accès (AP : *Access Point*) codés en dur, tels que le nom du réseau (SSID : *Service Set IDentifier*) et le mot de passe (ou clé), dans leurs croquis Arduino. Dans certains cas, les développeurs spécifient également des paramètres IP (Internet Protocol) fixes. Cependant, ces paramètres codés en dur peuvent poser des problèmes lorsque la topologie du réseau local sans fil (WLAN) change. C'est sérieusement ennuyeux de retourner à l'EDI Arduino ou à un environnement de développement similaire juste pour mettre à jour les programmes des cartes MCU déjà déployées, car les cartes doivent être reprises ou remplacées.

Dans cet article, je présenterai une solution qui offre une certaine souplesse pour connecter une carte microcontrôleur à un réseau local sans fil. Au lieu de coder en dur un seul jeu de paramètres d'identification WLAN dans les programmes du projet, pourquoi ne pas disposer d'un ensemble de paramètres réseaux AP préférés pouvant être choisis de manière interactive ? En établissant un dialogue interactif entre l'utilisateur et la carte microcontrôleur par le biais d'un écran tactile, d'une page web ou d'un moyen similaire, il est alors possible à volonté de spécifier de nouveaux paramètres WLAN pour la carte microcontrôleur. En outre, nous pouvons tirer parti de la fonction Wi-Fi Protected Setup (WPS = configuration wifi protégée) prise en charge par de nombreux points d'accès et routeurs. Le WPS permet à une carte microcontrôleur de rejoindre à volonté le réseau préféré de l'utilisateur. Cependant, une question se pose : devons-nous passer par les étapes de connexion du dialogue interactif ou du WPS à chaque redémarrage de la carte microcontrôleur ? La réponse est non, aussi longtemps que nous conservons les nouveaux paramètres WLAN spécifiés dans une mémoire non volatile, accessible par la carte microcontrôleur. Heureusement, l'ESP8266 dispose d'une mémoire morte programmable et effaçable électriquement (EEPROM) interne qui peut être éditée par quelques lignes de code pour mémoriser et récupérer les paramètres demandés. J'ai utilisé cette fonction pour stocker jusqu'à 10 identifiants WLAN définis par l'une ou l'autre des méthodes susmentionnées. Si l'utilisateur le souhaite, cette approche offre non seulement une flexibilité lors de la mise en réseau, mais permet également à la carte microcontrôleur de se connecter au réseau local sans fil possédant la meilleure couverture parmi les 10 variantes enregistrées. Toutefois, il est important de noter qu'il n'est pas recommandé d'écrire continuellement dans l'EEPROM interne, car la durée de vie d'une EEPROM dépend du nombre de cycles d'écriture qu'elle opère. Pour contourner cette restriction et préserver le rôle désigné du microcontrôleur, j'ai inclus une puce EEPROM externe dans la préparation du projet permettant à l'utilisateur d'obtenir le même résultat avec cette EEPROM externe. Cependant, pour notre exemple de configuration WLAN, nous n'allons pas écrire fréquemment, j'ai donc opté pour l'utilisation de l'EEPROM interne.

Matériel

La liste des composants pour ce projet est assez brève. J'ai utilisé la WeMos D1 Mini, qui est une carte basée sur l'ESP8266 connue pour sa simplicité d'utilisation dans l'EDI Arduino, ainsi que pour son faible encombrement. Un module d'affichage graphique OLED de 0,9 pouce ayant une résolution de 128×64 pixels est utilisé pour afficher les messages d'information et de débogage. Il est connecté au bus I²C du microcontrôleur, ainsi qu'à une puce EEPROM externe de 8 Ko. Pour sélectionner l'un des huit modes de fonctionnement du microcontrôleur (voir **tableau 1**), j'ai utilisé une combinaison de trois commutateurs DIP (*Dual Inline Package*), qui sont connectés à trois des GPIO (*General Purpose Input/Output*) numériques du WeMos (par exemple, D5, D6 et D7). Comme les broches des

