



Ateliers CREPP

Les ESP8266

Nicolas Le Guerroué

6 février 2022
26 pages

Table des matières

Page

1	Préambule	3
I	Mise en place d'un serveur ESP12	4
2	Un serveur Web avec ESP12	5
2.1	Architecture du mini-projet	5
2.2	Base des requêtes	6
2.2.1	Une petite explication sur les ports	6
2.2.2	Les types de requêtes	7
2.3	Connexion au routeur	7
2.4	Lancement du programme	7
2.5	Explication du programme	9
2.5.1	En tête du code	9
2.5.2	Fonction setup	11
2.5.3	Fonction loop	12
2.6	Code complet	12
II	Annexes	16
A	Utilisation de l'ESP12 sous Arduino	17
A.1	Installation des bibliothèques et cartes ESP8266	17
A.2	Recherche des cartes ESP8266	20
A.3	Recherche des cartes Arduino	22
B	Langage HTML	23
B.1	La forme de la page	23
B.2	L'en-tête	24
B.3	Le corps	25
B.3.1	Ajout des titres	25
B.3.2	Ajout des images	25
B.3.3	Ajout des liens	25

SECTION 1

PRÉAMBULE

- ▶ Document réalisé en \LaTeX 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 6 février 2022
- ▶ Taille de police : 11pt
- ☎ 06.20.88.75.12
- ✉ nicolasleguerroue@gmail.com
- ▶ 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

Première 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 2

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)

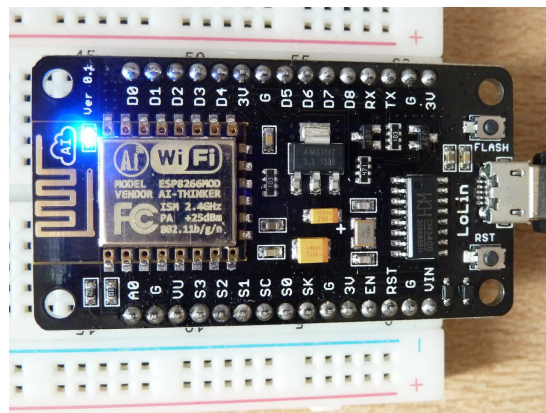


FIGURE 2.1 – La led interne de l'ESP

Architecture du mini-projet

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

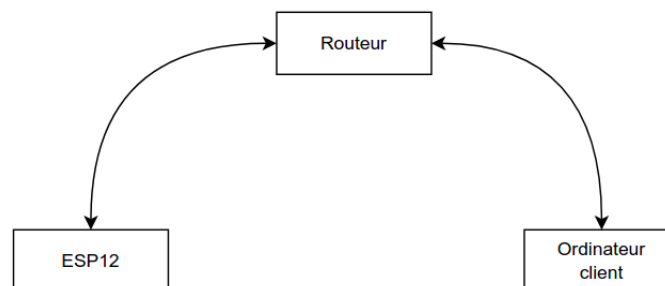


FIGURE 2.2 – Architecture du projet

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.

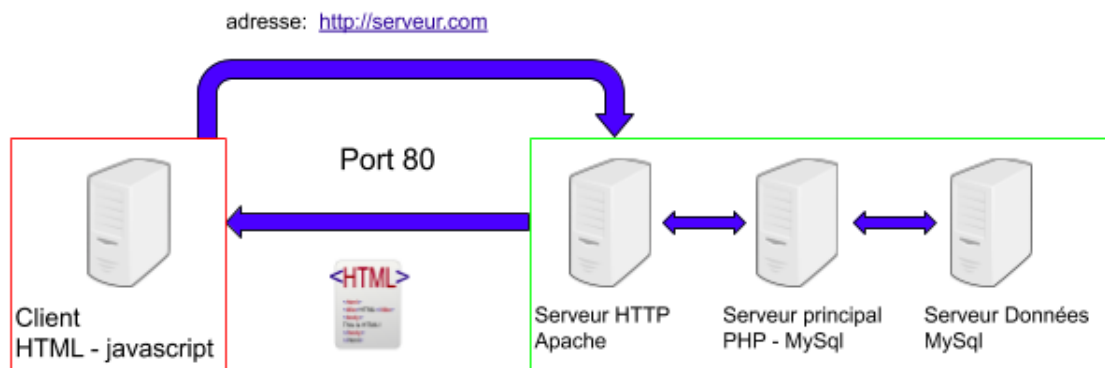


FIGURE 2.3 – Architecture client-serveur

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 :

