



Ateliers CREPP

Atelier Moteurs et ESP8266

Club de Robotique et d'Electronique
Programmable de Ploemeur

22 février 2022
45 pages

Table des matières

	Page
1 Préambule	4
I Les moteurs pas-à-pas	5
2 Présentation	6
2.0.1 Constitution	6
2.0.2 Les types de moteur pas-à-pas	6
2.0.3 Principe	7
3 Commande des moteurs pas-à-pas	12
3.1 Moteurs unipolaires	12
3.2 Moteurs bipolaires	13
3.2.1 Avantages et inconvénients des moteurs pas-à-pas	15
3.2.2 Domaines d'application	15
3.3 Comment distinguer les différents types de moteurs?	15
4 Exemples	16
4.1 Mise en pratique avec Arduino	16
4.1.1 Liste du matériel	17
4.1.2 Branchements	17
4.1.3 Code Arduino	18
4.2 Mise en pratique avec ESP8266	19
4.2.1 Liste du matériel	19
4.2.2 Branchements	20
4.2.3 Code ESP12	21
II Mise en place d'un serveur ESP12	23
5 Un serveur Web avec ESP12	24
5.1 Architecture du mini-projet	24
5.2 Base des requêtes	25
5.2.1 Une petite explication sur les ports	25
5.2.2 Les types de requêtes	26
5.3 Connexion au routeur	26

5.4	Lancement du programme	26
5.5	Explication du programme	28
5.5.1	En tête du code	28
5.5.2	Fonction setup	30
5.5.3	Fonction loop	31
5.6	Code complet	31
 III Annexes		35
 A Utilisation de l'ESP12 sous Arduino		36
A.1	Installation des bibliothèques et cartes ESP8266	36
A.2	Recherche des cartes ESP8266	39
A.3	Recherche des cartes Arduino	41
 B Langage HTML		42
B.1	La forme de la page	42
B.2	L'en-tête	43
B.3	Le corps	44
B.3.1	Ajout des titres	44
B.3.2	Ajout des images	44
B.3.3	Ajout des liens	44

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 22 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

Les moteurs pas-à-pas



Théorie sur les moteurs pas-à-pas et applications pratiques avec Arduino et ESP8266

SECTION 2

PRÉSENTATION

Les moteur pas-à-pas sont utilisés lorsqu'on souhaite un asservissement en position d'un axe de rotation avec une précision inégalée par les servomoteurs

Constitution

Les moteur pas-à-pas sont constitués de :

- ▶ Plusieurs bobines (un pôle forme une paire de bobines)
- ▶ Un aimant qui sert d'axe de rotation

Les types de moteur pas-à-pas

- ▶ moteur pas-à-pas à phase bipolaire

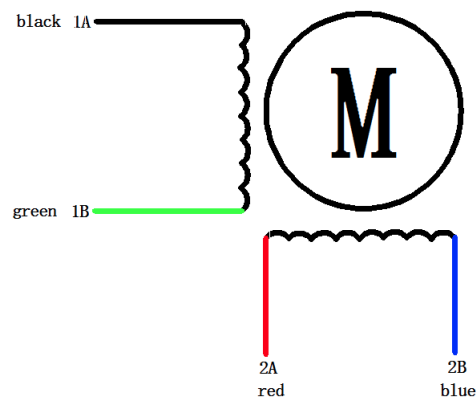


FIGURE 2.1 – moteur pas-à-pas bipolaire

- ▶ moteur pas-à-pas à phases unipolaires

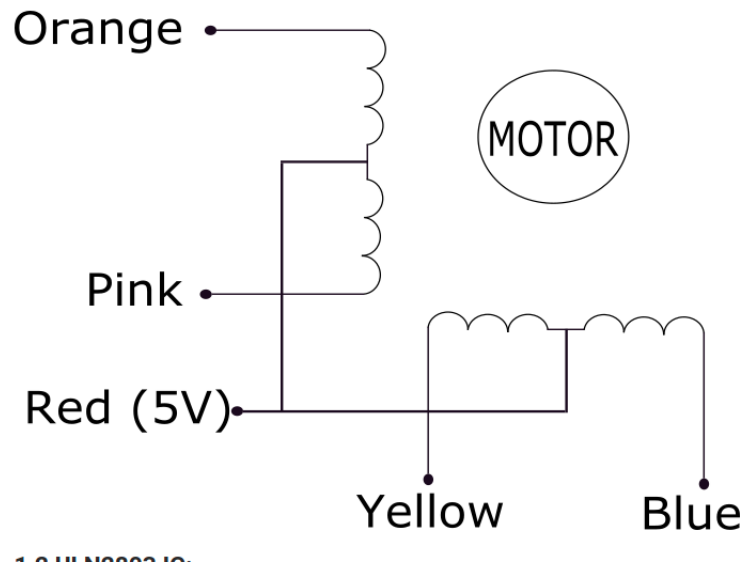


FIGURE 2.2 – moteur pas-à-pas unipolaires

- moteur pas-à-pas à reluctance variable (non abordés ici)

Principe

En faisant varier le champ électromagnétique des différentes phases, on peut faire varier la position angulaire de l'aimant.

