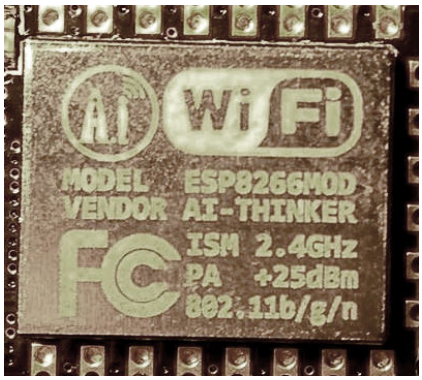
ESP8266, c’est bien ?

Voilà de quoi on va parler, du circuit ESP8266, par exemple intégré dans le module NodeMCU-V3 :



Dans ma recherche pour piloter ma petite pelle mécanique, voir l’article précédent, j’ai cherché plusieurs solutions qui n’ont pas toutes fonctionné. Par exemple, celles à base de système Bluetooth : pour l’instant c’est le fiasco ! Peut-être ai-je le mauvais oeil avec le Bluetooth, je ne sais pas ? Je vous raconterais mes déboires “bluetoothesques” dans un autre article.

La solution définie pour mon système de pelle mécanique soit le pilotage de 4 servomoteurs est le suivant :

Le pourquoi de ce choix ? C'est essentiellement parce que je souhaite séparer nettement la partie envoi de commande de la partie pilotage. ceci pour deux raisons

- *Plusieurs moyens de commande et à terme un système à apprentissage (j'en reparlerais)*
- *Rapide changement de commande*

Pour l'instant je ne pense pas doter ma pelle de capteurs, mais .....

Donc pour résumé et expliquer le pourquoi du ESP8266 dans mon cas : le pilotage (réalisé par un Arduino) réagit à des temps d'action d'entrées (commande) qui correspondent aux mouvements des servomoteurs. Possibilités d'apprentissage (enregistrement des mouvements commandés manuellement) et envoi d'un programme de mouvement (dépend de l'organe de commande).

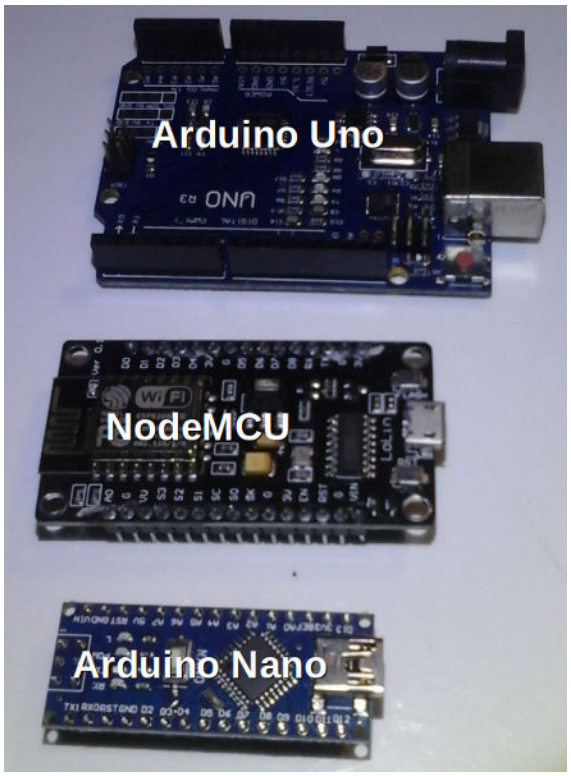
ESP8266

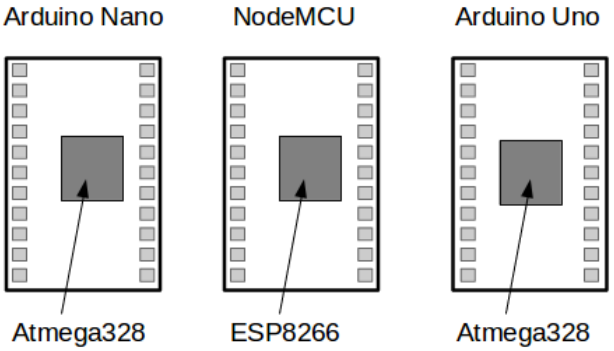
Voilà le décor est planté. Pourquoi suis-je arrivé au ESP8266 incorporé dans un NodeMCU ? Je voulais un système qui passe par le Wi-Fi. Un Yun ? un Uno avec un schield Wi-Fi ? Bref en fouillant un peu j'ai trouvé ce circuit, le NodeMCU, qui n'est pas un Arduino mais qui se programme de la même manière et cerise sur le gâteau avec l'IDE Arduino ! Le prix ???

