Le pot qui pense - ESP8266 connexion et développement

1. Le microcontrôleur ESP8266 Spécifications techniques

- Architecture : Xtensa lx106
- Fréquence du processeur : 80MHz overclockable à 160MHz
- Total de RAM disponible : 96Ko (partie de celle-ci réservée au système)
- BootROM : 64Ko
- FlashROM externe : code et données, via SPI Flash - 4 Mo.
- GPIO : 16 + 1 (GPIO est multiplexées avec d'autres fonctions, y compris externes FlashROM, UART, sommeil profond réveil, etc.).
- UART : Un RX/TX UART (aucun handshaking matériel), un UART TX uniquement.
- SPI : 2 SPI interfaces (celui utilisé pour FlashROM).
- I2C : Aucun I2C externes native (bitbang application disponible sur des broches).
- I2S : 1.
- Programmation : à l'aide de BootROM bootloader de UART. En raison de la FlashROM externe et toujours disponible BootROM bootloader, ESP8266 n'est pas brickable.

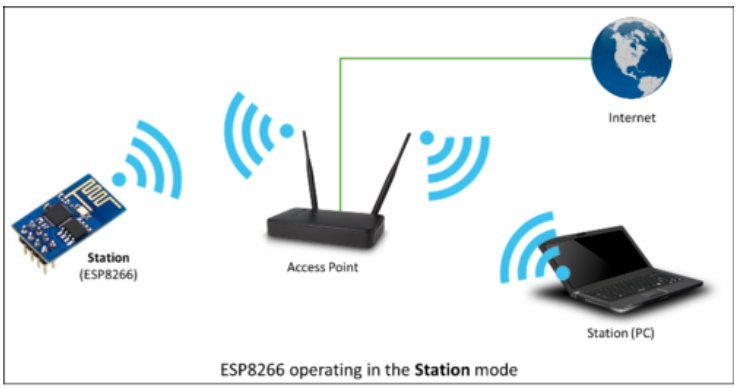
2. Les différents modes WiFi

Les appareils qui se connectent au réseau WiFi sont appelés stations (STA). La connexion au Wi-Fi est fournie par un point d'accès (AP - typiquement c'est une Box-Wifi), qui joue le rôle de concentrateur pour une ou plusieurs stations. Un point d'accès intègre généralement un routeur pour fournir un accès Internet au réseau Wi-Fi. Chaque point d'accès est identifié par un SSID (**Service Set IDentifier**), qui correspond au nom du réseau que vous sélectionnez lors de la connexion d'un appareil (**station**) au réseau WiFi.

Chaque module ESP8266 peut fonctionner comme une station (**STA**) pour le connecter au réseau WiFi. Il peut également fonctionner en tant que point d'accès logiciel (**soft-AP**) pour établir son propre réseau WiFi. Par conséquent, nous pouvons connecter plusieurs stations à de tels modules. Troisièmement, l'ESP8266 est également capable de fonctionner simultanément en mode station et en tant que point d'accès. Cela offre la possibilité de construire, par exemple, des réseaux maillés.

2.1. Mode Station

Le mode Station (**STA**) permet de connecter l'ESP8266 à un point d'accès (une Box-wifi).



Station

2.2. Mode Point d'Accès

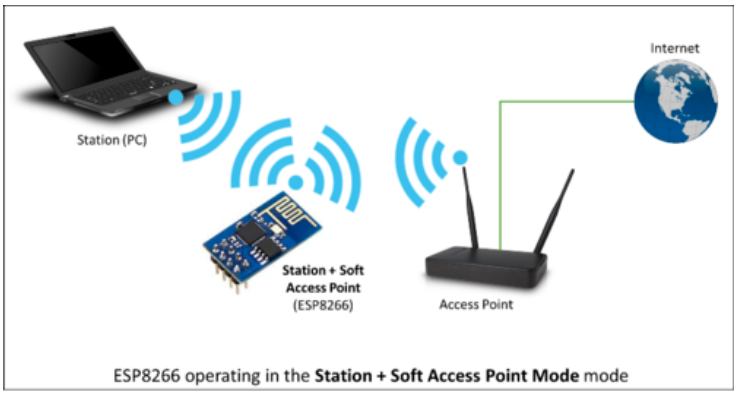
Un point d'accès (**AP**) est un appareil qui fournit un accès au réseau Wi-Fi à d'autres appareils (stations) et les connecte ensuite un réseau filaire. L'ESP8266 peut fournir des fonctionnalités similaires, à la différence qu'il ne possède pas d'interface avec un réseau filaire. Ce mode de fonctionnement est appelé point d'accès logiciel (**Soft-AP**). Le nombre maximal de stations connectées au soft-AP est de cinq.

