



Ateliers CREPP

Atelier Réseaux et Serveurs Web

Club de Robotique et d'Electronique
Programmable de Ploemeur

18 mars 2022
35 pages

Table des matières

Page

1	Préambule	3
1.1	Conventions	4
I	Les réseaux	5
2	Les réseaux	6
2.1	Choix des adresses IP	8
2.2	Quelques exemples de type de réseau	9
2.3	Récupération des adresse IP	9
II	Mise en place d'un serveur ESP12	12
3	Un serveur Web avec ESP12	13
3.1	Architecture du mini-projet	13
3.2	Base des requêtes	14
3.3	Connexion au routeur	15
3.4	Lancement du programme	15
3.5	Explication du programme	17
3.6	Code complet	20
III	Annexes	24
A	Utilisation de l'ESP12 sous Arduino	25
A.1	Installation des bibliothèques et cartes ESP8266	25
A.2	Recherche des cartes ESP8266	28
A.3	Recherche des cartes Arduino	30
B	Langage HTML	31
B.1	La forme de la page	31
B.2	L'en-tête	32
B.3	Le corps	33

SECTION 1

PRÉAMBULE

- ▶ Document réalisé en L^AT_EX par Nicolas Le Guerroué pour le Club de Robotique et d'Electronique Programmable de Ploemeur (CREPP)
- ▶ Permission vous est donnée de copier, distribuer et/ou modifier ce document sous quelque forme et de quelque manière que ce soit.
- ▶ Version du 18 mars 2022
- ▶ Taille de police : 11pt
- ☎ 06.20.88.75.12
- ✉ nicolasleguerroue@gmail.com
- 🐱 https://github.com/CREPP-PLOEMEUR/Supports_PDF/tree/main/Sources%20Latex
- ▶ **Dans la mesure du possible, évitez d'imprimer ce document si ce n'est pas nécessaire. Il est optimisé pour une visualisation sur un ordinateur et contient beaucoup d'images.**

Versions

octobre 2021	Fusion des supports d'ateliers
novembre 2021	Ajout de l'atelier sur les servomoteurs
décembre 2021	Ajout de l'atelier sur les moteurs pas-à-pas
janvier 2022	Ajout de l'annexe pour l'installation des bibliothèques ESP8266
février 2022	Ajout de l'annexe pour le serveur Web ESP8266 NodeMCU
mars 2022	Ajout de l'annexe pour la partie Réseaux (Adressage)






Conventions

Les commandes à saisir sont dans des encadrés similaires :

```
sudo apt-get update
```

Exemple de commande

Parfois, ces encadrés contiendront des instructions qu'il faudra placer dans certains fichiers.

- Les fichiers sont indiqués par le repère  fichier
- Les dossiers sont indiqués par le repère  dossier
- Les logiciels sont indiqués par le repère  logiciel¹
- Les adresses IP sont indiquées par le repère  Adresse IP
- Les adresses MAC sont indiquées par le repère  Adresse MAC

1. Sont également concernés les paquets Linux et les bibliothèques des langages

Première partie

Les réseaux

Introduction aux réseaux

SECTION 2

LES RÉSEAUX

Un réseau est un ensemble de machines¹ reliées entre elles. Dans notre cas, la liaison sur le réseau se fera en Ethernet ou WiFi.

Pour communiquer entre elles, il faut définir des adresses (IP²) exactement comme une adresse postale pour transmettre un courrier.

Les adresses MAC

Une adresse MAC (Medium Access Control) est une adresse unique qui désigne la machine³. Elle est de la forme **MAC** `XX:XX:XX:XX:XX:XX` (six octets usuellement exprimés et hexadécimal) et reste invariante dans le temps.

Contrairement aux adresses IP, les adresses MAC ne sont normalement pas attribuées explicitement par configuration ; elles sont au contraire attribuées à la production de la carte réseau par son fabriquant

La table ARP⁴ d'une machine sert à associer des adresses IP à des adresses MAC.

Les adresse IP

Une adresse IP est une adresse donnée à une machine qui rejoint le réseau.


A quoi ressemble une adresse IP ?

- Version 4 (Ipv4) : Les adresses sont codées sur 32 bits

IP `192.168.0.1` = **BIN** `11000000.10101000.00000000.00000001` (En Binaire)

- version 6 (Ipv6) : les adresses sont codées sur 128 bits

1. Serveurs, PC, Tablettes, téléphones, imprimantes
2. Internet Protocol
3. Plus précisément l'adresse de la carte réseau de la machine
4. Address Resolution Protocol

 FE80 :0000 :0000 :0000 :020C :76FF :FE21 :1C3B]. L'adresse de version 4 (IPv4) est encore actuellement la plus utilisée.

Dans un réseau, chaque machine possède une adresse IP fixée par l'administrateur du réseau. Il est interdit de donner la même adresse à 2 machines différentes sous peine de dysfonctionnement.

Une adresse IPv4 est une suite de 32 bits (4 octets) notée en général a.b.c.d avec a, b, c, et d des entiers « décimal » compris entre 0 et 255. Chaque valeur a, b, c ou d représente dans ce cas une suite de 8 bits.

Exemple 1. Une machine qui a comme adresse IP 134.214.80.12 (En décimal) :

- a vaut 134 soit (1000 0110) en binaire.
- b vaut 214 soit (1101 0110) en binaire.
- c vaut 80 soit (0101 0000)
- d vaut 12 soit (00001100).

En binaire, l'adresse IP s'écrit  10000110.1101 0110.0101 0000.0000 1100 et puisque le codage se fait sur 8 bits les valeurs seront obligatoirement comprises entre 0 et 255.

Le net-id et le host-id



Au sein d'un même réseau IP, toutes les adresses IP commencent par la même suite de bits. L'adresse IP d'une machine va en conséquence être composée de 2 parties :

- Net-id : La partie fixe de l'adresse IP
- Host-id : La partie réservée aux machines qui viennent sur le réseau

Masque de réseau IP

Le masque du réseau permet de connaître le nombre de bits du net-id. On appelle N ce nombre. Il s'agit d'une suite de 32 bits composée en binaire de N bits à 1 suivis de 32-N bits à 0. C'est le masque du réseau qui définit la taille d'un réseau IP : c'est-à-dire la plage d'adresses assignables aux machines du réseau.

Ainsi, pour connaître le net-id, on va faire un ET (& ou .) logique entre le masque et l'adresse IP.

Exemple 2. Prenons l'adresse IP  192.168.1.0 et le masque  255.255.255.0.

Que vaut le net-id ?

Pour rappel :

- ▶ $0.0 = 0$
- ▶ $0.1 = 0$

► $1.0 = 0$

► $1.1 = 1$

Solution 1. L'adresse IP `192.168.1.0` vaut BIN `1100000.10101000.00000000.00000001` et
L'adresse MAS `255.255.255.0` vaut BIN `11111111.11111111.11111111.00000000`

$$\begin{aligned} & 01100000.10101000.00000000.00000001 \\ \text{ET} & 11111111.11111111.11111111.00000000 \\ = & 11111111.11111111.00000000.00000000 = \text{BIN } 11111111.11111111.11111111.00000000 \end{aligned}$$

L'adresse IP `192.168.0.0` obtenue représente donc **l'adresse du réseau**.