Exemples avec phases bipolaires

En alimentant une paire de phases avec une tension positif, l'aimant se place dans l'alignement du champ électromagnétique formé par la paire de phase.

En alimentant la paire de phase avec une tension négative, l'aimant se place dans le sens contraire.

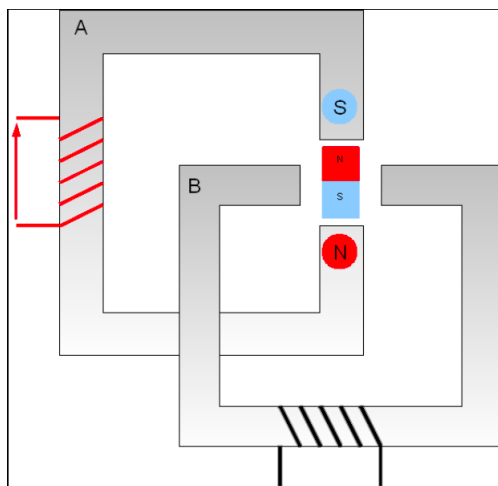


FIGURE 2.3 – Pas 1

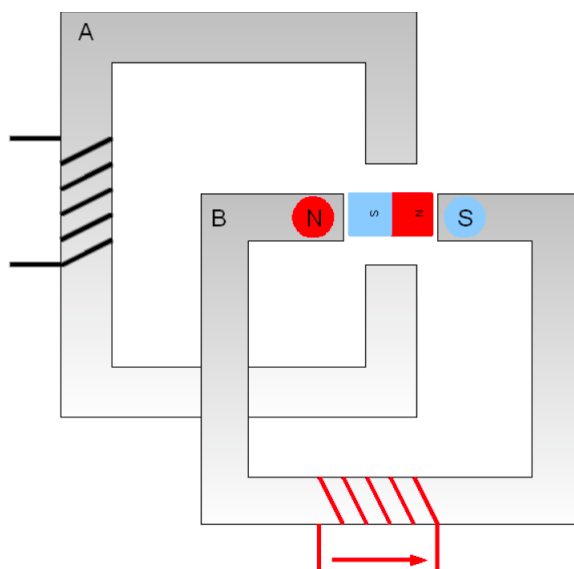


FIGURE 2.4 – Pas 2

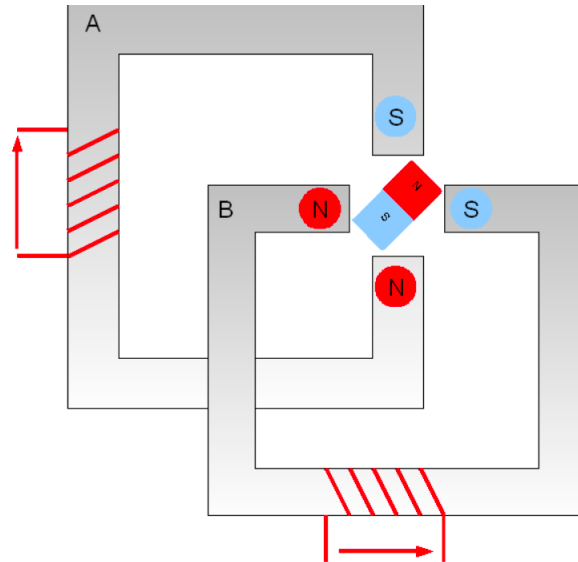


FIGURE 2.5 – Pas 3

Exemples avec phases unipolaires

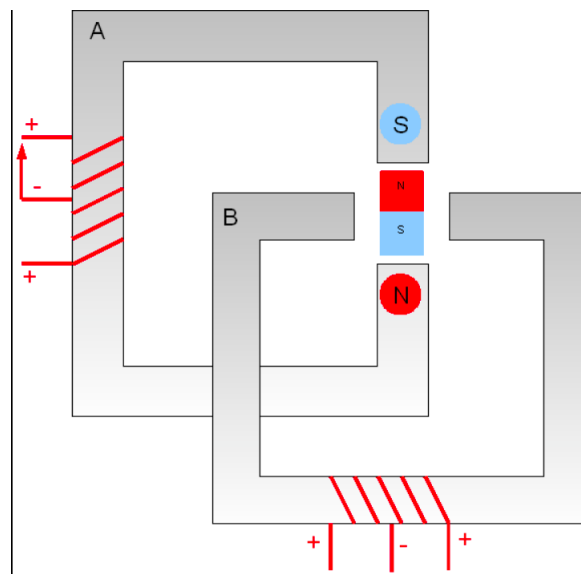


FIGURE 2.6 – Pas 1

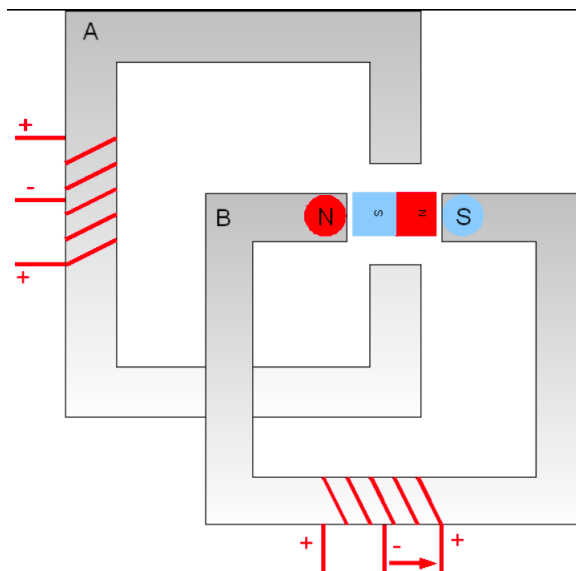


FIGURE 2.7 – Pas 2

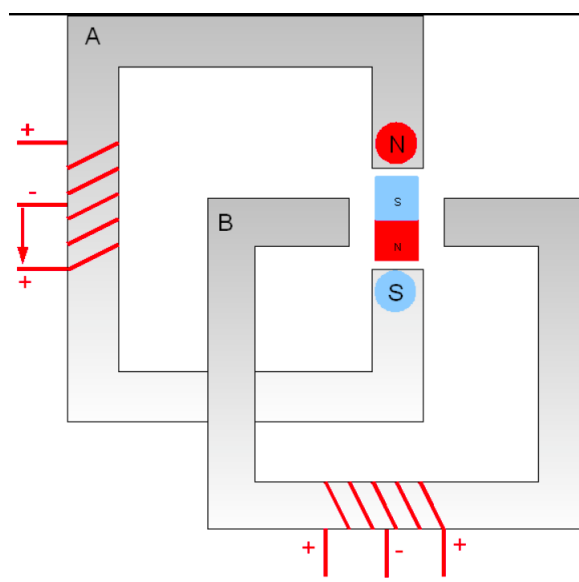


FIGURE 2.8 – Pas 3

Les moteurs possèdent plus de phases car un débattement de 45° est vite limité.

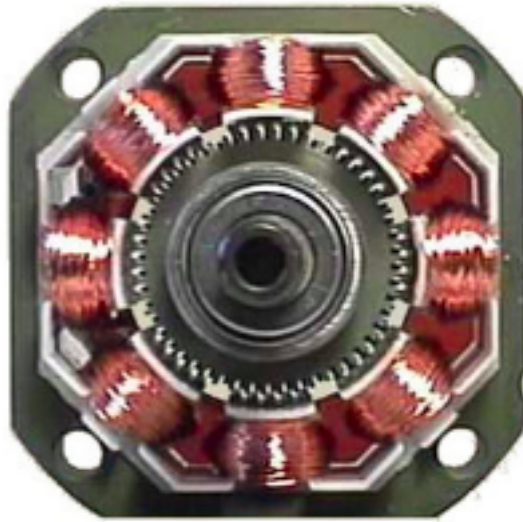


FIGURE 2.9 – Un intérieur de moteur

Les moteur pas-à-pas unipolaires présentent l'avantage de faire circuler un courant positif dans le circuit de commande. Ils sont donc plus simples à mettre en oeuvre mais ils nécessitent plus de bobinage.

SECTION 3

COMMANDE DES MOTEURS PAS-À-PAS

Moteurs unipolaires

Les moteur pas-à-pas unipolaires ont besoins d'être contrôlés via un circuit adaptés, le plus connu est le REF ULN2803



FIGURE 3.1 – Driver ULN2803

Pour les moteurs unipolaires, il faut mettre une des phases à la masse pour faire circuler le courant dans la phase¹

On constate sur la figure suivante un montage Darlington : deux transistors NPN forme un seul transistor dont le coefficient β est le produit de chacun des coefficients β de chaque transistor.

Cela permet de contrôler des charges avec très peu de courant de commande.²

-
1. Due au câble relié à l'alimentation positive du moteur
 2. Se référer à la partie **Circuits de puissance**