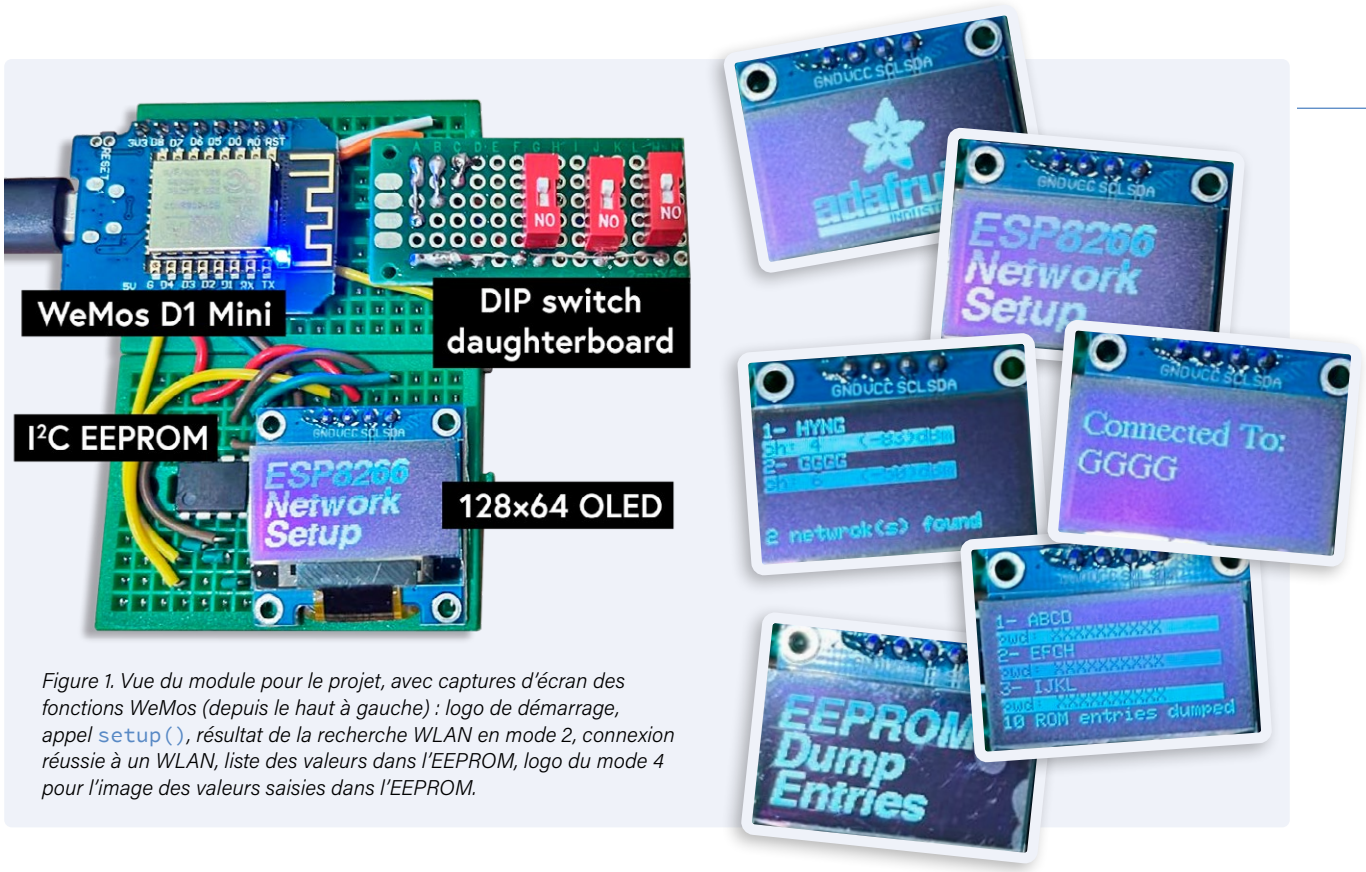


Figure 1. Vue du module pour le projet, avec captures d'écran des fonctions WeMos (depuis le haut à gauche) : logo de démarrage, appel `setup()`, résultat de la recherche WLAN en mode 2, connexion réussie à un WLAN, liste des valeurs dans l'EEPROM, logo du mode 4 pour l'image des valeurs saisies dans l'EEPROM.

Tableau 1. Configuration des commutateurs DIP des modes WeMos appropriés.