Les autres bits à 0 sont donc réservés aux périphériques du réseau. Etant sur 8 bits, il y a donc $2^8 - 2$ adresses disponibles sur ce réseau.

Les adresses interdites

Il est interdit d'attribuer à une machine d'un réseau l'adresse du réseau et l'adresse de broadcast (diffusion).

L'adresse de diffusion

Cette adresse permet à une machine d'envoyer une info à toutes les machines d'un réseau. Cette adresse est celle obtenue en mettant tous les bits de l'host-id à 1. Dans notre cas c'est l'adresse IP `192.168.0.255`

L'adresse de réseau

Cette adresse est celle obtenue après avoir fait le ET entre l'adresse IP et le masque de sous-réseau. Dans notre cas c'est l'adresse IP `192.168.0.0`

Choix des adresses IP

Sur un réseau local, les adresses sont choisies par un serveur DHCP⁵.

Les adresses peuvent être :

- Statiques : Un appareil sur le réseau possède la même adresse après connexion puis déconnexion.
- Dynamiques : l'adresse IP varie dans le temps au bout d'une période d'expiration après une déconnexion (bail d'une journée par exemple)

A de rares exceptions, vous n'êtes pas censé attribuer une adresse IP vous-même à une machine.

5. **DHCP** : Dynamic Host Configuration Protocol

Quelques exemples de type de réseau

Un réseau IP peut avoir une taille très variable :

- Une entreprise moyenne aura un réseau comportant une centaine de machines.
- Un campus universitaire aura un réseau comportant de quelques milliers à quelques dizaines de milliers de machines.
- Le réseau d'une multinationale (un grand fournisseur d'accès par exemple) peut comporter des millions de postes.

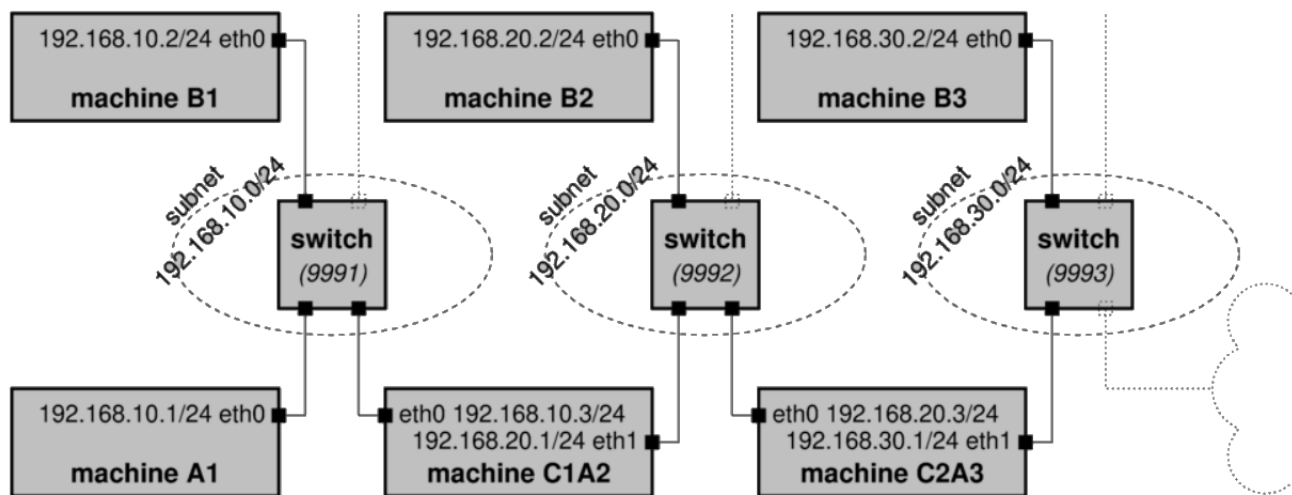


FIGURE 2.1 – Un réseau plus évolué

Il existe 2 types de réseau :

- Les réseaux publics Internet où chaque équipement connecté doit posséder une adresse unique et enregistrée au niveau mondial.
- Les réseaux privés, dans ce cas le choix des adresses est libre et ne doivent être uniques que dans ce réseau.

Si un réseau privé doit être interconnecté avec le réseau Internet, il faudra alors utiliser des adresses privées qui ne puissent correspondre à des adresses publiques utilisées sur Internet. Des plages d'adresses réservées à usage privé existent et elles ne sont donc pas acheminées par les routeurs Internet, ce qui supprime tout risque de conflit (cf. document annexe).

Récupération des adresse IP

Voici 2 commandes pour récupérer l'adresse IP et le masque de sous-réseau.

Pour Windows

Sous un terminal Windows⁶

```
ipconfig
```

Récupération des informations IP sous Windows

```
C:\Users\Admin>ipconfig

Configuration IP de Windows

Carte Ethernet Ethernet :

    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6. . . . . : 2a01:e0a:227:a340:a4f7:9a72:aa0a:347f
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:2d8f:1bca:f420:b777
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:3d02:1319:b97a:cbdb
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:4448:9d1:fe45:3916
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:907f:4342:42d7:8451
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:99ff:f749:3636:af2e
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:ac70:9311:f199:596e
    Adresse IPv6 temporaire . . . . . : 2a01:e0a:227:a340:ddc1:2e12:5b54:a4ed
    Adresse IPv6 de liaison locale. . . . : fe80::a4f7:9a72:aa0a:347f%7
    Adresse IPv4. . . . . : 192.168.1.41
    Masque de sous-réseau. . . . . : 255.255.255.0
    Passerelle par défaut. . . . . : fe80::72fc:8fff:fe47:689c%7
                                   192.168.1.254
```

FIGURE 2.2 – La commande ipconfig

Pour Linux

Sous un terminal Linux, on peut utiliser la commande `ifconfig`⁷

```
ifconfig
```