Le mode soft-AP constitue une étape intermédiaire avant de connecter l'ESP à un réseau WiFi en mode station. A ce stade le SSID et le mot de passe du réseau ne sont pas encore paramétrés. Le module démarre d'abord en mode soft-AP, afin que nous puissions nous y connecter à l'aide d'un ordinateur portable ou d'un téléphone portable. Ensuite, nous pouvons fournir des informations d'identification au réseau cible. Une fois cela fait, ESP peut passer en mode station et se connecter à la Box-wifi locale.

AccesPoint

2.3. Mode Station + Point d'Accès

Une autre application pratique du mode soft-AP consiste à configurer des réseaux maillés . ESP peut fonctionner à la fois en mode soft-AP et en mode station afin de pouvoir agir en tant que nœud d'un réseau maillé.



Station+AP

3. REPL (Read Evaluate Print Loop)

REPL est le nom donné à l'invite interactive MicroPython à laquelle vous accédez lorsque vous vous connectez sur l'ESP8266. Utiliser le REPL est de loin le moyen le plus simple de tester votre code et d'exécuter des commandes.

Il existe deux manières d'accéder au REPL: soit via une connexion filaire via le port série USB->UART, soit via le WiFi.

3.1. REPL sur le port série

Le REPL est toujours disponible sur le périphérique série UART0, qui est connecté aux broches GPIO1 pour TX et GPIO3 pour RX. Le débit en bauds du REPL est de 115 200. Si votre carte possède un convertisseur série-USB, vous pouvez accéder au REPL directement à partir de votre PC.

Pour accéder à l'invite via USB-série, vous devez utiliser un programme d'émulateur de terminal. Sous Windows, **TeraTerm** est un bon choix. Sous Linux, vous pouvez utiliser le programme **Screen**.

Par exemple, sous Linux, vous pouvez taper :

```
$ screen /dev/ttyUSB0 115200
```

Une fois la connexion établie via le port série, vous pouvez vérifier son fonctionnement en appuyant plusieurs fois sur Entrée. L'invite Python REPL, indiquée par **>>>**, s'affiche.

3.2. WebREPL - une invite via WiFi

WebREPL vous permet d'utiliser l'invite Python via WiFi, en vous connectant via un navigateur. Les dernières versions de Firefox et de Chrome sont prises en charge.

Pour votre commodité, le client WebREPL est hébergé à l' adresse http://micropython.org/webrepl. Vous pouvez également l'installer localement à partir du référentiel GitHub https://github.com/micropython/webrepl.

> **Note :** Avant de vous connecter à WebREPL, vous devez définir un mot de passe et l'activer via une connexion série USB normale. Les versions initiales de MicroPython for ESP8266 étaient livrées avec WebREPL activé automatiquement au démarrage et avec la possibilité de définir un mot de passe via WiFi lors de la première connexion, mais WebREPL devenant de plus en plus connu et populaire, la configuration initiale est passée à une connexion filaire pour une sécurité améliorée :
>
> ```
> >>> import webrepl_setup
> ```
>
> Suivez les instructions à l'écran et les invites. Pour que toute modification soit active, vous devez redémarrer votre appareil. Concrètement il va vous demander d'activer WebREPL au démarrage (**E**) puis de saisir un mot de passe afin de sécuriser la connexion (**crepp**). Ensuite il propose de redémarrer l'ESP8266.

Pour utiliser WebREPL, connectez votre ordinateur au point d'accès de l'ESP8266 : MicroPython-xxxxxx, (xxxxxx correspond aux derniers chiffres de l'adresse MAC de ESP8266). Ensuite, renseigner le mot de passe du réseau : **micropythoN**. (Attention : N majuscule pour la dernière lettre).

Une fois que vous êtes sur le réseau de l'ESP8266, lancer le fichier **webrepl.html** (*clic droit > ouvrir avec Firefox*) puis cliquez sur le bouton **Connecter** (si vous vous connectez via un routeur, vous devrez peut-être modifier l'adresse IP. Par défaut, l'adresse IP est correcte lorsque l'ESP8266 est connecté en mode Point d'Accès). Lorsque la connexion s'établie, vous devriez avoir une invite de mot de passe.

Une fois que vous avez saisi le mot de passe (**crepp** - renseigné lors de l'étape de configuration), appuyez à nouveau sur Entrée. Un message d'invitation ressemblant à **>>>**. Vous pouvez maintenant commencer à taper des commandes Python !

> **Important** : Le sous-système WiFi est géré par des tâches en arrière-plan qui doivent être exécutées périodiquement. Toute fonction ou tâche nécessitant plus de 15ms (millisecondes) peut entraîner le blocage du sous-système WiFi. Pour éviter ces pannes potentielles, il est conseillé de suspendre le sous-système WiFi avec wifi.suspend() avant l'exécution de tâches ou de fonctions dépassant ce seuil de 15ms.