Mode	DIP 1	DIP 2	DIP 3	Action du WeMos
1	0	0	0	WLAN sélectionné par défaut
2	0	0	1	Analyse des WLAN disponibles
3	0	1	0	Effacement de l'EEPROM interne
4	0	1	1	Image de l'EEPROM interne
5	1	0	0	Initialisation de l'EEPROM interne
6	1	0	1	Vérification de l'EEPROM externe
7	1	1	0	WPS
8	1	1	1	Setup interactif du WLAN

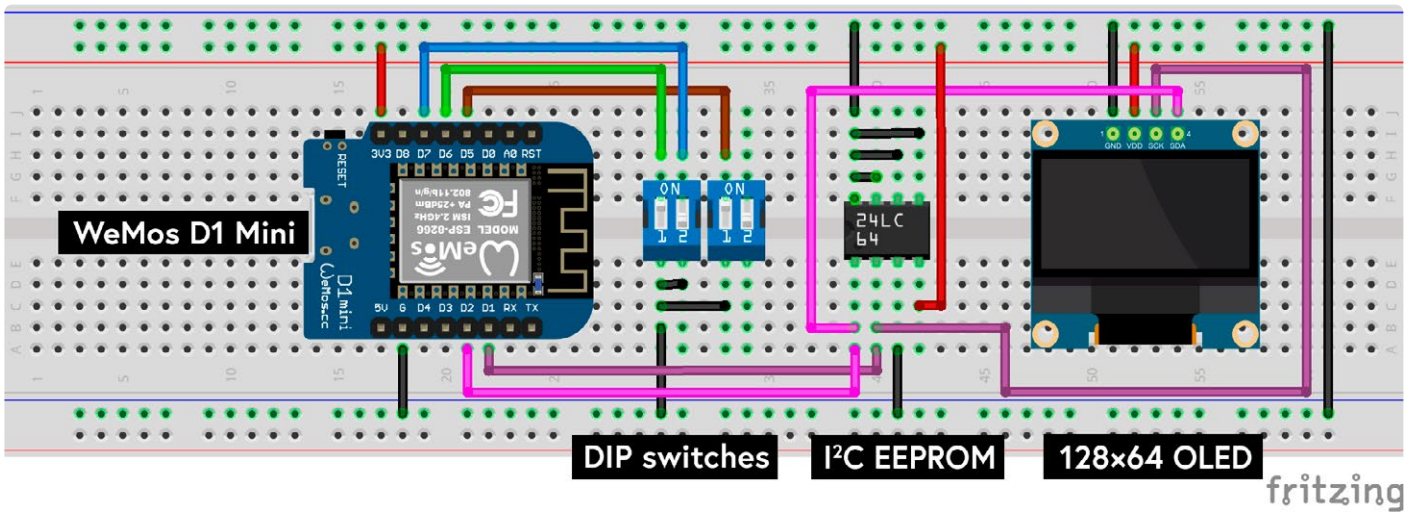


Figure 2. Câblage du projet à l'aide de Fritzing et les outils pour platine d'expérimentation intégrant un quatrième commutateur DIP supplémentaire pour des options de réseau étendues.

interrupteurs DIP sont minuscules, ils n'entrent pas dans la platine d'expérimentation ; j'ai donc construit une carte fille pour les accueillir. La **figure 1** montre l'implémentation du projet sur la platine d'expérimentation ainsi que les captures d'écran du WeMos en action. Le schéma de câblage des composants est présenté à la **figure 2**.

Logiciel

Le projet met l'accent sur les pratiques éprouvées de développement logiciel, telles qu'un code bien structuré et modulaire, permettant de le réutiliser. Dans ce but, j'ai divisé le programme de 800 lignes en six fichiers distincts, chacun gérant un élément matériel ou une fonctionnalité spécifique (voir le **tableau 2**). Cette approche rend le code plus facile à gérer et aide l'utilisateur à comprendre le code assez complexe. Le fichier du programme principal se concentre sur les modes de mise en réseau du WeMos dans la fonction `setup()`, tout en laissant la fonction `loop()` libre pour le code d'application du WeMos. Pendant la phase de développement, j'ai utilisé `loop()` pour gérer le clignotement de la LED présente sur le module et m'assurer que le WeMos fonctionne correctement.