Récupération des informations IP sous Linux

6. Saisir **cmd** dans le gestionnaire de programme

7. Si non installée sous Linux, saisir `sudo apt-get install net-tools`

```
nico@nico-ThinkPad-L580:~$ ifconfig
enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether e8:6a:64:7d:be:44 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Boucle locale)
    RX packets 2228 bytes 337363 (337.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2228 bytes 337363 (337.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp5s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.49 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2a01:e0a:227:a340:2080:e8d3:7ee3:2b7e prefixlen 64 scopeid 0x0<global>
    inet6 fe80::c48e:e202:5ade:c5fc prefixlen 64 scopeid 0x20<link>
    inet6 2a01:e0a:227:a340:d757:63b3:a442:734e prefixlen 64 scopeid 0x0<global>
    ether fc:77:74:1e:76:ed txqueuelen 1000 (Ethernet)
    RX packets 42721 bytes 38163746 (38.1 MB)
    RX errors 0 dropped 6 overruns 0 frame 0
    TX packets 24604 bytes 6842107 (6.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

FIGURE 2.3 – La commande ipconfig

Deuxième partie

Mise en place d'un serveur ESP12

Led ESP8266

Contrôle de la LED sur la broche **D4**

Allumer **Eteindre**

Serveur Web avec ESP12

SECTION 3

UN SERVEUR WEB AVEC ESP12

L'objectif de ce chapitre est de créer un serveur Web pour contrôler la led interne de l'ESP12, une carte basée sur les ESP8266 (Broche D4)

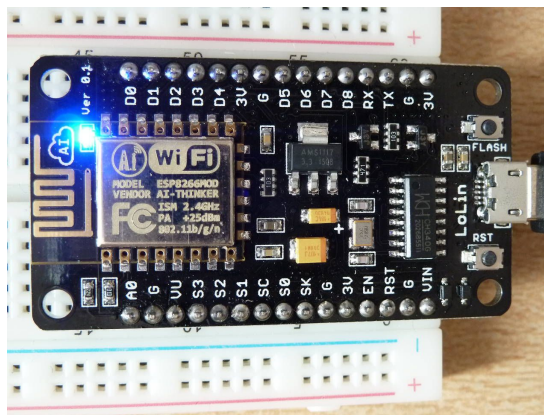


FIGURE 3.1 – La led interne de l'ESP

Architecture du mini-projet

Pour communiquer entre le client (utilisateur) et l'ESP12, nous utiliserons un routeur qui servira de passerelle.

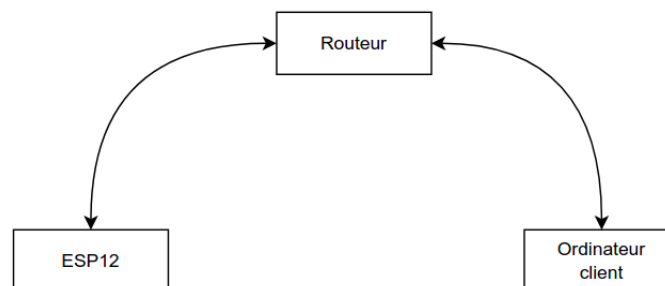


FIGURE 3.2 – Architecture du projet

L'ESP12 se connecte dans un premier temps au routeur. Une fois connecté, tout client sur le

même réseau peut se connecter à l'ESP12 en saisisant l'adresse de l'ESP12 dans le navigateur.

Base des requêtes

Dès que nous allons sur une page Web, nous faisons une requête, c'est à dire que l'on va demander l'affichage d'une page Web à un serveur distant.

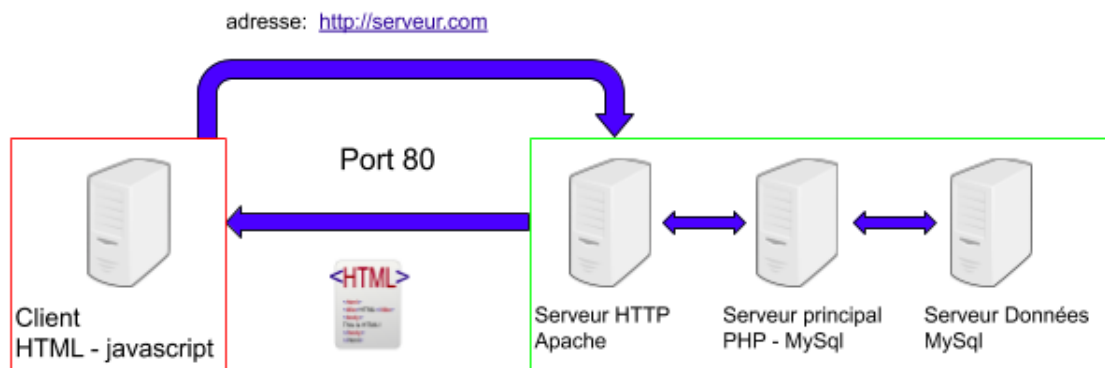


FIGURE 3.3 – Architecture client-serveur

Afin de récupérer une page sur le serveur, nous avons besoin de connaître 3 éléments :

- ▶ L'adresse IP ou l'adresse du serveur : Ici ce sera l'adresse IP de l'ESP12
- ▶ L'emplacement de la page sur le serveur : L'emplacement `/'` désigne la racine du serveur
- ▶ Le port de communication entre le client et le serveur : `port 80` par défaut pour le protocole HTTP¹

Une petite explication sur les ports

Pour recevoir et transmettre des données à d'autres ordinateurs, un ordinateur (ou serveur) a besoin de ports. Cependant, les ports physiques tel que le port Ethernet communiquent avec beaucoup de services.

Ainsi, des ports virtuels ont été créés.

Chaque port virtuel est codé sur 16 bits et permet de faire communiquer un service ou un logiciel.² Il y a donc potentiellement 65536 ports disponibles. Certains numéros de port sont réservés à certains services

Les ports de 0 à 1023 sont déjà réservés à des services particuliers et le port par défaut pour les serveurs Web est le 80.

1. le protocole HTTPS utilise le port 443

2. Par exemple, le service SSH communique sur le port 22

Les types de requêtes

Lorsque nous nous connectons à un serveur Web, nous faisons principalement deux types de requêtes³

► Requêtes GET

Les requêtes GET sont des requêtes avec des éléments passés via l'URL de la page. Elles sont donc visibles via la barre d'adresse :

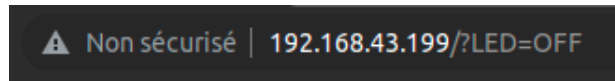


FIGURE 3.4 – L'adresse avec la requête GET

Chaque élément est séparé avec le symbole **LBL** `&` et la liste des arguments commencent avec le symbole **LBL** `?`. Comme nous avons un seul élément passé via l'URL, nous n'avons pas le symbole **LBL** `&`.

Ici, nous avons un argument **ARG** `LED` avec la valeur **VAL** `OFF`.

► Requêtes POST

Ces requêtes ne passent pas les arguments via l'adresse URL. Cette section ne sera pas abordée dans le cadre de l'atelier.

Connexion au routeur

La carte ESP12 a besoin du nom du routeur ainsi que de son mot de passe. Dans le programme **FILE** `serveurCREPP.ino`, il faut préciser le nom et le mot de passe sur les lignes suivantes :