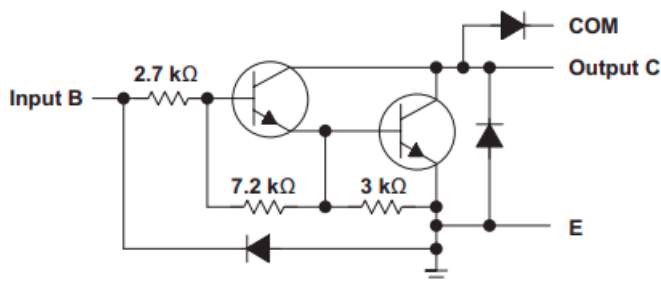


FIGURE 3.2 – Contrôle d'une phase

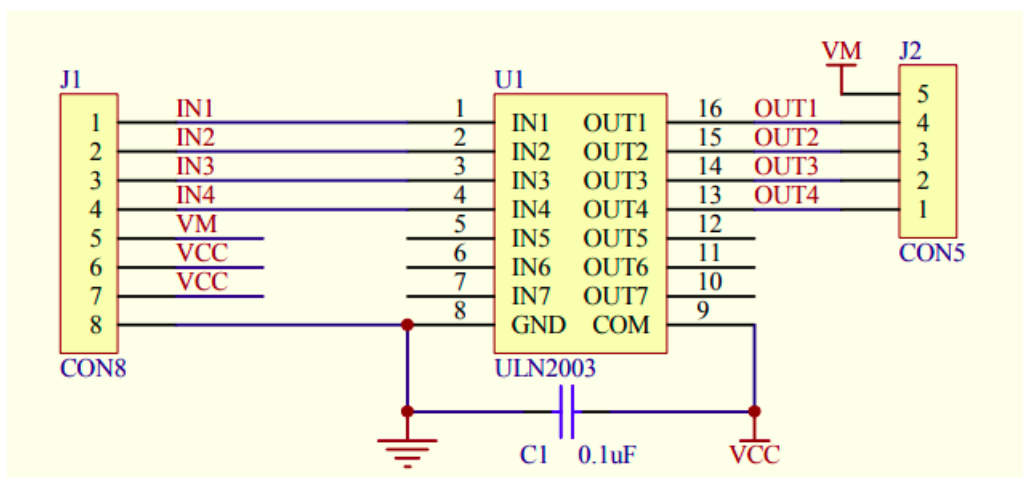


FIGURE 3.3 – Driver de controle

Moteurs bipolaires

On a vu qu'il fallait inverser la tension de commande au borne des bobines. Pour cela on peut utiliser le montage en pont en H.

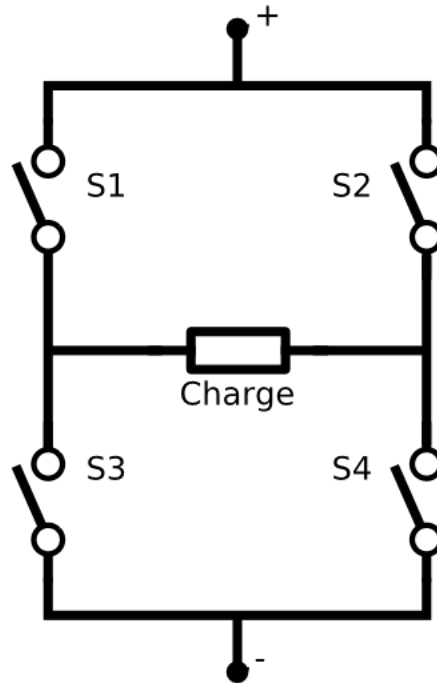


FIGURE 3.4 – Structure du pont en H

- En activant S1 et S4 (fermeture du circuit), la charge est parcourue par un courant allant de gauche à droite
- En activant S2 et S3 (fermeture du circuit), la charge est parcourue par un courant allant de droite à gauche

Et qui dit inversion de courant dit inversion de tension. Notre objectif est atteint, nous pouvons mettre des tensions positives et négatives aux bornes des phases de nos moteurs.

Ce principe est également utilisé pour contrôler les moteurs à courant continu
On peut utiliser le circuit REF L298

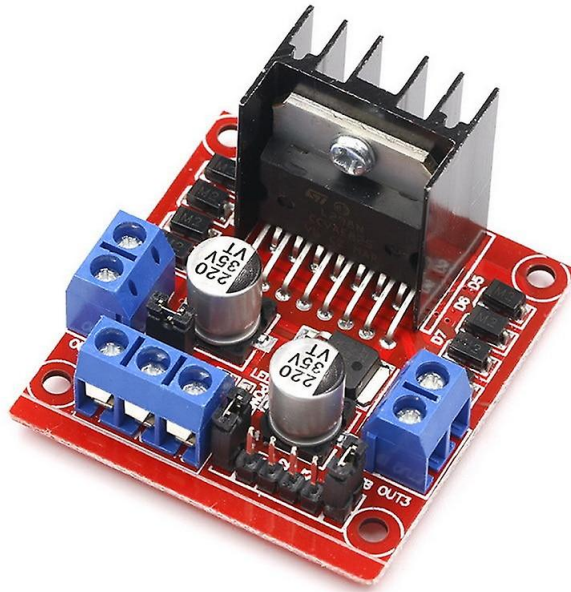


FIGURE 3.5 – Un pont en H intégré

Avantages et inconvénients des moteurs pas-à-pas

- Très grande précision en boucle ouverte³
- Couple élevé en bas régime
- Plus lent qu'un servomoteur
- Complexité de mise en oeuvre

Domaines d'application

- Imprimantes
- Machines CNC

Comment distinguer les différents types de moteurs ?

- 2 fils = moteur à courant continue
- 3 fils = servomoteurs
- 4 fils = moteur pas-à-pas bipolaire
- 5 fils = moteur pas-à-pas unipolaire

3. Contrôle sans asservissement, contrairement aux servomoteurs

SECTION 4

EXEMPLES

Mise en pratique avec Arduino

Nous utiliserons un moteur pas-à-pas **28BYJ-48** de type unipolaire.