Utilisation de l'EEPROM pour le réseau WLAN

Le module ESP8266 dispose de 4 Mo de mémoire flash, dont 4 Ko sont réservés à l'émulation d'une EEPROM habituellement disponible sur un module Arduino. Par défaut, notre croquis mémorise les dernières informations d'identification de l'AP dans l'EEPROM interne. Avec une mémoire interne (ou externe) non volatile, notre programme peut rappeler les informations d'identification de l'AP à chaque fois qu'il démarre à la suite de la mise sous tension. L'EEPROM est également accessible au moyen d'appels de fonction de la bibliothèque EEPROM de l'Arduino, et je reprendrai cette approche pour sauvegarder les données du réseau qui doivent être non volatiles, lorsque le module est hors tension. Le concept principal consiste à maintenir la structure proposée dans le **tableau 3** pour conserver les informations d'identification pour autant de AP WLAN que l'EEPROM interne ou externe peut contenir. Chaque donnée successive dans la structure contient le SSID de l'AP et son mot de passe dans 32 et respectivement 64 octets de mémoire. Cela fait 96 octets au total par point d'accès. Chaque fois que l'utilisateur ajoute un nouveau WLAN à la liste des réseaux préférés, le code ajoute à cette structure les informations d'identification du SSID et du mot de passe du nouveau réseau. L'utilisateur doit effacer l'EEPROM lors de la première exécution du programme, en utilisant le mode 3, marquant ainsi la fin de la liste ne contenant que des entrées vierges. L'utilisateur aura le choix entre appliquer les informations d'identification WLAN codées en dur, en utilisant le mode 1, ou vérifier la connectivité en utilisant les informations d'identification de l'EEPROM, ce qu'on obtient en utilisant le mode 8. Pour ce projet, ce mécanisme est limité à un maximum de 10 WLAN différents, par la constante `MEMCNT` déclarée dans le code, tout en effaçant le contenu de la structure et l'initialisant avec une liste prédéfinie d'informations d'identification AP choisies par l'utilisateur. Si nécessaire, il peut augmenter ou diminuer la taille de `MEMCNT`. Les fonctions de base relatives à l'accès à l'EEPROM peuvent être consultées à l'adresse [2]. Je vais maintenant détailler les modes de fonctionnement du WeMos, chacun d'entre eux nécessitant l'emploi des commutateurs DIP (conformément au **tableau 1**), puis le redémarrage du WeMos pour appliquer le changement.

Tableau 2. Répartition des fichiers du code Arduino et leurs fonctions.

Fichier .ino file	Description	Nombre de lignes
wemos-d1-mini-network	Variables globales, setup du mode réseau	240
wlan-utils	Paramètres des connexions WLAN : WPS setup, recherche/listing WLAN, analyse connectivité	110
tft-utils	Paramètres de l'affichage TFT	130
internal-eeprom-utils	Initialisation de l'EEPROM avec cinq identifiants WLAN prédéfinis, réinitialisation de l'EEPROM, lecture/écriture des valeurs	180
external-eeprom-utils	Vérification de la connectivité et de la taille de l'EEPROM externe	40
ap-web-setup	Setup interactif du WLAN : lancement du serveur web, affichage des SSID WLAN détectés, enregistrement du WLAN sélectionné dans l'EEPROM	100

Tableau 3. tableau de gestion de l'EEPROM pour des paramètres AP multiples.

Taille des données (octets)	Description	Description
2	nombre d'AP	entier
32	AP1 SSID	chaîne de caractères
64	AP1 mot de passe	chaîne de caractères
.	.	.
.	.	.
.	.	.
32	APn SSID	chaîne de caractères
64	APn mot de passe	chaîne de caractères

Mode 1 : paramètres WLAN par défaut (codés en dur)

Un microcontrôleur a besoin d'informations d'identification WLAN, composées d'un SSID et d'un mot de passe pour accéder au point d'accès souhaité. Les informations d'identification WLAN peuvent être soit codées en dur dans le programme flashé, soit transmises interactivement au microcontrôleur si l'utilisateur est équipé d'un clavier et d'un écran, ou du moniteur de l'EDI relié via le port série. Pour ce projet, j'utilise la méthode conventionnelle qui consiste à fixer les informations d'identification de l'AP directement dans le programme.

```
14:34:00.668 -> scan available wlans
14:34:05.788 -> Disconnecting previously connected WiFi
14:34:08.148 -> scan completed
14:34:08.148 -> 1 Networks found
14:34:08.148 -> 1: GGGG (6)  (-63)
14:34:08.268 ->
```