```
//Par exemple
const char* ssid      = "Creafab_invite";
const char* password = "MonTraficEstJournalise";
//Il faut mettre le nom du routeur chez soi et le mot de passe associé
```

Identifiant et mot de passe

Lancement du programme

Pour sélectionner la carte ESP12, veuillez vous reporter à l'annexe **UTILISATION DE L'ESP12 SOUS ARDUINO**.

3. Les webSockets ne seront pas abordées

Remarque

Il faudra activer la liaison série de la carte. Pour cela, lors du choix de la carte dans le logiciel Arduino, dans la section **Outils > Types de cartes > ESP12 NodeMCU**, il faut bien vérifier que la case **Serial** est activée

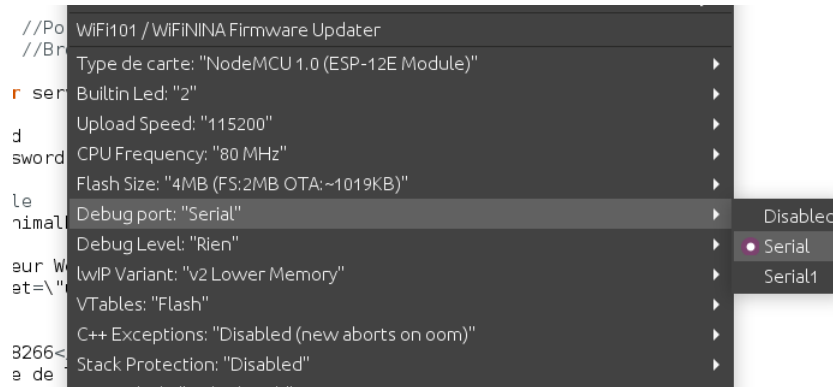


FIGURE 3.5 – Sélection du mode Série

Une fois le programme téléversé, il vous faudra récupérer l'adresse IP de l'ESP12.

Pour cela, une fois que le code est téléversé, veuillez ouvrir la fenêtre du moniteur série, l'adresse IP de l'ESP va apparaître au bout de quelques secondes.

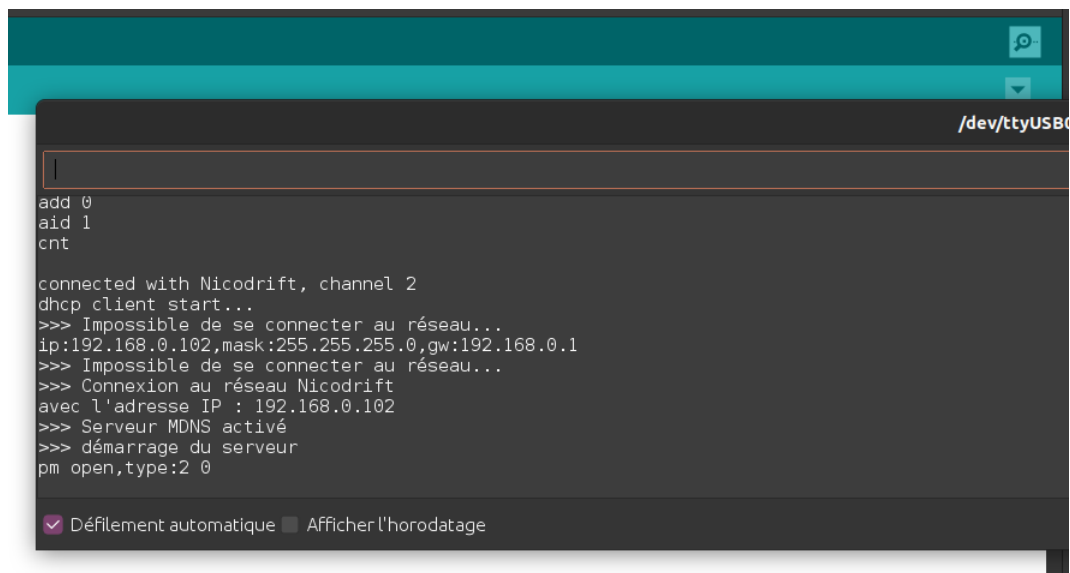


FIGURE 3.6 – Affichage de l'adresse IP

Si aucune données n'apparaît, faite un reset de la carte ESP12 en appuyant sur la touche **RST** de la carte.

Il ne vous reste plus qu'à rentrer l'adresse IP obtenue dans un navigateur internet.

En l'occurrence, l'adresse IP dans ce cas là est `192.168.0.102`

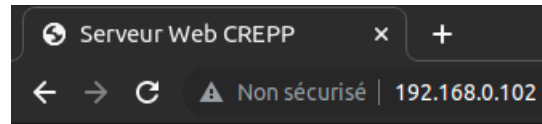


FIGURE 3.7 – Connexion au serveur

Le résultat doit être le suivant

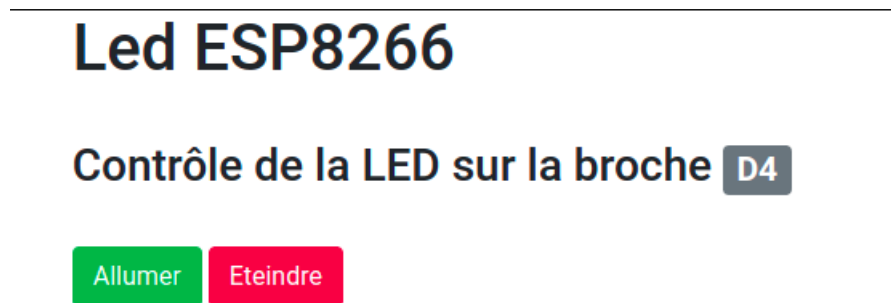


FIGURE 3.8 – Résultat

Explication du programme

Une explication du langage HTML est disponible en annexe (section HTML)

En tête du code

Après avoir importé les bibliothèques liées à l'ESP12, nous définissons un objet **ESP8266WebServer** qui attend comme argument le port du serveur, c'est à dire le 80.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#define PORT 80 //Port par défaut
#define LED D4 //Broche de la LED

ESP8266WebServer server(PORT);
```

Importation des bibliothèques

On précise ensuite le mot de passe et le nom du routeur de communication.

```
const char* ssid      = "Nom-reseau-Internet";
const char* password = "Mot-de-passe-Routeur";
```

Identifiant et mot de passe du routeur

Ensuite, nous allons définir et créer notre page HTML.