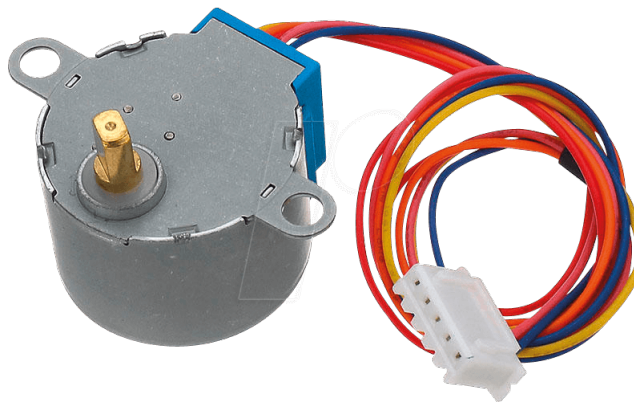


FIGURE 4.1 – Le moteur 28BYJ-48

Les caractéristiques sont les suivantes :

- Nombre de pas : 2048
- Tension d'alimentation : 5V

Pour augmenter le nombre de pas, on ajoute un train d'engrenage.

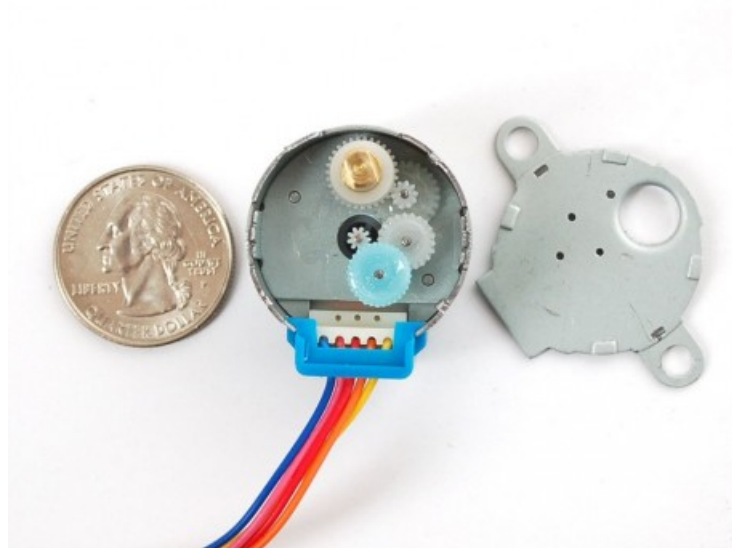


FIGURE 4.2 – Une augmentation du nombre de pas

Liste du matériel

- Carte Arduino Uno
- Driver ULN2803
- Moteur pas-à-pas 28BYJ-48 ou équivalent
- Câbles

Branchements

Nous utiliserons les broches 8, 9, 10 et 11 et l'alimentation 5V du moteur sera fournie par la broche +5V de l'Arduino