environ 3euros 

“Petite comparaison” entre un Arduino et un NodeMCU

	Atmega328	**ESP8266**
Circuit	Arduino Uno	NodeMCU
Tension	5 V	3.3 V
Adressage	8 bits	32 bits
Fréquence d'horloge	16 MHz	80 MHz
Mémoire flash	32 kiB	0
Mémoire RAM	2 kiB	96 kiB
E/S numérique	14x dont 6 PWM	16x toutes PWM
Entrées analogiques	6 (10 bits)	1 (10 bits)
WiFi	non	oui





Une petite comparaison physique et des capacités de chaque carte. Il faut noter que sous le terme NodeMCU c’est un peu plus qu’une carte et un microcontrôleur puisque le firmware est compris.

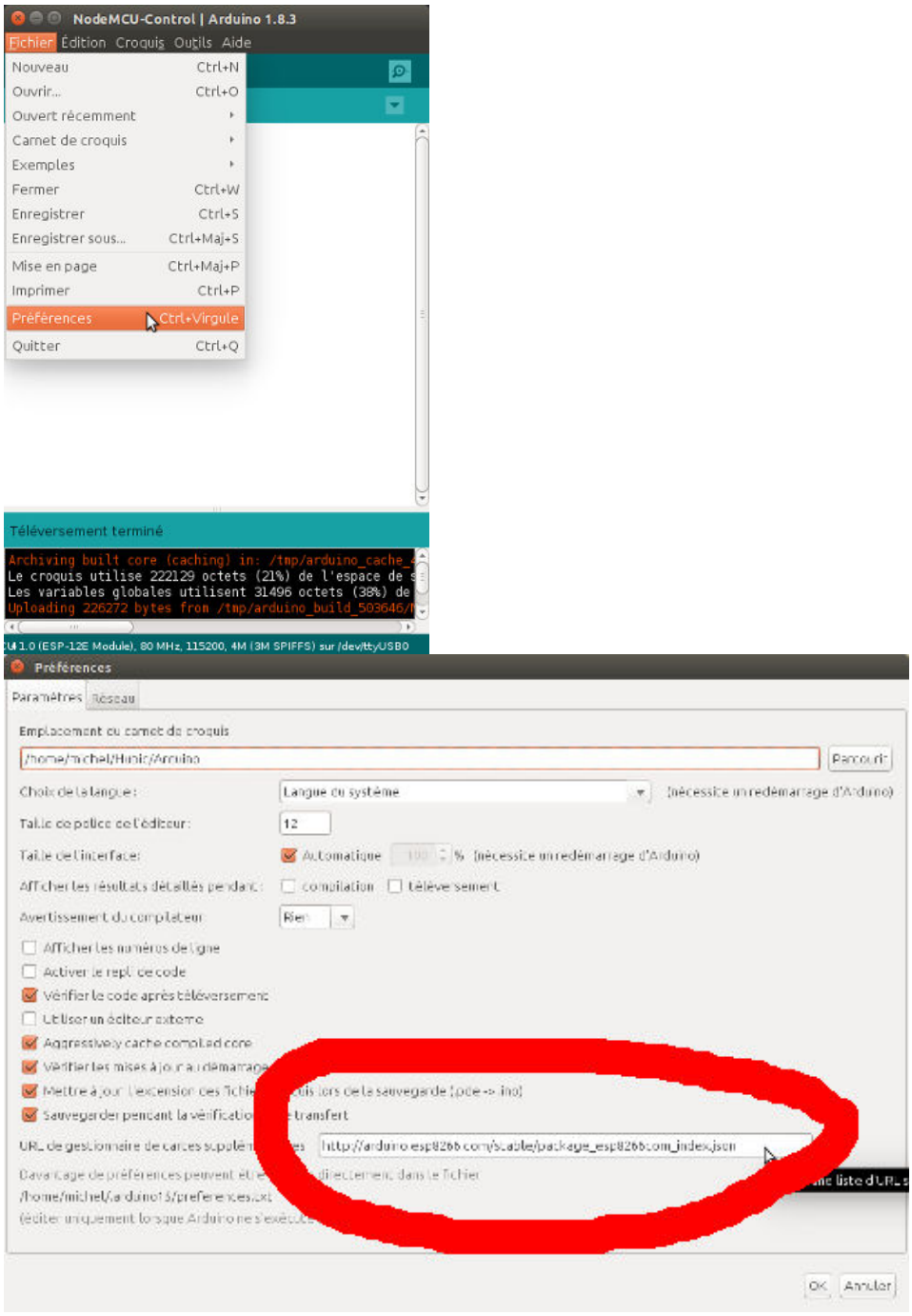
<u>Coté historique :</u>

ESP8266 voit le jour en Chine en 2013-2014 et le NodeMCU vers fin 2014. Pour l’Arduino c’est en Italie autour de 2005 et l’Atmega328 vers 1997 soit presque 15 ans d’écart !