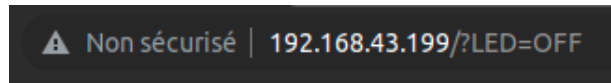


FIGURE 2.4 – L'adresse avec la requête GET

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

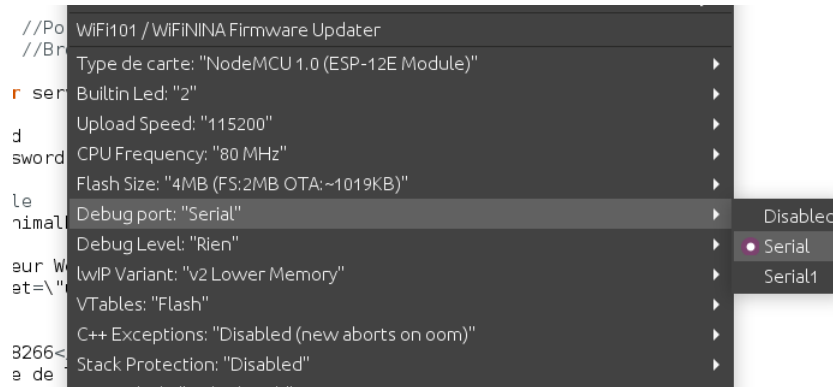


FIGURE 2.5 – Sélection du mode Série

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.

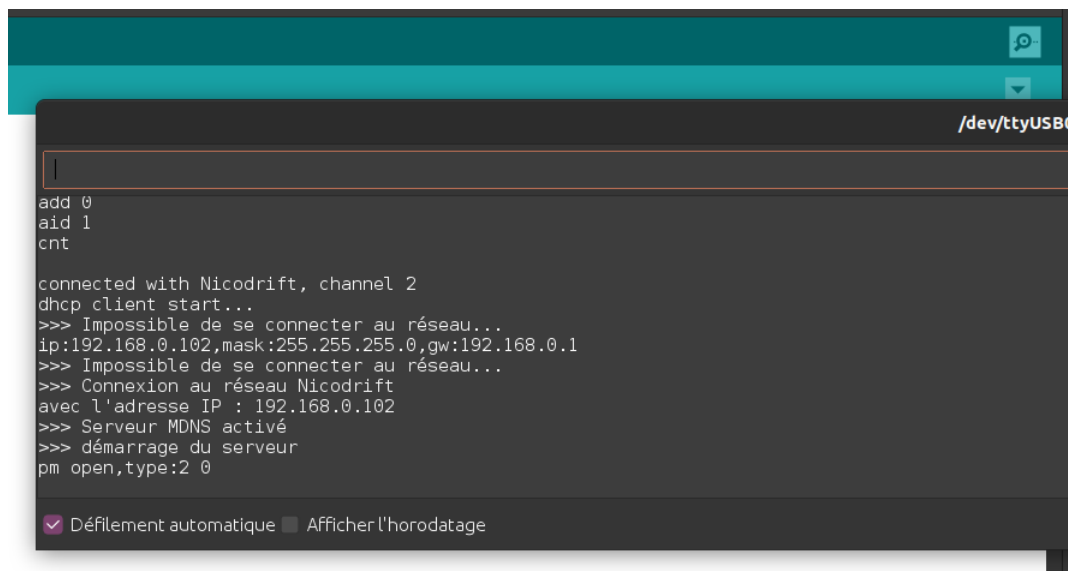


FIGURE 2.6 – Affichage de l'adresse IP

Si aucune données n'apparait, 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`