Figure 3. Messages du moniteur série de l'EDI Arduino – mode 2, balayage du réseau sans fil.

```
14:35:43.671 -> check external eeprom
14:35:48.751 -> Disconnecting previously connected WiFi
14:35:48.871 -> check external eeprom
14:35:48.871 -> External eeprom isConnected with Status:
14:35:48.871 ->
14:35:48.871 -> TEST: determine size of external eeprom
14:35:48.871 -> 80      FF
14:35:48.871 -> 100     0
14:35:48.871 -> 200     0
14:35:48.871 -> 400     0
14:35:48.871 -> 800     FF
14:35:48.911 -> 1000    AA
14:35:48.911 -> 2000    AA
14:35:48.911 -> external eeprom size: 8192 Bytes
```

Figure 4. Messages du moniteur série - mode 6, vérification de l'intégrité de l'EEPROM externe.

Mode 2 : balayage WLAN

Dans ce mode, le WeMos se déconnecte premièrement de tout WLAN connecté afin de pouvoir rechercher d'autres réseaux présents. Les réseaux locaux sans fil détectés sont répertoriés en fonction de leur SSID et de l'intensité du signal en décibels (dB). Ce mode donne à l'utilisateur un aperçu des possibilités de la mise en réseau de la carte WeMos et du mode à choisir pour accéder à un WLAN. La **figure 3** présente le résultat du démarrage du WeMos dans ce mode, où un seul WLAN a été détecté et connecté par la suite, comme le montre la **figure 1d**.

Mode 3 : effacement de l'EEPROM interne

Ce mode permet au WeMos d'effacer son EEPROM interne afin de préparer la création d'une nouvelle liste de WLANs choisis. Dans ce mode tous les octets de l'EEPROM sont effacés et obtiennent la valeur « 0×00 «.

Mode 4 : image de l'EEPROM interne

Ce mode permet au WeMos d'extraire la structure des informations d'identification du WLAN, comme expliqué dans le **tableau 3**, de l'EEPROM interne. Les **figures 1e** et **f** présentent les captures d'écran des valeurs actuellement mémorisées dans l'EEPROM. Dans ce mode, une série de SSID et de mots de passe formatés sont affichés pour les informations d'identification des points d'accès mémorisés.

Mode 5 : initialisation de l'EEPROM interne

Ce mode permet à l'utilisateur de déposer dans l'EEPROM interne les informations d'identification de 10 WLANs choisis. Cette action est une extension de la manière traditionnelle de mémorisation des informations d'identification codées en dur pour un seul WLAN. Lors du redémarrage en mode 8, le WeMos consulte la liste des informations d'identification pour les connexions WLAN. Si les tentatives de connexion ne permettent pas d'accéder à l'un des WLANs à choix, l'utilisateur devra alors recourir au mode interactif ou WPS.

Mode 6 : vérification de l'état de l'EEPROM externe

Dans ce mode, le WeMos vérifie l'état de l'EEPROM externe, ici un 24LC64 de 8 KB. La **figure 4** montre le résultat de cette vérification, sur le moniteur série. Je n'ai pas utilisé ce composant externe pour la suite du projet puisque l'utilisation de l'EEPROM interne est favorable. L'utilisateur peut se référer au code de l'EEPROM I²C [3] et à la bibliothèque [4] s'il désire l'intégrer dans son projet.