Deux versions de pages Web seront proposées :

- Une version minimale sans aucune mise en forme ajoutée.

```
const String minimalPageContent = "<html>\n
<head>\n
  <title>Serveur Web CREPP</title>\n
  <meta charset=\"utf-8\"/> \n
</head>\n
<body>\n
  <h1>Led ESP8266</h1><br>\n
  <h3>Contrôle de la LED sur la broche D4</h3><br>\n
  <a href=\"//?LED=ON\"><button >Allumer</button></a>\n
  <a href=\"//?LED=OFF\"><button >Eteindre</button></a>\n
</body>\n
</html>";
```

Page minimale

- Une version plus élaborée en utilisant la bibliothèque  qui permet de faire de jolie mise en forme très facilement.

```
const String fullPageContent = "<html>\n
<head>\n
  <title>Serveur Web CREPP</title>\n
  <meta charset=\"utf-8\"/> \n
  <link rel=\"stylesheet\" href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css\" integrity=\"sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T\" crossorigin=\"anonymous\">\n
</head>\n
<body style=\"margin-left:5%;\">\n
  <h1>Led ESP8266</h1><br>\n
  <h3>Contrôle de la LED sur la broche <span class=\"badge badge-secondary\">D4</span></h3><br>\n
  <a href=\"//?LED=ON\"><button class=\"btn btn-success\">Allumer</button></a>\n
  <a href=\"//?LED=OFF\"><button class=\"btn btn-danger\">Eteindre</button></a>\n
</body>\n
```

```
</html>";
```

Page plus élaborée

Fonction setup

Lançons nous dans le code de la fonction **setup**

On définit la led en sortie et on se connecte au routeur.

```
void setup() {  
  
    pinMode(LED, OUTPUT);           //LED en sortie  
    digitalWrite(LED, LOW);         //LED éteinte  
    Serial.begin(115200);            //Communication à 115200 bits/s  
    WiFi.begin(ssid, password);     //Connexion  
    Serial.println("");              //Retour à la ligne
```

Initialisation

Puis on vérifie que nous sommes bien connecté et on affiche les informations de connexion.

```
while (WiFi.status() != WL_CONNECTED)  
{  
    delay(500);  
    Serial.println(">>> Impossible de se connecter au réseau...");  
} //Fin while  
  
Serial.print(">>> Connexion au réseau ");  
Serial.println(ssid);  
Serial.print("avec l'adresse IP : ");  
Serial.println(WiFi.localIP());
```

Vérification de la connexion

Enfin on vérifie que le MDSN⁴ est activé (optionnel)

```
if (MDNS.begin("esp8266")) {        //Multicast DNS  
    Serial.println(">>> Serveur MDNS activé");  
}  
}
```

Vérification de la connexion

On définit (toujours dans le setup) les pages accessibles par le client ainsi que la fonction appelée lors de la requête. Il s'agit de l'emplacement `/'` et sa fonction associée est la fonction **mainPage**

4. Multicast DNS, un serveur de résolution de nom de domaine

```
server.on("/", mainPage);           //Affichage de la page principale si requête sur '/' -> saisir IP dans le navigateur
```

Redirection sur la page principale

On redirige également l'utilisateur sur une page dédiée si l'adresse demandée n'existe pas.

```
server.onNotFound(notFoundPage);    //Affichage de la page d'erreur si adresse non valide
```

Redirection en cas d'erreur

Enfin, on initialise le serveur.

```
server.begin();                     //Initialisation du serveur  
Serial.println(">>> démarrage du serveur");
```

Initialisation du serveur

Fonction loop

Le code à l'intérieur de la fonction **loop** se contente de gérer de manière transparente le comportement du serveur.

```
void loop()  
{  
    server.handleClient(); //Gestion des clients sur le serveur  
} //Fin loop
```

Fonction principale

Code complet

```
#include <ESP8266WiFi.h>  
#include <ESP8266WebServer.h>  
#include <ESP8266mDNS.h>  
  
#define PORT 80 //Port par défaut  
#define LED D4  //Broche de la LED  
  
ESP8266WebServer server(PORT);  
  
const char* ssid      = "Nom-reseau-Internet";  
const char* password  = "Mot-de-passe-Routeur";  
  
//Page principale
```

```
const String minimalPageContent = "<html>\n\
<head>\n\
  <title>Serveur Web CREPP</title>\n\
  <meta charset=\"utf-8\"/> \n\
</head>\n\
<body>\n\
  <h1>Led ESP8266</h1><br>\n\
    <h3>Contrôle de la LED sur la broche D4</h3><br>\n\
    <a href=\"/?LED=ON\"><button >Allumer</button></a>\n\
    <a href=\"/?LED=OFF\"><button >Eteindre</button></a>\n\
  </body>\n\
</html>";

const String fullPageContent = "<html>\n\
  <head>\n\
    <title>Serveur Web CREPP</title>\n\
    <meta charset=\"utf-8\"/> \n\
    <link rel=\"stylesheet\" href=\"https://stackpath.bootstrapcdn.com/\n\
bootstrap/4.3.1/css/bootstrap.min.css\" integrity=\"sha384-\n\
gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T\" \n\
crossorigin=\"anonymous\">\n\
  </head>\n\
  <body style=\"margin-left:5%;\">\n\
    <h1>Led ESP8266</h1><br>\n\
    <h3>Contrôle de la LED sur la broche <span class=\"badge badge-secondary\n\
\">D4</span></h3><br>\n\
    <a href=\"/?LED=ON\"><button class=\"btn btn-success\">Allumer</button\n\
></a>\n\
    <a href=\"/?LED=OFF\"><button class=\"btn btn-danger\">Eteindre</button\n\
></a>\n\
  </body>\n\
</html>";

void setup() {

  pinMode(LED, OUTPUT);           //LED en sortie
  digitalWrite(LED, LOW);         //LED éteinte
  Serial.begin(115200);           //Communication à 115200 bits/s
  WiFi.begin(ssid, password);    //Connexion
  Serial.println("");             //Retour à la ligne

  while (WiFi.status() != WL_CONNECTED)
  {
```

```
        delay(500);
        Serial.println(">>> Impossible de se connecter au réseau...");
    }

    Serial.print(">>> Connexion au réseau ");
    Serial.println(ssid);
    Serial.print("avec l'adresse IP : ");
    Serial.println(WiFi.localIP());

    if (MDNS.begin("esp8266")) {    //Multicast DNS
        Serial.println(">>> Serveur MDNS activé");
    }

    server.on("/", mainPage);        //Affichage de la page principale si requête sur '/' -> saisir IP dans le navigateur
    server.onNotFound(notFoundPage); //Affichage de la page d'erreur si adresse non valide

    server.begin();                  //Initialisation du serveur
    Serial.println(">>> démarrage du serveur");

} //Fin setup

void loop()
{

    server.handleClient(); //Gestion des clients sur le serveur

} //Fin loop

void mainPage() //Page principale
{

    if(server.arg("LED")=="ON") //Lecture de l'argument 'LED'
    {
        digitalWrite(LED, LOW); //On allume la led
    } //Fin if
    else
    {
        digitalWrite(LED, HIGH); //On eteint la led
    } //Fin else

    server.send(200, "text/html", fullPageContent); //On envoie la page principale
}
```

```
}//Fin mainPage

void notFoundPage() //Gestion si mauvaise URL
{
    server.send(404, "text/plain", "Page introuvable !\n\n");
}

//Fin notFoundPage
```

Code complet

Troisième partie

Annexes

ANNEXE A

UTILISATION DE L'ESP12 SOUS ARDUINO

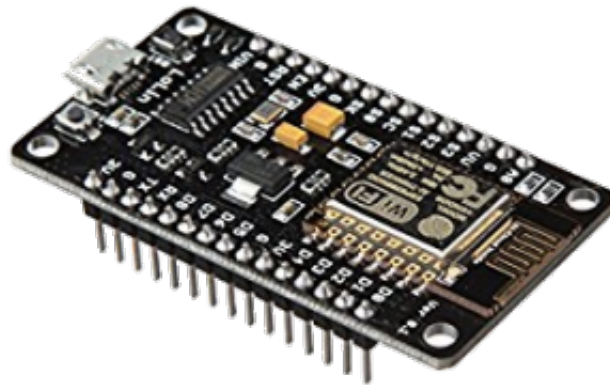


FIGURE A.1 – ESP12 NodeMCU

Installation des bibliothèques et cartes ESP8266

La carte ESP12 NodeMCU est prévue pour être programmée directement via l'IDE¹ Arduino. Cette carte fait partie de la grande famille des ESP8266.