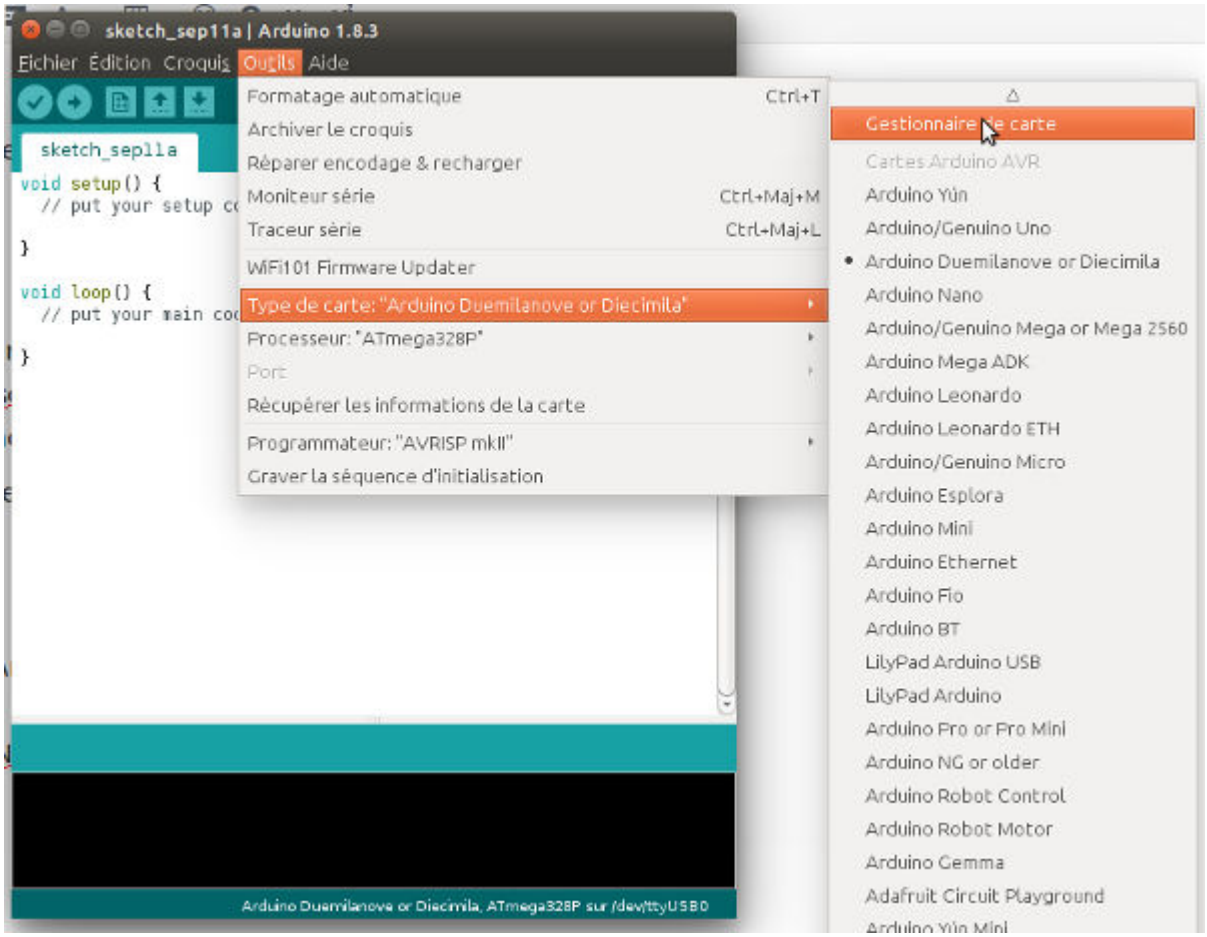
LES ÉTAPES POUR SON UTILISATION À LA “MODE ARDUINO” :

Premièrement installer la bibliothèque du ESP8266-NodeMCU, voici quelques copies d’écrans qui vous guideront, je pense que c’est simple et rapide (j’ai réussi, c’est vous dire) :