- D1 sur IN1
- D2 sur IN3
- D5 sur IN2
- D6 sur IN4
- Vin sur Vcc

- Gnd sur Gnd

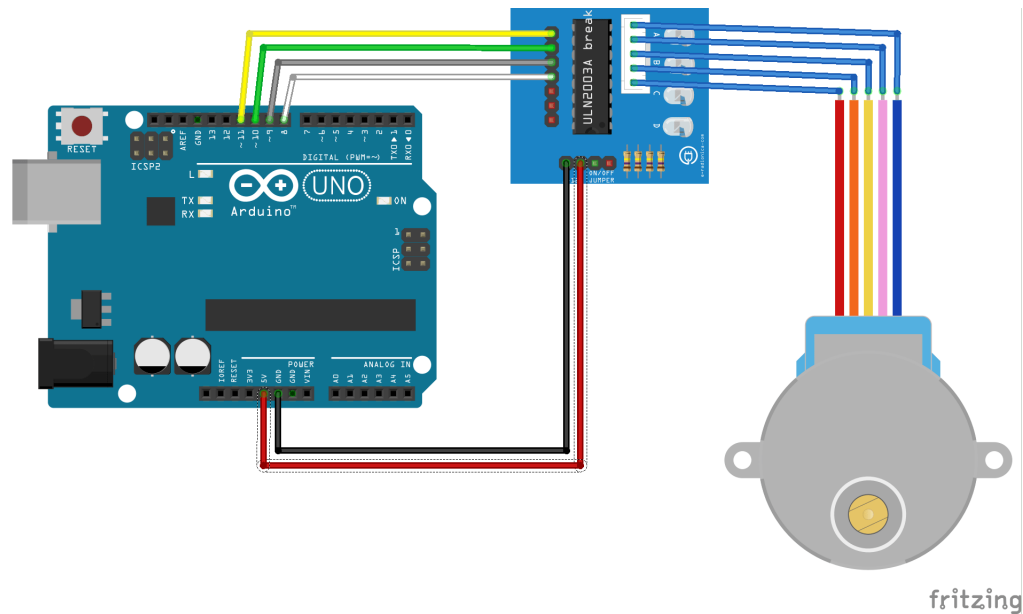


FIGURE 4.3 – Schéma Arduino

Code Arduino

Ce code fait tourner le moteur d'un tour, attend 2 secondes puis fait un tour dans l'autre sens avec un délai de 2s.

```
#include <Stepper.h> //Inclusion de la bibliothèque Stepper

int nbPas = 2048; //Nombre de pas pour le moteur 28BYJ-48

#define IN1 8 //Broche IN1
#define IN2 9 //Broche IN2
#define IN3 10 //Broche IN3
#define IN4 11 //Broche IN4

Stepper moteur(nbPas, IN1, IN3, IN2, IN4); //Création de l'objet moteur

void setup() {

    moteur.setSpeed(10); //On définit la vitesse à 10 tr/min

} //Fin setup

void loop() {
```

```
moteur.step(nbPas); //On avance de nbPas pas, c'est à dire un tour complet
(sens horaire)
delay(2000);        //pause de 2s
moteur.step(-nbPas); //On avance de -nbPas pas, c'est à dire un tour complet
(sens anti-horaire)
delay(2000);        //pause de 2s

} //Fin loop
```

Code minimaliste Arduino

Mise en pratique avec ESP8266

Nous utiliserons le même moteur pas-à-pas 28BYJ-48

Liste du matériel

- Carte ESP8266 NodeMCU (ESP-12)



FIGURE 4.4 – ESP12 NodeMCU

Cette carte fait partie de la famille des ESP8266 et se programme directement avec l'Editeur Arduino. L'installation des bibliothèques pour l'ESP12 est détaillée en annexe du document.

- Driver ULN2803
- Moteur pas-à-pas 28BYJ-48 ou équivalent (moteur pas-à-pas unipolaire)
- Câbles

Branchements

Les numéros des broches sont différents sur les cartes ESP812 (modèle NodeMCU). Voici les équivalences des broches entre le code et l'emplacement physique.

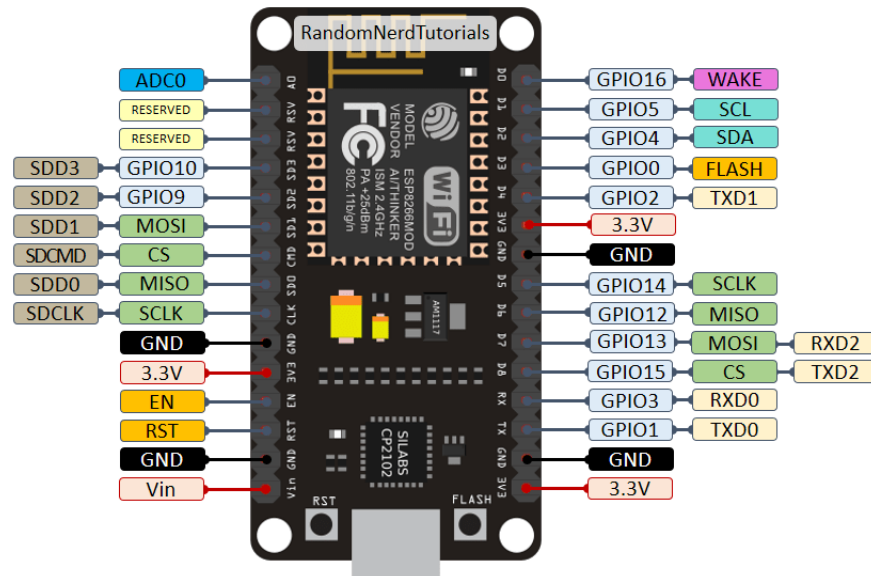


FIGURE 4.5 – Broches ESP12

Nous utiliserons les broches D1, D2, D5 et D6 et l'alimentation 5V du moteur sera fournie par la broche Vin de l'ESP12