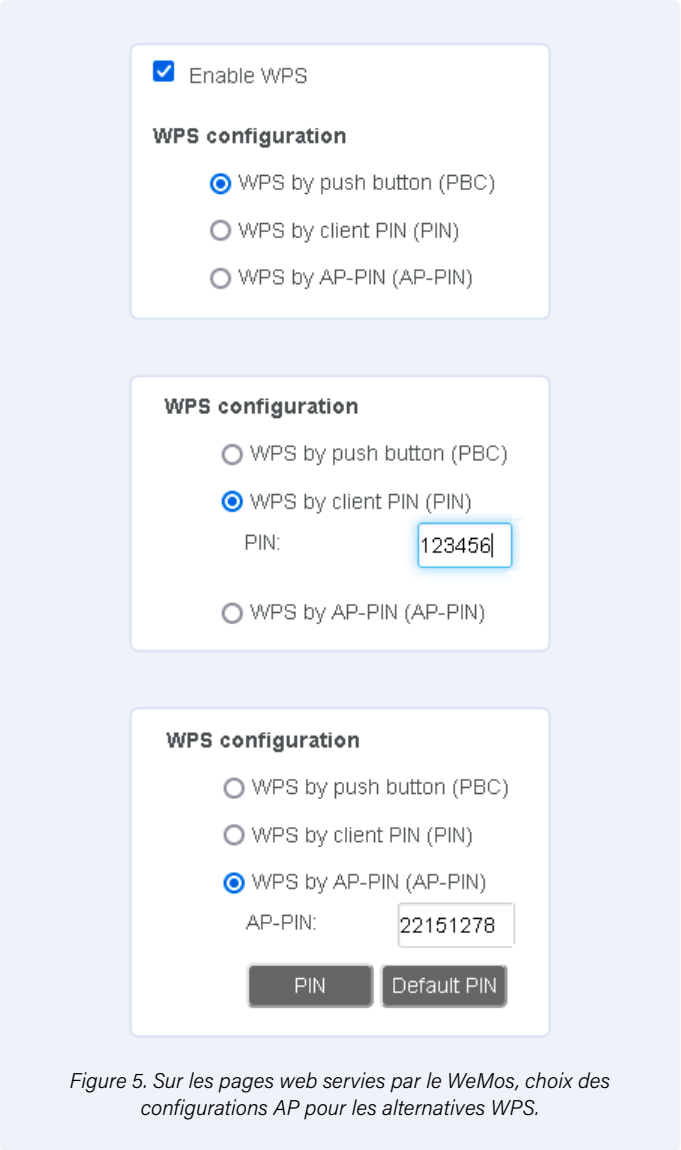


Figure 5. Sur les pages web servies par le WeMos, choix des configurations AP pour les alternatives WPS.

Figure 6. Bouton WPS et indicateur d'un point d'accès typique.

```
14:37:03.552 -> Disconnecting previously connected WiFi
14:37:03.632 -> Try wps if necessary
14:37:03.632 ->
14:37:03.632 -> Could not connect to WiFi ... state= '7'
14:37:03.672 -> Please press WPS button on your router
14:37:03.672 -> WPS config start
```

Figure 7.Messages du moniteur série - mode 7, configuration WPS.

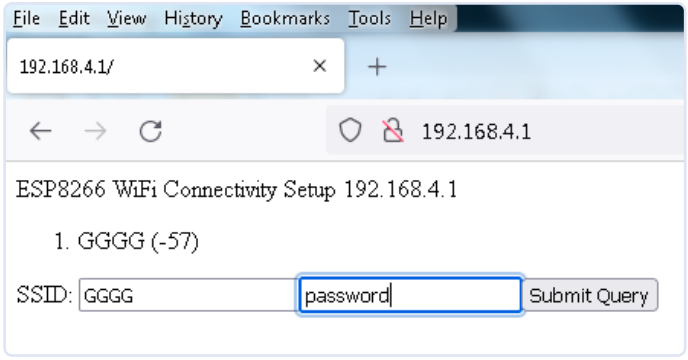


Figure 8. Mode 8, sélection interactive du WLAN.

Mode 7 :connexion directe au WLAN (WPS)

Le WPS est une fonction supportée par les AP modernes qui simplifie le processus de connexion des appareils sans fil à un WLAN. Plutôt que de fournir au microcontrôleur les informations d'identification de l'AP, par exemple le SSID et le mot de passe, le microcontrôleur peut fournir le code de sécurité de l'AP ou son propre code prédéfini (voir la **figure 5**). Une alternative consiste à cliquer sur un bouton-poussoir WPS sur le point d'accès pour établir une connexion directe avec le microcontrôleur requérant (voir **figure 6**). La **figure 7** présente le résultat de l'établissement d'une connexion à l'aide du WPS**.**

Dans ce projet, je limite la connexion WPS à l'alternative du bouton poussoir de l'AP car cela permet un algorithme plus simple et plus générique, s'adaptant à n'importe quel AP à portée de main, comme suit :

```
bool wpsSuccess = WiFi.beginWPSConfig();
if (wpsSuccess) {
    String newSSID = WiFi.SSID();
    if (newSSID.length() == 0)  { wpsSuccess = false; }
}
```