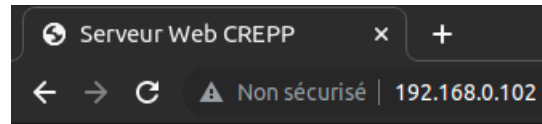


FIGURE 2.7 – Connexion au serveur

Le résultat doit être le suivant

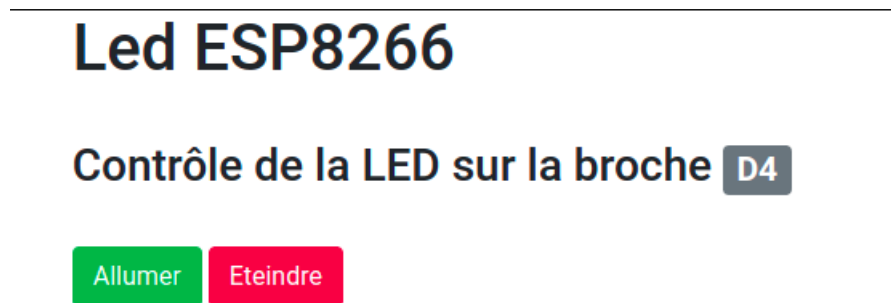


FIGURE 2.8 – Résultat

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>\
<head>\
  <title>Serveur Web CREPP</title>\
  <meta charset=\"utf-8\"/> \
</head>\
<body>\
  <h1>Led ESP8266</h1><br>\
  <h3>Contrôle de la LED sur la broche D4</h3><br>\
  <a href=\"//?LED=ON\"><button >Allumer</button></a>\
  <a href=\"//?LED=OFF\"><button >Eteindre</button></a>\
</body>\
</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>\
<head>\
  <title>Serveur Web CREPP</title>\
  <meta charset=\"utf-8\"/> \
  <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\">\
</head>\
<body style=\"margin-left:5%;\">\
  <h1>Led ESP8266</h1><br>\
  <h3>Contrôle de la LED sur la broche <span class=\"badge badge-secondary\">D4</span></h3><br>\
  <a href=\"//?LED=ON\"><button class=\"btn btn-success\">\
Allumer</button></a>\
  <a href=\"//?LED=OFF\"><button class=\"btn btn-danger\">\
Eteindre</button></a>\
</body>
```