Pour installer les bibliothèques et cartes sur le logiciel Arduino, il faut réaliser les étapes suivantes :

- Ouvrir les préférences du logiciel Arduino dans  Fichiers - Préférences

1. IDE : Environnement de Développement Intégré

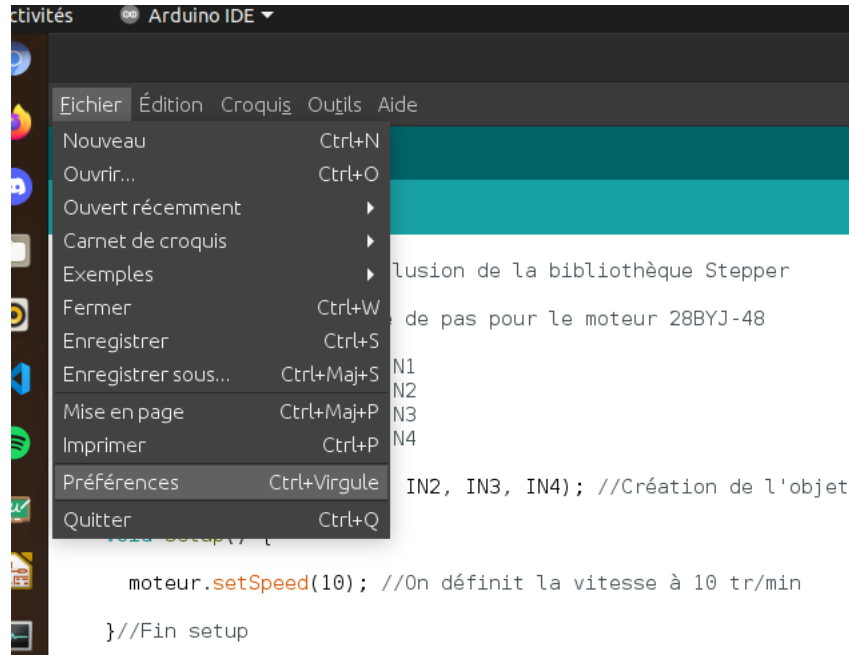


FIGURE A.2 – Préférences Arduino

- Dans le champ **URL de gestionnaire de cartes supplémentaires** , mettre le lien suivant :

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

Avertissement

Veuillez vérifier l'URL après le copier-coller car les underscores ("tirets du 8") peuvent disparaître.

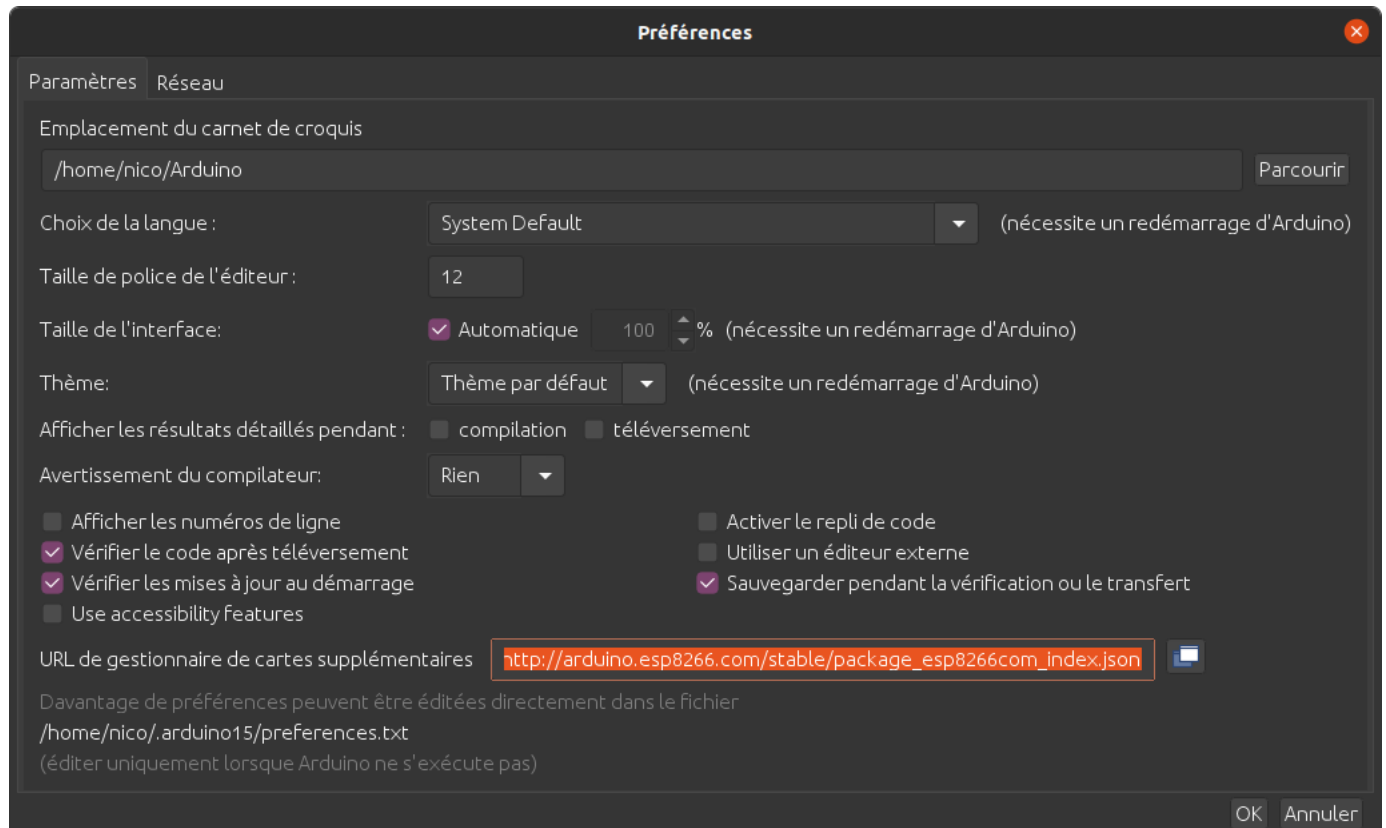



FIGURE A.3 – Lien pour les cartes ESP8266

Puis faire  OK

- ▶ Fermer le logiciel Arduino
- ▶ Lancer le logiciel Arduino
- ▶ Allez dans  Outils - Type de carte - Gestionnaire de carte