La carte WeMos doit être à proximité de l'AP choisi pour enregistrer les paramètres de l'AP dans l'EEPROM du module. Pour reconnecter le module au même AP il n'est pas nécessaire de cliquer à nouveau sur le bouton WPS, car nous permettons au module de récupérer les paramètres AP mémorisés, pour les réutiliser, comme ceci :

```
WiFi.mode(WIFI_STA);
WiFi.begin(WiFi.SSID().c_str(),WiFi.psk().c_str());
```

Notez que le module ESP8266 qui forme le cœur de la carte WeMos mémorise automatiquement dans l'EEPROM interne les dernières informations d'identification de réseau détectées avec succès, de sorte que nous devons placer notre structure de gestion, décrite dans le **tableau 3**, ailleurs dans l'EEPROM afin d'éviter tout conflit avec les autres mécanismes du microcontrôleur.

Mode 8 : configuration interactive du WLAN

Dans ce mode, le WeMos vérifie d'abord la validité des informations d'identification contenues dans l'EEPROM. Si le microcontrôleur réussit à se connecter à un point d'accès, le processus se termine et la fonction `loop()` prend le relais. Dans le cas contraire, le microcontrôleur passe en mode interactif, dans lequel elle analyse les WLANs à proximité et lance un serveur web avec l'adresse IP 192.168.4.1 pour afficher les SSID des points d'accès détectés. L'utilisateur doit régler l'adaptateur WLAN de son ordinateur sur les paramètres IP fixes du réseau privé WeMos, car le WeMos ne fournit pas de service DHCP. Une page web semblable à celle de la **figure 8** affiche les résultats de l'analyse et permet à l'utilisateur de choisir le réseau qu'il désire. L'utilisateur clique ensuite sur le bouton *Submit Query* pour enregistrer les informations d'identification du WLAN dans l'EEPROM et établir une connexion avec le WLAN sélectionné.

L'utilisation d'EEPROMs représente un atout sérieux lorsqu'il s'agit de mettre des MCUs en réseau sans fil. Dans cet article, plutôt que de coder en dur les informations d'identification du WLAN dans le code du MCU, l'accessibilité à une EEPROM externe ou interne du MCU est utilisée pour conserver et mettre à jour de manière dynamique les informations d'identification de plusieurs points d'accès afin d'établir des connexions. Le code source accompagnant cet article est modulaire, bien structuré, facile à comprendre, et propose une version réutilisable pour gérer les EEPROMs, les WLANs, et pour communiquer avec le MCU à travers le web.

VF : Jean-Philippe Nicolet — 230268-04

À propos de l'auteur

Gamal Labib est un passionné depuis deux décennies de systèmes embarqués et est actuellement mentor (chez codementor.io). Il est titulaire d'un Master en génie mécanique (Meng) et d'un diplôme de doctorat (PhD) en informatique. En plus d'écrire pour des magazines techniques, il est professeur agrégé invité dans les universités égyptiennes et consultant informatique certifié.

Questions ou commentaires ?