3.3. Utiliser le REPL

Une fois que vous avez une invite, vous pouvez commencer à expérimenter ! Tout ce que vous tapez à l'invite sera exécuté une fois que vous aurez appuyé sur la touche Entrée. MicroPython exécutera le code que vous avez entré et affichera le résultat (s'il en existe un). S'il y a une erreur dans le texte que vous avez entré, un message d'erreur s'affiche.

Essayez de taper ce qui suit à l'invite:

```
>>> print('Bonjour ESP8266!')
Bonjour ESP8266!
```

Si vous connaissez déjà un peu de python, vous pouvez maintenant essayer quelques commandes de base ici. Par exemple:

```
>>> 1 + 2
3
>>> 1/2
0.5
>>> 12 ** 34
```

```
492223524295202670403711324312200806 4
```

Si votre carte a une LED connectée à GPIO2 (Oui les modules ESP-12 le font), vous pouvez l'allumer et l'éteindre à l'aide du code suivant:

```
>>> import machine
>>> led = machine.Pin(2, machine.Pin.OUT) # GPIO2
>>> led.on()
>>> led.off()
```

Notez que la méthode d'une broche peut éteindre le voyant et off l'allumer (ou vice-versa), selon la façon dont le voyant est câblé sur la carte. Pour résoudre ce problème, la classe machine.Signal est fournie.

3.3.1. ligne

Vous pouvez modifier la ligne actuelle que vous entrez en utilisant les touches fléchées gauche et droite pour déplacer le curseur, ainsi que les touches de suppression et de retour arrière. De plus, appuyer sur Accueil ou sur ctrl-A déplace le curseur au début de la ligne et appuyer sur Fin ou sur ctrl-E pour aller à la fin de la ligne.

3.3.2. Historique de saisie

Le REPL mémorise un certain nombre de lignes de texte que vous avez saisies précédemment (jusqu'à 8 sur l'ESP8266). Pour rappeler les lignes précédentes, utilisez les touches fléchées haut et bas.

3.3.3. Complétion par onglet

Appuyez sur la touche Tab pour compléter automatiquement le mot que vous entrez. Cela peut être très utile pour connaître les fonctions et méthodes d'un module ou d'un objet. Essayez-le en tapant «ma» puis en appuyant sur Tab. Il doit terminer par «machine» (en supposant que vous avez importé une machine dans l'exemple ci-dessus). Tapez ensuite "." Et appuyez à nouveau sur Tab pour voir une liste de toutes les fonctions du module de la machine.

3.3.4. Continuation de ligne et mise en retrait automatique

Certaines choses que vous tapez auront besoin de «continuer», c'est-à-dire qu'il faudra plus de lignes de texte pour créer une instruction Python appropriée. Dans ce cas, l'invite devient ... et le curseur s'indente automatiquement du montant correct afin que vous puissiez commencer à taper immédiatement la ligne suivante. Essayez ceci en définissant la fonction suivante:

```
>>> def toggle(p):
... p.value(not p.value())
...
...
...
>>>
```

Dans ce qui précède, vous devez appuyer sur la touche Entrée trois fois de suite pour terminer l'instruction composée (c'est-à-dire les trois lignes ne comportant que des points). L'autre façon de terminer une instruction composée consiste à appuyer sur la touche retour arrière pour atteindre le début de la ligne, puis à appuyer sur la touche Entrée. (Si vous avez commis une erreur et que vous souhaitez sortir du mode continuation, appuyez sur ctrl-C; toutes les lignes seront ignorées.)

La fonction que vous venez de définir vous permet de basculer la led. L'objet toggle que vous avez créé précédemment doit toujours exister (recréez-le s'il ne l'est pas) et vous pouvez basculer le voyant à l'aide de:

```
>>> toggle(led)
```

Basculons maintenant la LED dans une boucle (si vous n'avez pas de LED, vous pouvez simplement imprimer du texte au lieu d'appeler toggle, pour voir l'effet):

```
>>> import time
>>> while True:
```

```
... toggle(led)
... time.sleep_ms(500)
...
...
...
>>>
```

Cela fera basculer la LED à 1Hz (une demi-seconde allumée, une demi-seconde éteinte). Pour arrêter le basculement, appuyez sur ctrl-C, ce qui déclenchera une exception KeyboardInterrupt et s'échappera de la boucle.

Le module de temps fournit des fonctions utiles pour retarder et chronométrer. Utilisez l'onglet de complétion pour découvrir ce qu'ils sont et jouez avec eux!