- D1 sur IN1
- D2 sur IN3
- D5 sur IN2
- D6 sur IN4
- Vin sur Vcc
- Gnd sur Gnd

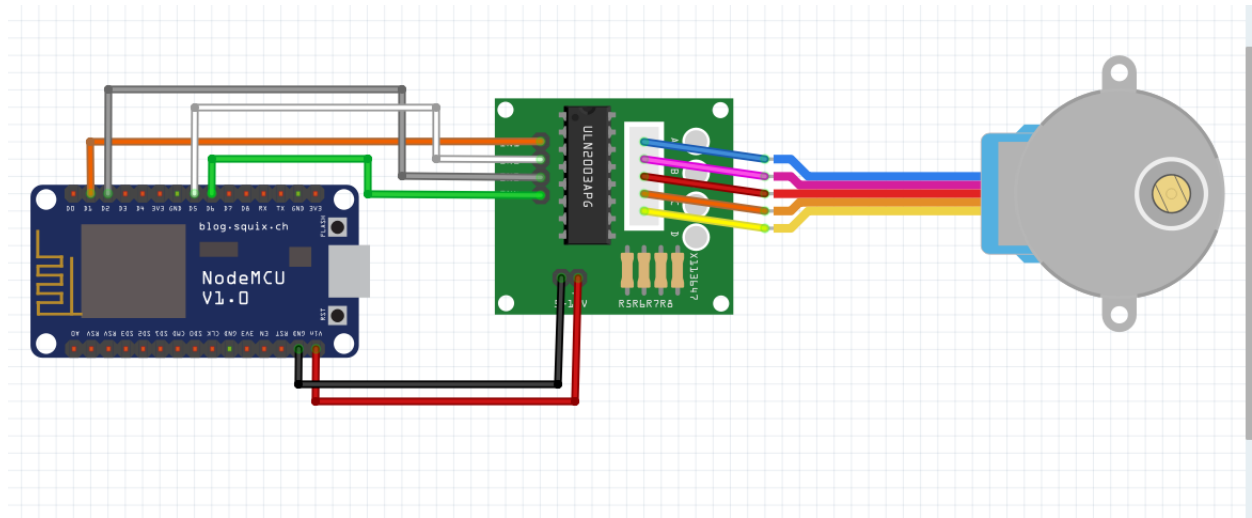


FIGURE 4.6 – Schéma ESP12

Code ESP12

Ce code fait tourner le moteur d'un tour, attend 2 secondes puis fait un tour dans l'autre sens avec un délai de 2s.

```
#include <Stepper.h> //Inclusion de la bibliothèque Stepper

int nbPas = 2048; //Nombre de pas pour le moteur 28BYJ-48

#define IN1 D1 //Broche IN1
#define IN2 D5 //Broche IN2
#define IN3 D2 //Broche IN3
#define IN4 D6 //Broche IN4

Stepper moteur(nbPas, IN1, IN3, IN2, IN4); //Création de l'objet moteur

void setup() {

    moteur.setSpeed(10); //On définit la vitesse à 10 tr/min

} //Fin setup

void loop() {

    moteur.step(nbPas); //On avance de nbPas pas, c'est à dire un tour
    complet (sens horaire)
    delay(2000); //pause de 2s
```

```
moteur.step(-nbPas); //On avance de -nbPas pas, c'est à dire un tour
complet (sens anti-horaire)
delay(2000);          //pause de 2s

} //Fin loop
```

Code minimaliste ESP12

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 5

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)

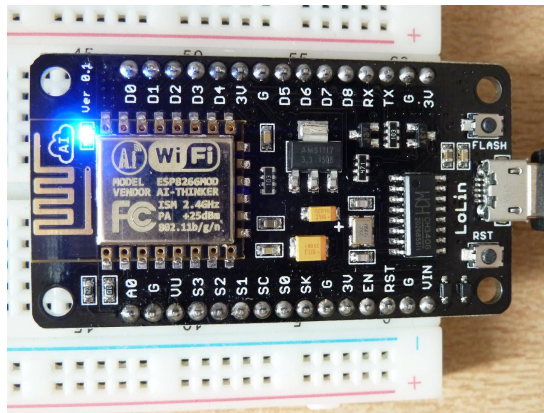


FIGURE 5.1 – La led interne de l'ESP

Architecture du mini-projet

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

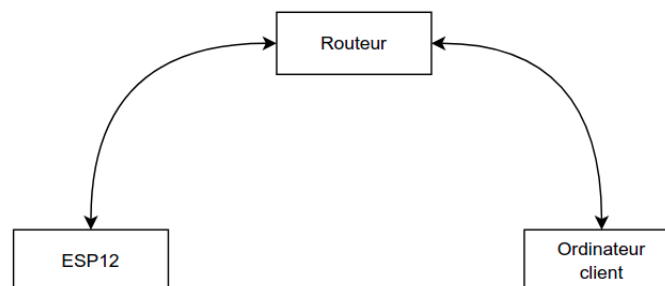


FIGURE 5.2 – Architecture du projet

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.

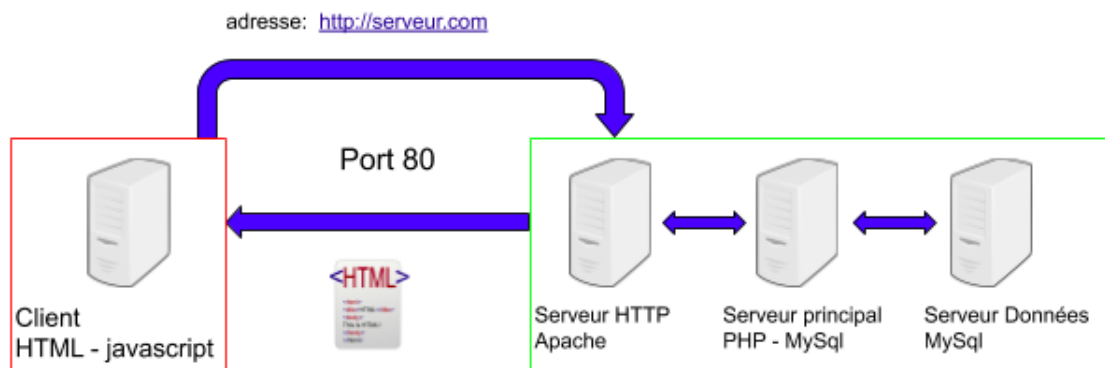


FIGURE 5.3 – Architecture client-serveur

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 :