```
</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

Deuxième partie

Annexes

ANNEXE A

UTILISATION DE L'ESP12 SOUS ARDUINO

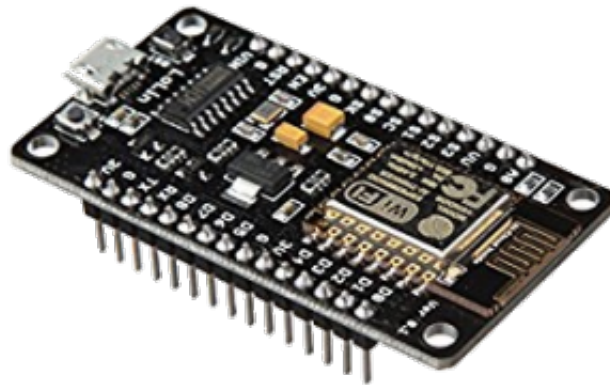


FIGURE A.1 – ESP12 NodeMCU

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

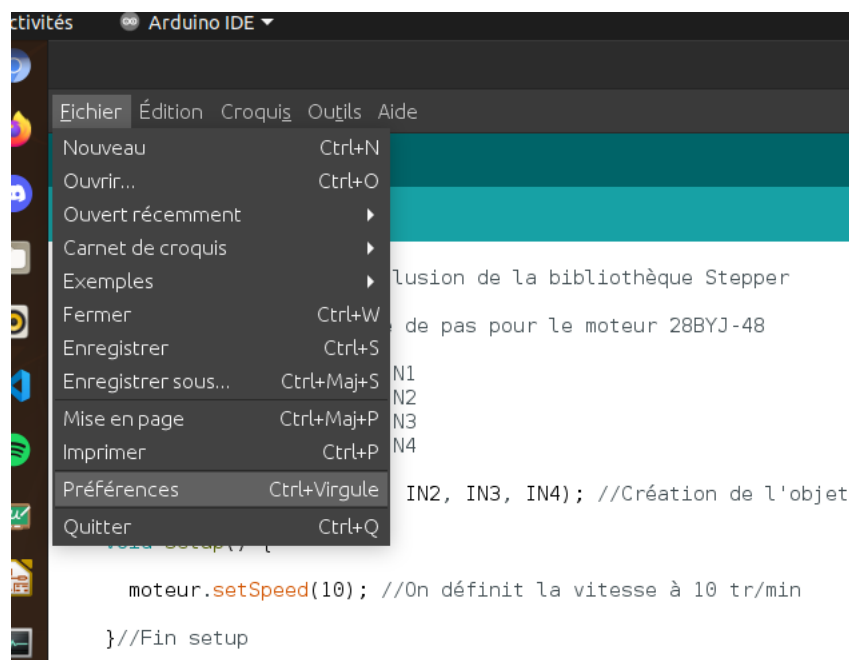


FIGURE A.2 – Préférences Arduino

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

URL [http ://arduino.esp8266.com/stable/package_esp8266com_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