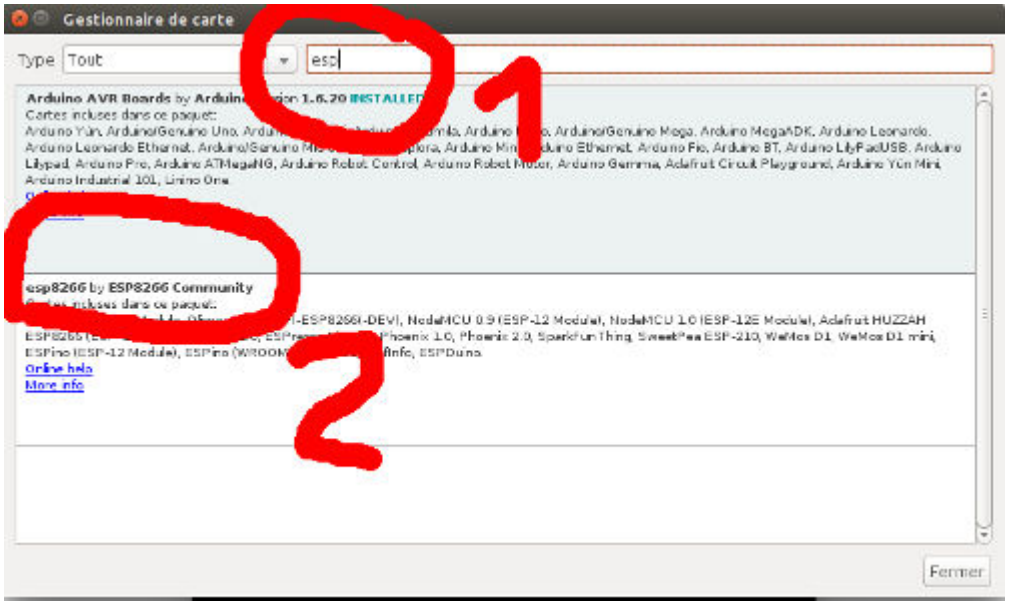
- *Ouvrez votre IDE Arduino, j’utilise la version 1.83. Il est nécessaire d’indiquer le lieu pour trouver cette nouvelle bibliothèque, Ceci se passe sous “Fichiers / Préférences” et inscrire le chemin suivant dans la zone entourée en rouge puis cliquez sur “ok” : http://arduino.esp8266.com/stable/package_esp8266com_index.json*



- *Ensuite, cliquez sur “Outils / Type de carte / Gestionnaire de carte” :*



- *Le gestionnaire de carte se chargera de diverses informations attendez que la ligne supérieure droite soit accessible et typez : ESP (majuscule ou minuscule)*



- *Vous aurez plusieurs propositions (2 c'est déjà plusieurs), choisissez "esp8266" en cliquant droit dans la bande en question.*

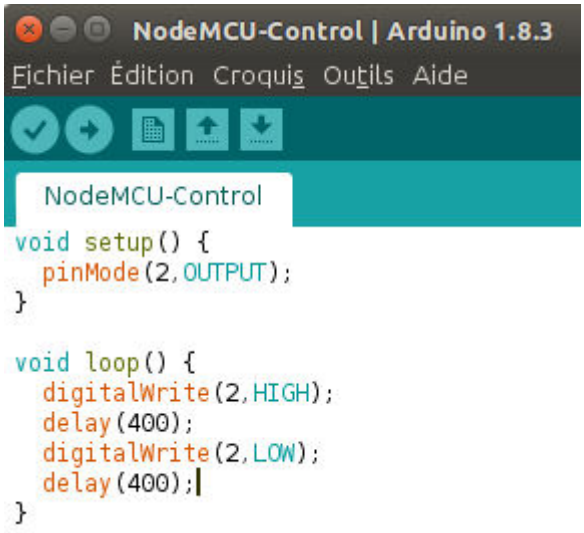
- *Choisissez la dernière version (normalement c'est celle proposée par défaut et cliquez sur installer. Allez boire un café ……. mais non c'est très rapide, mais vous pouvez quand le boire votre café !*

Voilà c'est fini pour cette partie, maintenant vous utiliserez votre module NodeMCU "comme un Arduino", branchement sur le port USB et programmation depuis l'IDE Arduino.

PREMIERS PAS AVEC LA BÊTE !