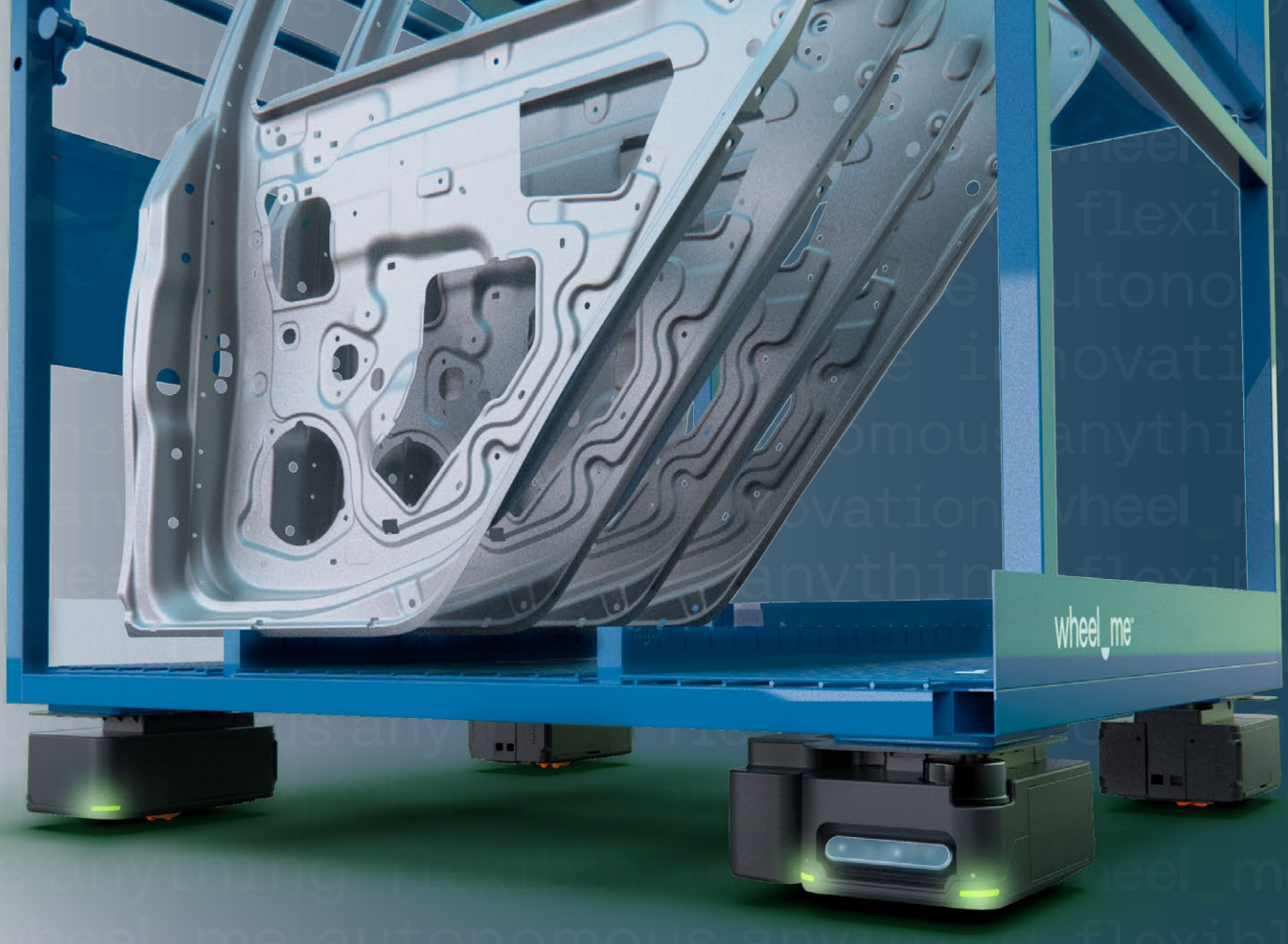
Envoyez un courriel à l'auteur (drgamallabib@yahoo.co.uk) ou contactez Elektor (redaction@elektor.fr).

Produits

- **WeMos D1 mini Pro – Module WiFi basé sur ESP8266**
  https://elektor.fr/19185
- **Écran OLED 0.96 pouce (bleu, I2C, 4-Pin)**
  https://elektor.fr/18747
- **Hans Henrik Skovgaard, *Home Appliance Hack-and-IoT Guidebook* (+ carte ESP8266 gratuite), Elektor 2022**
  https://elektor.fr/20370

LIENS

[1] Téléchargement du logiciel : https://elektormagazine.fr/230268-04
[2] Utilisation de l'EEPROM avec l'ESP8266 : https://aranacorp.com/en/using-the-eeprom-with-the-esp8266
[3] Arduino avec une EEPROM I²C : https://playground.arduino.cc/Code/I2CEEPROM
[4] Bibliothèque pour l'EEPROM I²C : https://github.com/RobTillaart/I2C_EEPROM



Genius by wheel_me

Bénéficiez d'une efficacité maximale dans l'intralogistique avec le Genius de wheel.me :
Quatre roues de robot adaptables permettant le transport autonome de marchandises à un prix compétitif.

- Navigation autonome
- Mouvements omnidirectionnels
- Capteurs haute technologie
- Contrôle à distance par l'application wheel.me
- Charge utile ajustable
- Recharge intermédiaire sans interruption des opérations

wheel_me

pour plus de renseignements,
consultez www.wheel.me