3.3.5. Mode coller

Appuyez sur ctrl-E pour entrer dans un mode de collage spécial. Cela vous permet de copier et coller un bloc de texte dans le REPL. Si vous appuyez sur **ctrl-E**, l'invite du mode coller apparaît:

```
mode coller ;  Ctrl - C pour annuler , Ctrl - D pour terminer
===
```

Vous pouvez ensuite coller votre texte. Notez qu'aucune des touches ou commandes spéciales ne fonctionne en mode coller (par exemple, Tabulation ou retour arrière), elles sont simplement acceptées telles quelles. Appuyez sur ctrl-D pour terminer la saisie du texte et l'exécuter.

3.3.6. Autres commandes de contrôle

Il y a quatre autres commandes de contrôle:

```
Ctrl-A sur une ligne vierge passera en mode REPL brut. Cela ressemble à un mode de collage permanent, sauf que les caractè
Ctrl-B sur un blanc comme passe en mode REPL normal.
Ctrl-C annule toute entrée ou interrompt le code en cours d'exécution.
Ctrl-D sur une ligne vierge effectuera une ré-initialisation logicielle.
```

Notez que ctrl-A et ctrl-D ne fonctionnent pas avec WebREPL.

4. Le processus de démarrage de l'ESP8266

Au démarrage, le firmware MicroPython de l'EPS8266 exécute **_boot.py** le script de modules internes de réveil. Il monte le système de fichiers dans la FlashROM, ou s'il n'est pas disponible, effectue la première fois le programme d'installation du module et crée le système de fichiers. Cette partie de la procédure de démarrage est considéré comme fixe et non disponibles à la personnalisation pour les utilisateurs finaux (*même si vous compiler depuis les sources, veuillez vous abstenir de changements; la personnalisation du processus de démarrage précoce est disponible uniquement pour les utilisateurs avancés et les développeurs, capable de diagnostiquer eux-mêmes des problèmes découlant de la modification du processus standard*).

Une fois que le système de fichiers est monté, **boot.py** est exécuté. La version standard de ce fichier est créée au cours de la première exécution et dispose de commandes pour démarrer le démon **WebREPL** (désactivé par défaut, configurable avec le module webrepl_setup), etc.. Ce fichier est personnalisable par l'utilisateur final (par exemple, vous pouvez régler certains paramètres ou ajouter d'autres services qui doivent être exécutés pour une mise en service du module). Mais n'oubliez pas que les modifications incorrectes à boot.py peuvent provoquer un reboot en boucle, exigeant de reflasher le module à partir de zéro.

La dernière étape de la procédure de démarrage minimal, **main.py** est exécuté à partir du système de fichiers, si il existe. Ce fichier est un outil pour lancer une application utilisateur à chaque démarrage (au lieu d'utiliser REPL). Pour des petites applications de test, vous pouvez les nommer directement **main.py**, mais au lieu de cela, il est recommandé de garder vos applications dans des fichiers séparés (par exemple : **my_app.py**) et de les appeler dans **main.py** comme ceci :

```
import my_app
my_app.main()
```

Cela vous permet de garder la structure de votre application claire, ainsi que de permettre d'installer plusieurs applications sur une carte et de basculer de l'une à l'autre.

5. Debogage et rechargement des programmes microPython via WebRepl

Utiliser 2 fichiers : **main.py** et **potsol.py**

- Au départ laisser vide main.py et écrire le programme dans potsol.py
- Créer dans potsol.py une fonction main() qui pourra être lancer par :

```
>>> import potsol
>>> potsol.main()
```

Pour recharger le nouveau programme sans se déconnecter de la carte

- Ctrl+C pour sortir de la boucle (si il y a une boucle d'affichage)
- Uploader le nouveau fichier potsol.py sur l'ESP8266 avec le bouton **"Send to Device"**
- Puis pour décharger l'ancien module (ou programme), réimporter le nouveau et le lancer :

```
>>> import sys
>>> del sys.modules['potsol'] # décharge du module
>>> import potsol   # ré-importation du nouveau module
>>> potsol.main()   # lancement du nouveau programme
```

Lorsque le programme est fonctionnel, activer son lancement au démarrage de l'ESP8266 dans main.py :

```
import potsol
potsol.main()
```

5.1. Commandes courantes de gestion de fichiers

```
>>> import os
>>> os.listdir() # liste les fichiers présents sur l'ESP8266
['boot.py', 'webrepl_cfg.py', 'ads1x15.py', 'main.py', 'potsol.py']
>>> os.rename("potsol.py", "potquipense.py") # renomme le fichier potsol.py en potquipense.py
>>> os.remove("potquipense.py") # supprime le fichier potquipense.py
```

6. Document

Crepp 2018 - yvoz.lg@gmail.com
Document source composé en Markdown + Pandoc