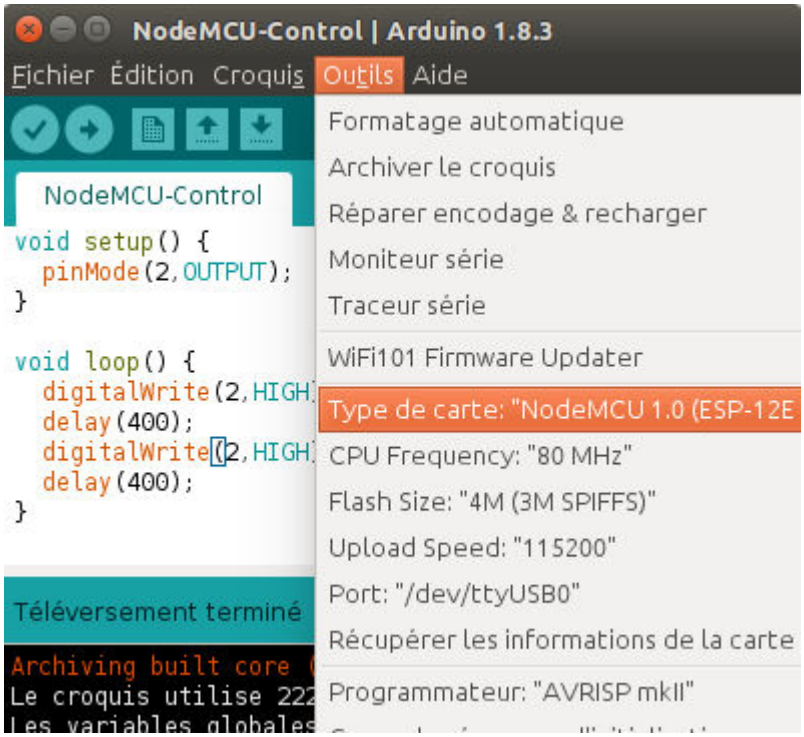
Un petit programme pour savoir si tout est fonctionnel. On va faire clignoter la led du circuit NodeMCU. Quoi ?!? Seulement ça ! Vous pensiez faire un pilotage de drone avec un suivi en vison sur votre TV ? Eeeh bien, commençons par la led on verra le reste après ……

Voici le programme, ça ira ? pas besoin d'explications !



Je ne vous fais pas l'injure à Vous, les "Paganini" du clavier, de le mettre en téléchargement. Il y a peut-être des perspicaces qui trouveront que le numéro (2) de la sortie led de contrôle est différent de l'Arduino (normalement sortie 13). C'est vrai et bravo de l'avoir remarqué ! Nous y reviendrons plus tard.

Avant le téléversement sur la carte, n'oubliez pas de sélectionner le "type de carte" et de brancher le module au PC par USB :



Une petite vidéo pour visualiser le résultat

Pour revenir à la question initiale "L'ESP8266, c'est bien ?" Oui, oui, oui, c'est bien, on peut allumer une led !! Hors plaisanterie, si vous arrivez à faire fonctionner votre circuit, c'est parfait, nous pouvons continuer dans notre découverte.

Pins NodeMCU-ESP8266

Comme subtilement introduit lors du premier programme, faites attention : les numéros des pins IDE Arduino / NodeMCU ne sont pas directement compatibles. Pour une raison étrange et que je connais pas, les numéros de pins écrit sur la carte NodeMCU ne correspond pas à celle de l'ESP8266 et donc pas à celle de l'IDE Arduino lors de la programmation, quoi qu'il en soit voici la table de compatibilité :

```
static const uint8_t D0   = 16;
static const uint8_t D1   = 5;
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;
static const uint8_t D4   = 2;
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;
static const uint8_t D9   = 3;
static const uint8_t D10  = 1;
```