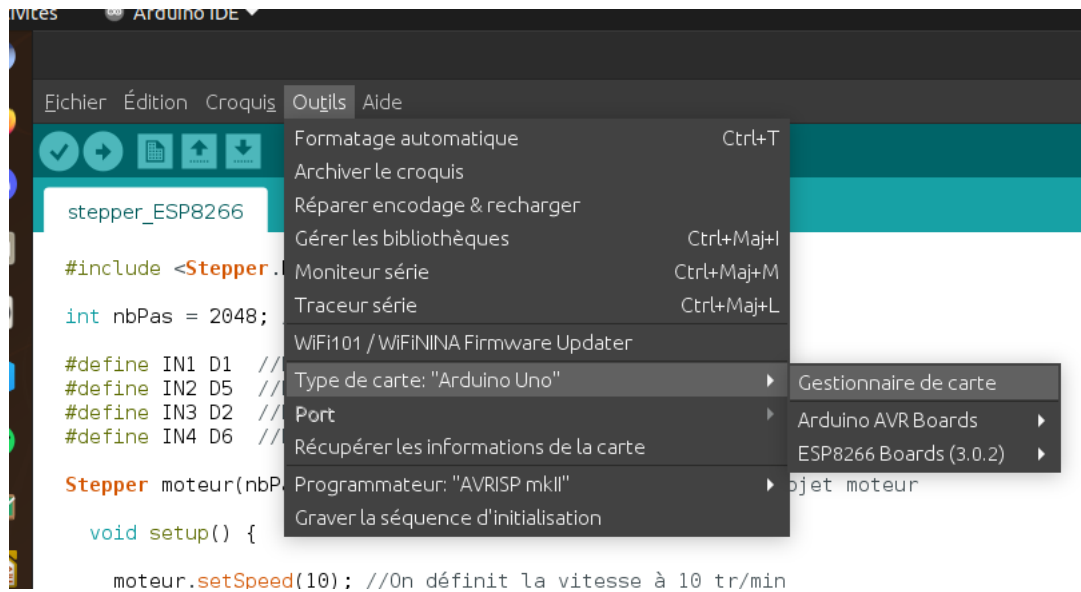


FIGURE A.4 – Gestionnaire des cartes ESP8266

et faire une recherche avec le mot clé **KEY** `esp8266`

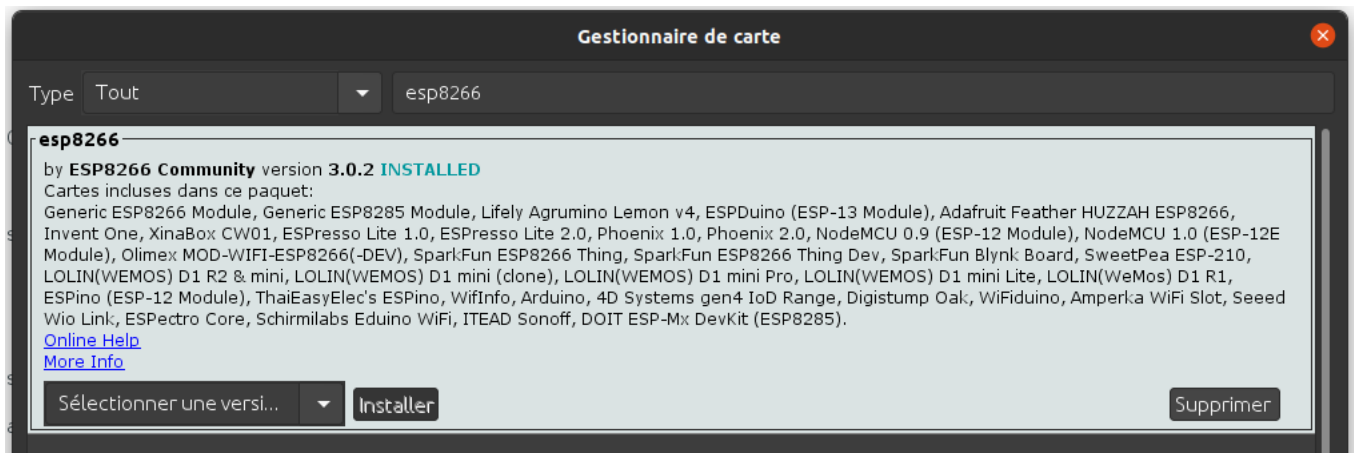


FIGURE A.5 – Installation des bibliothèques ESP8266

Il ne vous reste plus qu'à cliquer sur **KEY** `Installer` et redémarrer le logiciel Arduino.