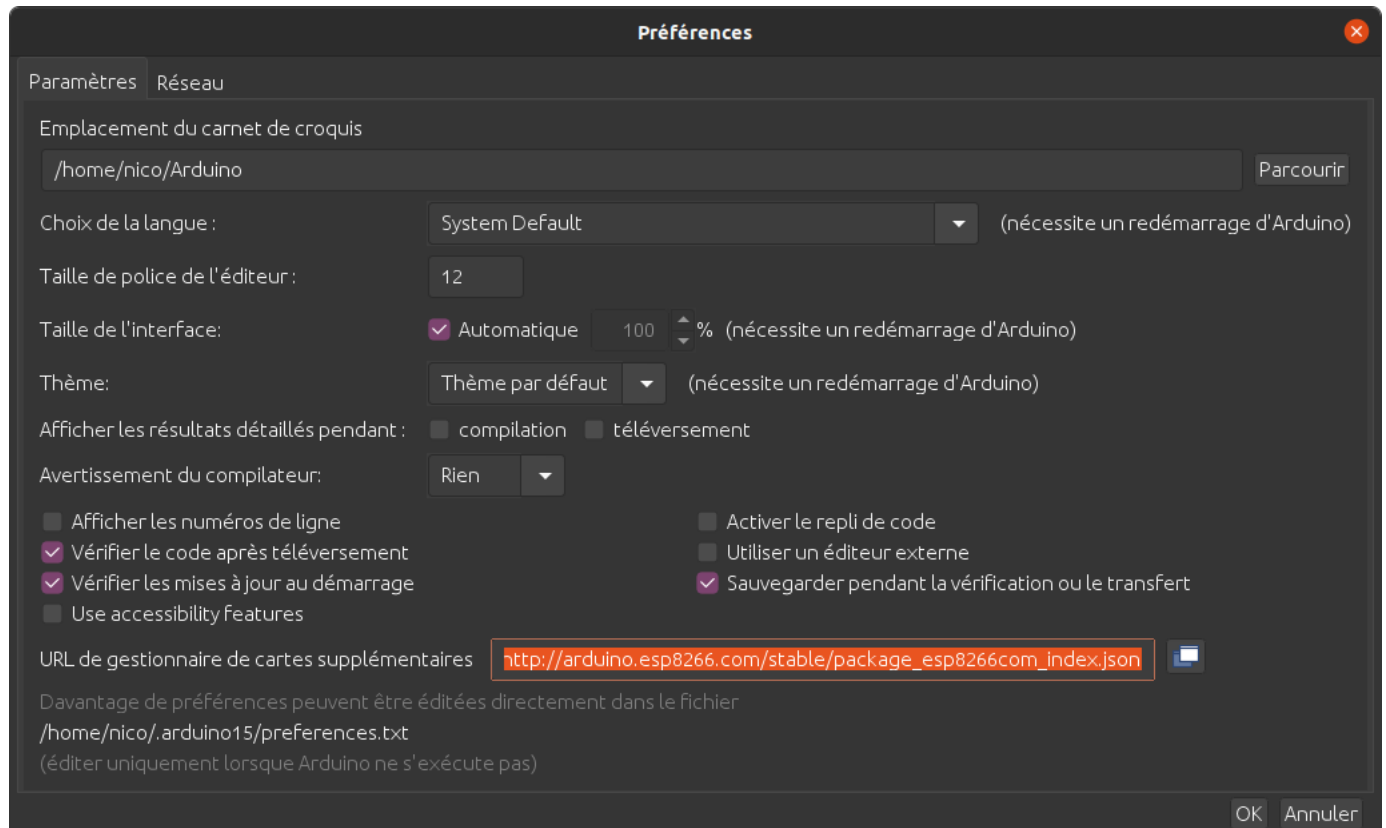



FIGURE A.3 – Lien pour les cartes ESP8266

Puis faire  OK

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

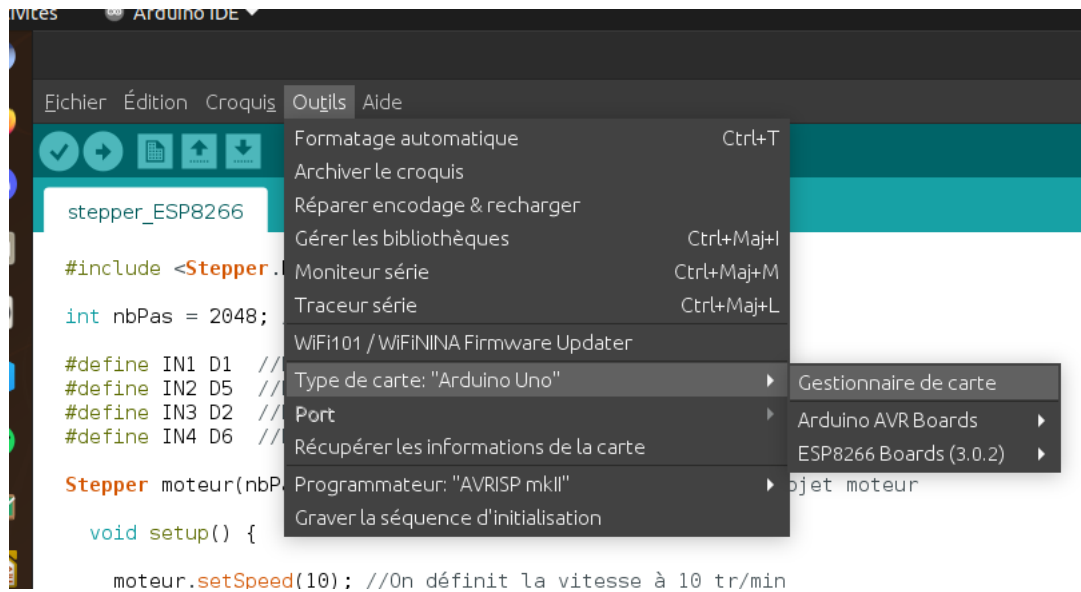


FIGURE A.4 – Gestionnaire des cartes ESP8266

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)`

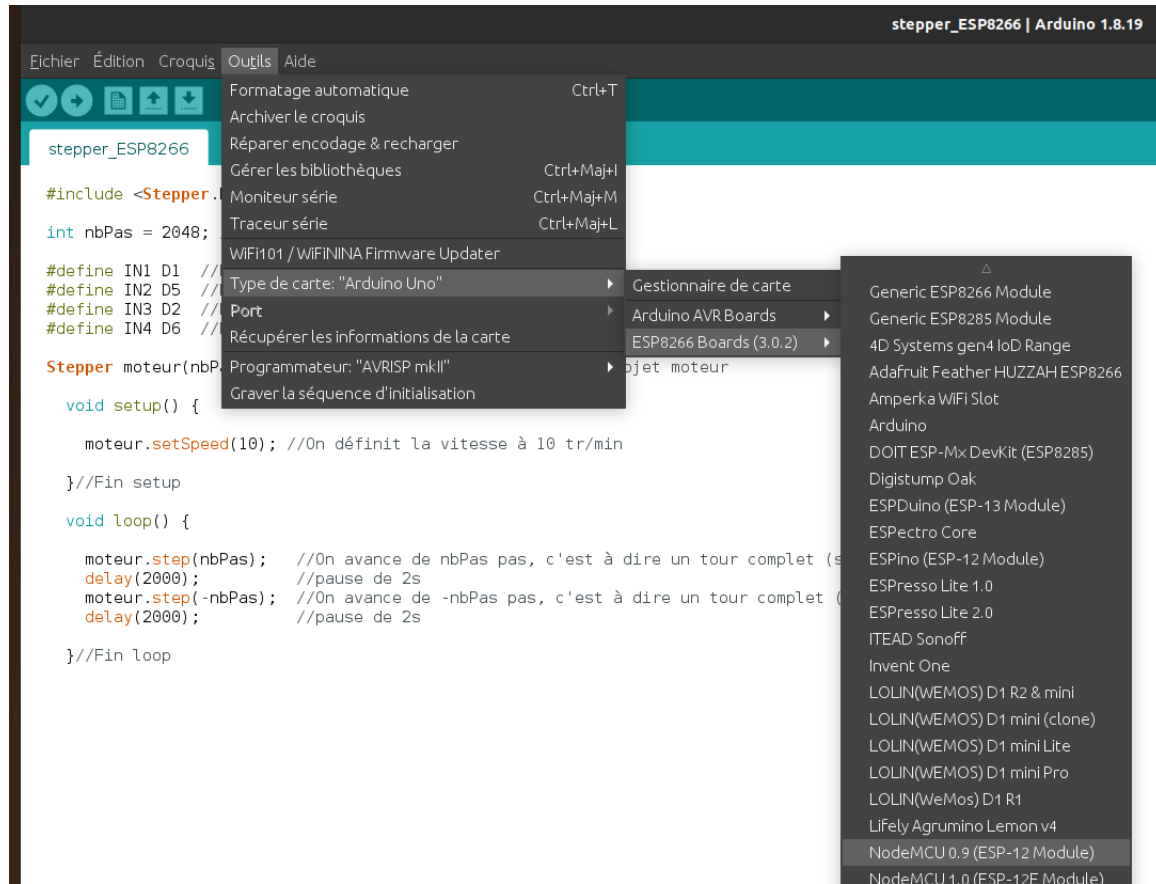


FIGURE A.6 – Sélection de la carte ESP12

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

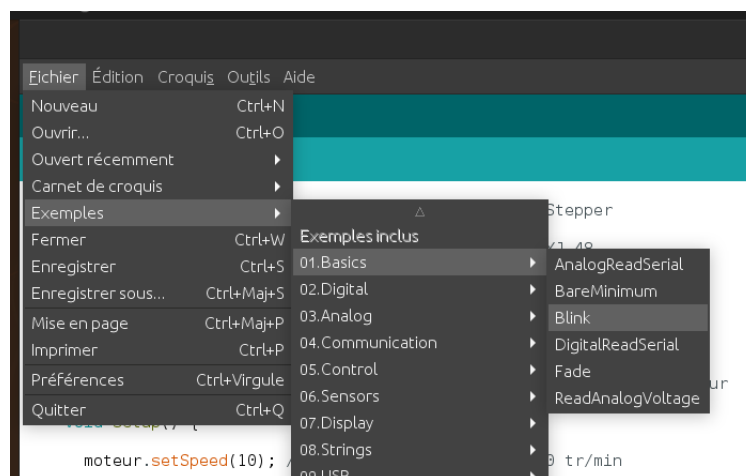


FIGURE A.7 – Emplacement de l'exemple Blink

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.

Table des figures

2.1	La led interne de l'ESP	5
2.2	Architecture du projet	5
2.3	Architecture client-serveur	6
2.4	L'adresse avec la requête GET	7
2.5	Sélection du mode Série	8
2.6	Affichage de l'adresse IP	8
2.7	Connexion au serveur	9
2.8	Résultat	9
A.1	ESP12 NodeMCU	17
A.2	Préférences Arduino	18
A.3	Lien pour les cartes ESP8266	19
A.4	Gestionnaire des cartes ESP8266	20
A.5	Installation des bibliothèques ESP8266	20
A.6	Sélection de la carte ESP12	21
A.7	Emplacement de l'exemple Blink	21

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.