Vous retrouvez tout cela ici sur le github

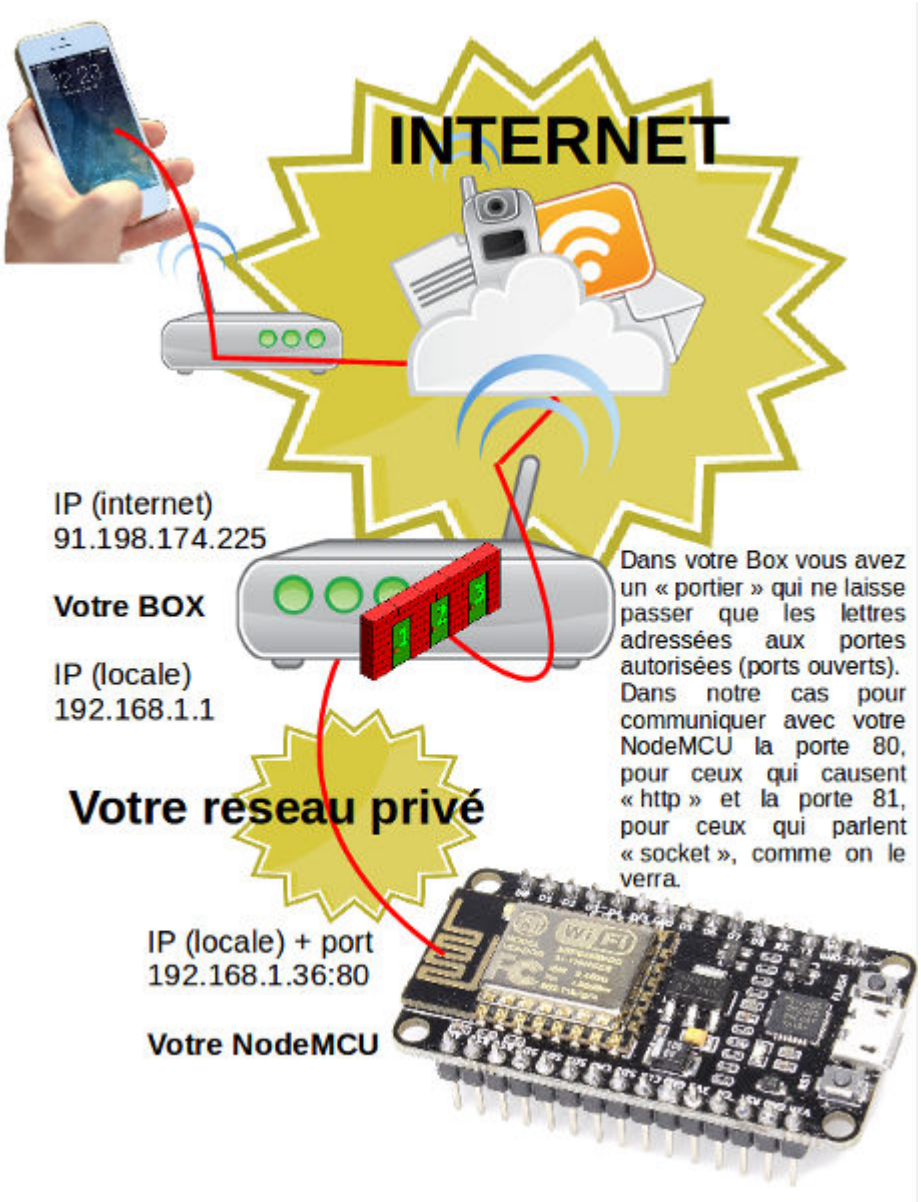
Si vous reprenez la déclaration ci-dessus en lieu et place de pin 4 vous utiliserez D4 ce qui sera directement traduit par 2. Dans cette liste vous trouvez la correspondance 7-D7-13 que l'on va utiliser sur le prochain exemple.

N'oubliez pas que le NodeMCU fonctionne en 3.3V

La partie Wi-Fi

Comment utiliser ce module pour dialoguer via le Wi-Fi ? Il est nécessaire d'ajouter quelques bibliothèques et de programmer l'affaire, mais peut-être avant de se lancer nous

allons essayer de visualiser un peu les choses. Ceci particulièrement, si vous avez envie de piloter votre ESP8266 depuis votre mobile via Internet par exemple.