Recherche des cartes ESP8266

Lors de la programmation d'une carte ESP8266 NodeMCU, il faudra donc aller dans

KEY `Outils - Type de carte - ESP8266 Boards NodeMCU X.X (ESP12 Module)`

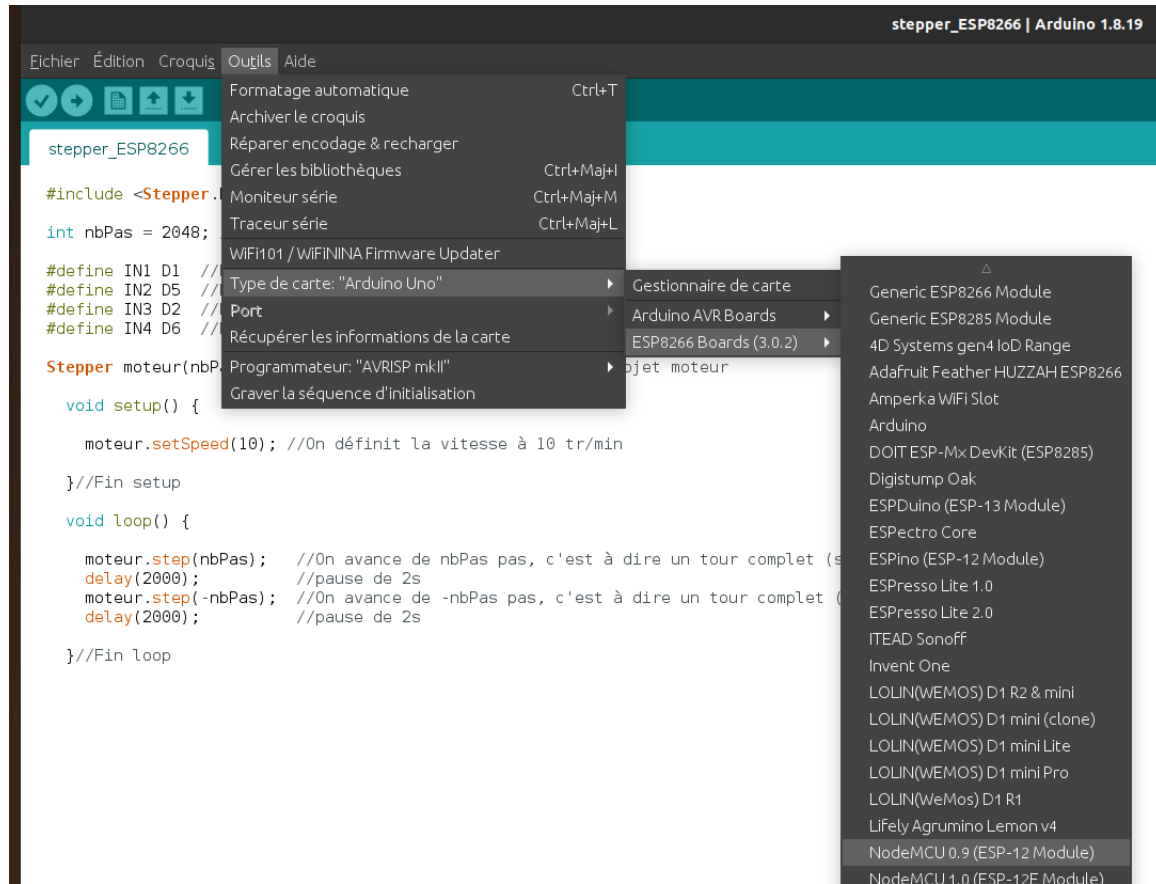


FIGURE A.6 – Sélection de la carte ESP12

Afin de tester le bon fonctionnement, nous vous invitons à tester le programme **Blink** disponible dans les exemples.

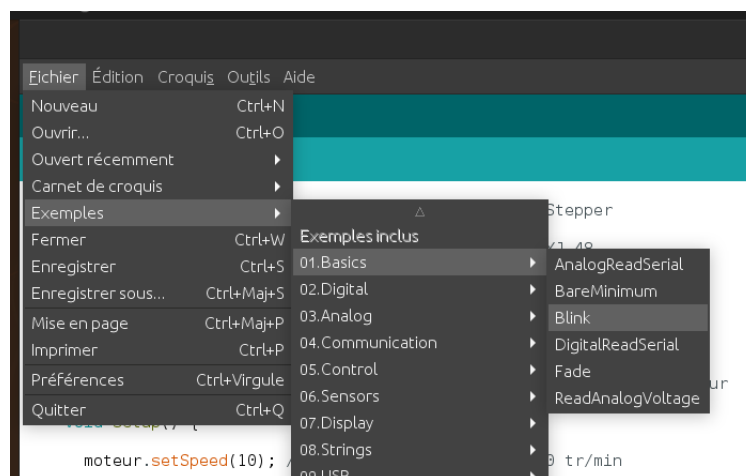


FIGURE A.7 – Emplacement de l'exemple Blink

La led bleue de l'ESP12 devrait clignoter si l'installation s'est correctement effectuée.

Recherche des cartes Arduino

Pour la programmation des cartes Arduino, il suffira de sélectionner

KEY Outils - Type de carte - Arduino AVR Boards - Carte X en fonction du modèle de votre carte.

ANNEXE B

LANGUAGE HTML

Le HTML¹ est un langage contenant des balises, c'est à dire des marqueurs spécifiques pour organiser la page Web.

Il existe deux types principaux de balises :

- ▶ Les balises en paire (Par exemple `< h1 >< /h1 >`)
- ▶ Les balises orphelines (Par exemple `< img >`)

Une balise commence par un chevron ouvrant et se termine par un chevron fermant. Toutes les balises fermantes (pour les balises en paire) sont de la forme `< /nom_balise >`

Toute page HTML commencer par la balise `<html>`

Remarque

Pour mettre du code en commentaire, c'est à dire ne pas le visualiser dans la page Web de rendu, il faut mettre le code entre `<!-->` et `-->`

```
<html>
  <!-- Ceci est mon début de page HTML -->
</html>
```

Première balise HTML

La forme de la page

La page Web est scindée en 2 entités :

- ▶ L'en-tête (header), marqué avec la balise `< head >< /head >`
- ▶ Le corps (body), marqué avec la balise `< body >< /body >`

1. HyperText Markup Language

```
<html>
  <head>
    <!-- Ceci est un header -->
  </head>

  <body>
    <!--! Ceci est un body -->
  </body>
</html>
```

Page minimaliste HTML

L'en-tête

L'en-tête va contenir les informations et les paramètres de la page, notamment :

- ▶ Le titre de la page
- ▶ Les importations des bibliothèques (feuilles de style)
- ▶ Les icônes
- ▶ L'encodage de la page (UTF-8)

```
<head>
  <!--! Titre en haut de la page -->
  <title>Titre de la page</title>

  <!--! Ajout d'une feuille de style -->
  <link rel="stylesheet" href="style.css" type="text/css">

  <!--! Ajout d'une icone -->
  <link rel="icon" href="icone.ico">

  <!--! Encodage UTF-8 -->
  <meta charset="utf-8">

</head>
```

L'en-tête

Le corps

Le corps va contenir l'ensemble des informations affichées sur la page

- ▶ Les titres, sous-titre, sous-sous-titre
- ▶ Les paragraphes
- ▶ Les images
- ▶ Les liens
- ▶ Les sections de code
- ▶ ...

Ajout des titres

Un titre en HTML est vue avec un niveau hierarchique. Un titre de page est au niveau 1, un sous-titre avec un niveau 2, etc...

Pour mettre un titre, il faut donc écrire `< h1 > Titre < /h1 >`, un sous-titre c'est

`< h2 > Sous – titre < /h2 >`

Ajout des images

Pour ajouter une image, il faut connaître son emplacement dans le système (ordinateur) ou bien sur internet (url).

Il s'agit de la balise orpheline `< img >`

```
<!--! Image locale -->


<!--! Image Internet -->
<img url="www.crepp.oreg/image.png" alt="Impossible de charger l'image">
```

Ajout d'une image

Ajout des liens

Pour ajouter une image, il faut connaître son emplacement dans le système (ordinateur) ou bien sur internet (url).

Il s'agit de la balise orpheline `< img >`

```
<!--! Lien -->  
<a href="monChemin" > Texte du lien</a>
```

Ajout d'un lien

Les liens permettent de pointer sur d'autres pages, que ce soit sur le serveur courant ou bien un autre.

Table des figures

2.1	Un réseau plus évolué	9
2.2	La commande ipconfig	10
2.3	La commande ipconfig	11
3.1	La led interne de l'ESP	13
3.2	Architecture du projet	13
3.3	Architecture client-serveur	14
3.4	L'adresse avec la requête GET	15
3.5	Sélection du mode Série	16
3.6	Affichage de l'adresse IP	16
3.7	Connexion au serveur	17
3.8	Résultat	17
A.1	ESP12 NodeMCU	25
A.2	Préférences Arduino	26
A.3	Lien pour les cartes ESP8266	27
A.4	Gestionnaire des cartes ESP8266	28
A.5	Installation des bibliothèques ESP8266	28
A.6	Sélection de la carte ESP12	29
A.7	Emplacement de l'exemple Blink	29