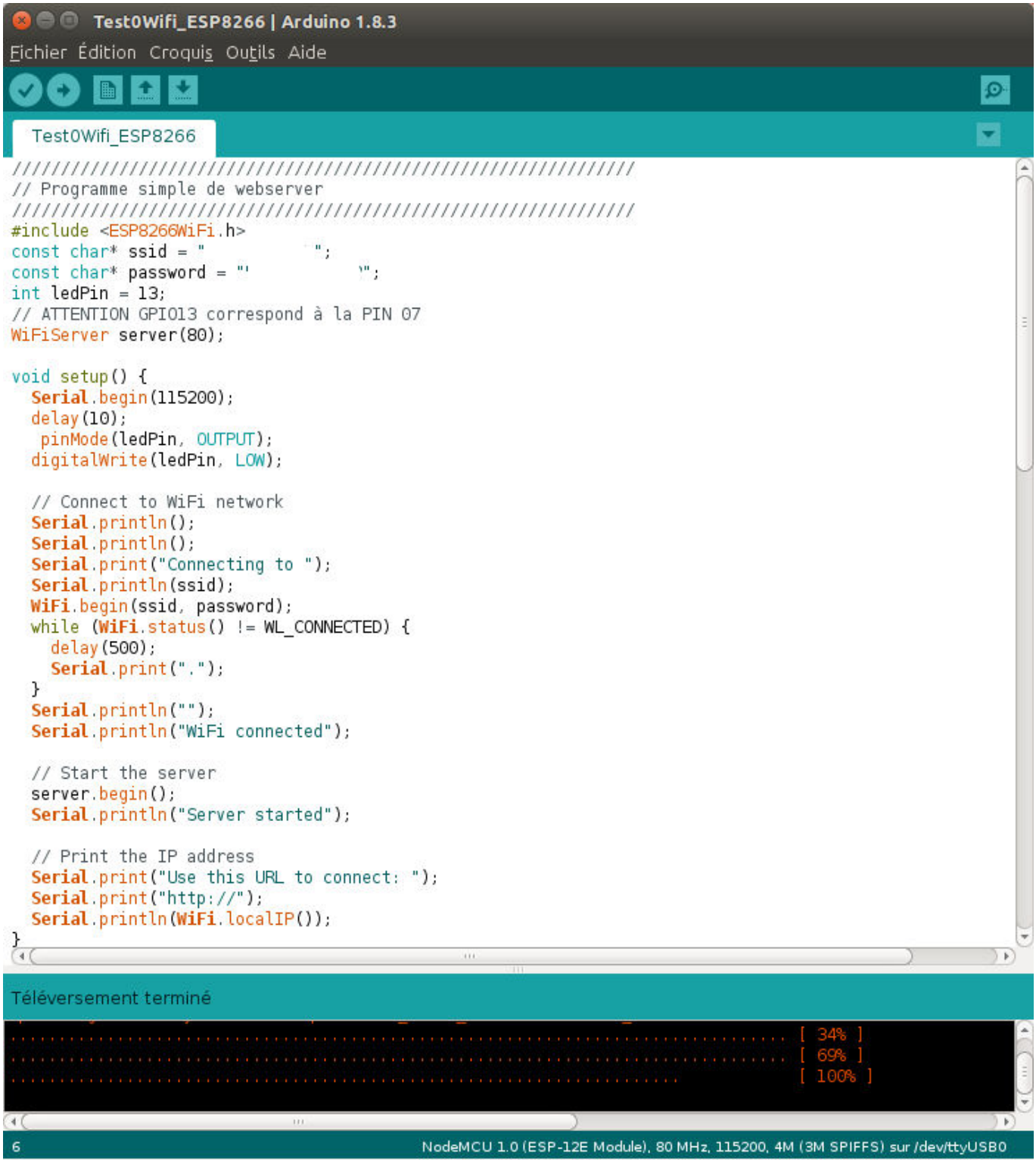
Serveur web

La première étape est la création d'un serveur web. Pour cela il est nécessaire d'ajouter au minimum la librairie ESP8266WiFi.h pour pouvoir créer un serveur web.

Voici l'allure du programme :

- *Dans l'entête : ajout de la bibliothèque, déclaration du "serverWeb".*

- *Dans setup() : Initialisation du serveur.*
- *Dans loop() : attente d'un client et réponse à la requête.*



fichier à télécharger : Test0Wifi

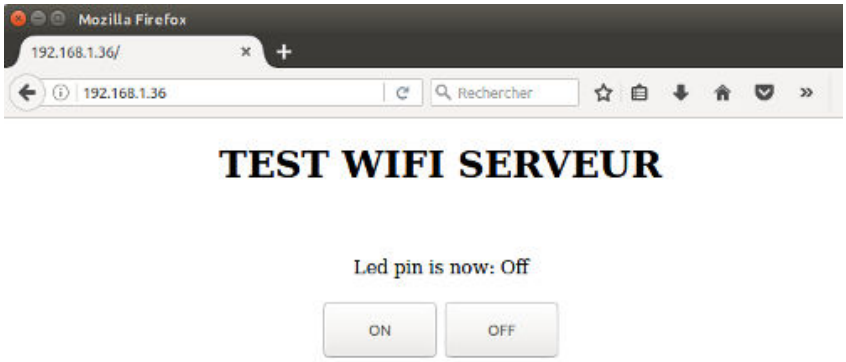
REMARQUES :

- *Le ssi, le mot de passe et le port 80 pour le HTML rien de plus standard*
- *Dans la boucle loop(), vous avez la déclaration de la page html, avec client.println(..... Cette façon de procéder est "rudimentaire" mais je pense qu'elle permet une bonne compréhension de la chose*
- *La commande "Serial.println(WiFi.localIP());" dans le setup permet d'afficher le numéro IP de votre serveurWeb du NodeMCU. Pour cela, ouvrir bien sûr le moniteur série de l'IDE Arduino. Dans mon cas 192.168.1.36*

Une fois le programme téléversé dans le ESP8266 NodeMCU et n'oubliez pas de brancher la led de contrôle, pin 7 du NodeMCU. Sur le moniteur série de l'IDE Arduino, vous aurez quelque chose comme l'image ci-contre. Vous verrez le numéro IP de votre composant.

Vérifiez la vitesse de transmission moi j'ai réglé cela sur 115200 (c'est pour dire qu'en Suisse "ça fa vite").

Vous pouvez ouvrir votre navigateur avec http://VotreNuméroIP et vous aurez la vue ci-dessous : la page web que votre NodeMCU-ESP8266 vous retourne :



Voici une petite vidéo du résultat

Voilà, vous êtes satisfait, car vous avez répondu encore une fois à la question "L'ESP8266, c'est bien ?" Oui, oui, oui, c'est bien, on peut allumer une led !! ……..

Serveur web interactif

Dans l'exemple présenté, si vous cliquez sur le bouton de la page web, le module vous retourne une nouvelle page mise à jour. À chaque nouvelle requête du client, le serveur retourne une nouvelle page. Bien maintenant imaginons la situation ou l'état de la page doit être modifié car en interne, le programme du NodeMCU ESP8266 a changé l'état de la led, par exemple un bouton pression branché sur une entrée du module dont le programme tient compte. Dans ce cas un rafraîchissement de la page est "forcé" par le serveur sur votre navigateur.

Prenons un autre cas un peu plus compliqué, si vous désirez que l'état de la led change lorsque votre mulot (souris selon Chirac) passe au-dessus de l'image de la led. Donc pour résumé :

- *curseur souris hors image led = led éteinte*
- *curseur souris sur image led = led allumée*

En fait ce que vous voulez c'est : un événement navigateur → action ESP8266

Pour réaliser cela nous allons utiliser les sockets. Ici il est peut-être temps de spécifier divers petits termes et surtout d'en bien comprendre le sens, par exemple si je vous dis : TCP/IP vous avez une idée claire de quoi je parle ? Sinon faites un petit tour ici pour parfaire votre vision de la chose. Maintenant que l'on sait transporter les messages par paquet d'un ordinateur à un autre (merci TCP/IP), il est nécessaire de spécifier "comment" et "à qui" parler. On peut imager la chose comme suit :

- *Votre téléphone avec les lignes (ondes ou fils) et toute l'infrastructure pour communiquer est en place (TCP/IP), maintenant il faut le même langage (essayez Chinois-Français) et un système pour appeler et répondre : les sockets !*

Un socket est attaché à un port (une porte) comme vous l'avez certainement deviné et pour le créer cela est simple et demande peu de code. Ce système est très ancien (relativement à l'informatique) car dans les débuts d'Unix cela était déjà implémenté.

Le programme se présente comme suit :

- *un chargement de bibliothèques*
- *String page() : la déclaration de la page web retournée par le serveur*
- *void webSocketEvent() : la gestion des événements sur le socket*
- *l'initialisation des éléments : serveur, socket, serial (moniteur série Arduino)*
- *et la fonction principale loop() répétée indéfiniment.*

Je stoppe tout de suite les remarques styles "elle est où l'image de la led ?", je l'ai remplacé par un bouton nommé "survol". Je suis d'accord c'est de la flemme, j'assume ! 😀

Par rapport au programme précédent vous trouverez une fonction de création de page web, String page(){…}, c'est pour une meilleure lecture et une plus grande facilité de maintien que tout ce qui concerne la page web a été groupé dans cette fonction. Vous trouverez d'autres méthodes sur internet (fichiers séparés ou autre instructions d'intégration).

L'autre changement est l'introduction des sockets, je vous laisse lire le code (FichierProgramme) ce n'est pas très complexe. regardons ensemble comment sont

déclarés, créés, initialisés et utilisés ces sockets.

- *WebSocketsServer webSocket = WebSocketsServer(portSocket); //port 81*
- *……..*
- *webSocket.begin();*
- *webSocket.onEvent(webSocketEvent);*
- *……..*
- *webSocket.loop();*
- *//////////////////*
- *void webSocketEvent() (gestion des événements)*

Conclusion

Je pense que vous la connaissez :

“L’ESP8266, c’est bien ?”

Oui, oui, oui, c’est bien, on peut allumer une led .

DIVERS LIENS POUR MIEUX COMPRENDRE L’ESP8266 :

- *Comparaison NodeMCU-Arduino*

- *Exemple de code pour divers capteurs (humidité, ...) Comprendre la programmation webserveur*
- *Simple http serverweb pour le ESP8266*
- *ESP8266 et une camera sur wifi ArduCAM for ESP8266*
- *Démonstration de programmation ESP8266, codeproject*