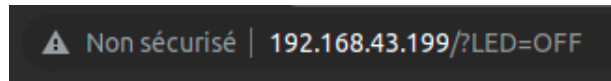


FIGURE 5.4 – L'adresse avec la requête GET

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

Ici, nous avons un argument `LED` avec la valeur `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 `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

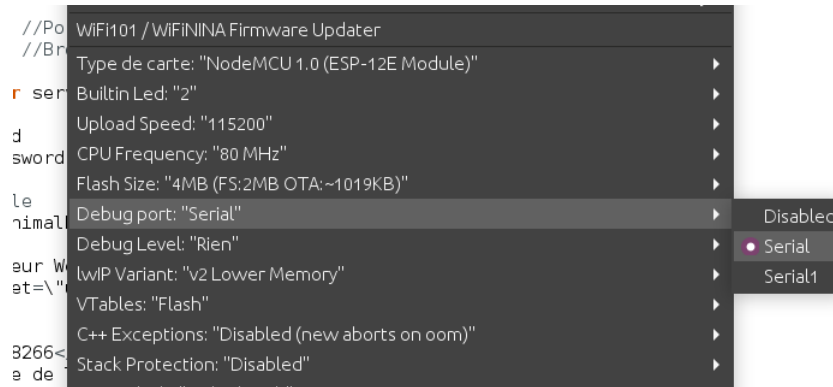


FIGURE 5.5 – Sélection du mode Série

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.

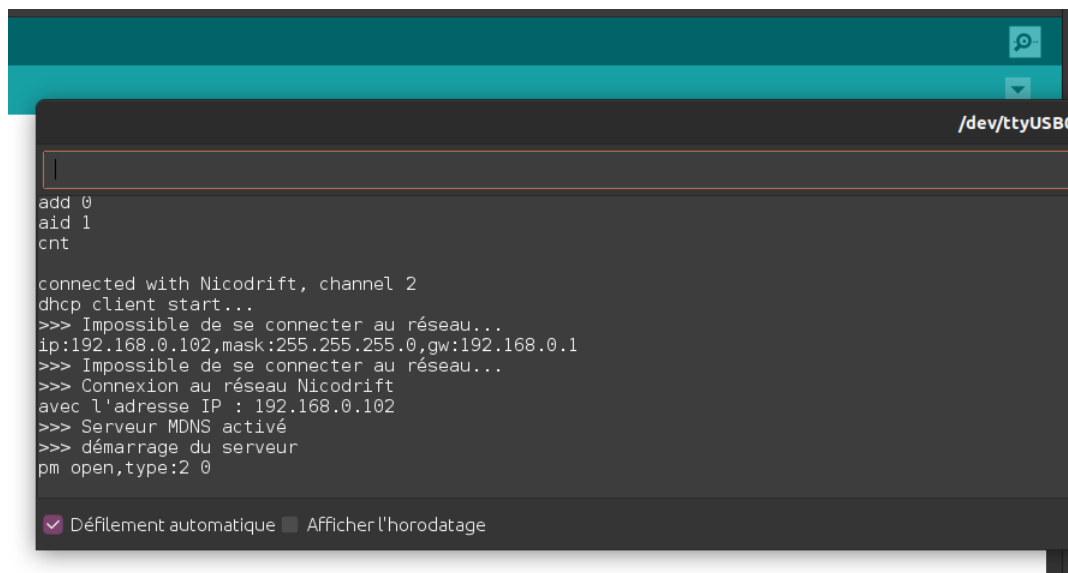


FIGURE 5.6 – Affichage de l'adresse IP

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`

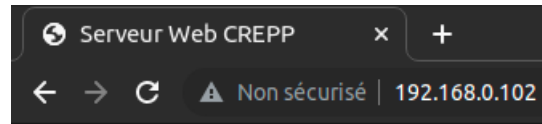


FIGURE 5.7 – Connexion au serveur

Le résultat doit être le suivant

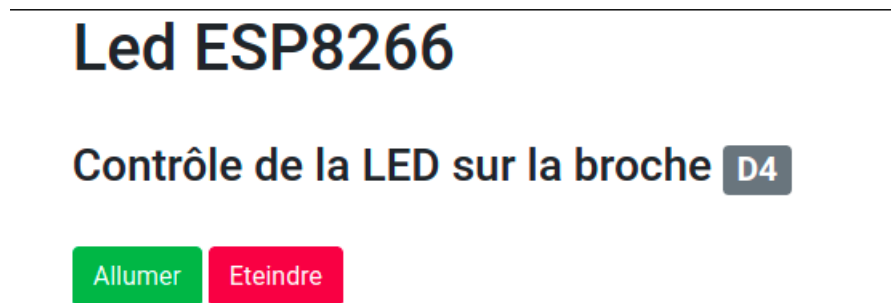


FIGURE 5.8 – Résultat

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 requ
ête 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

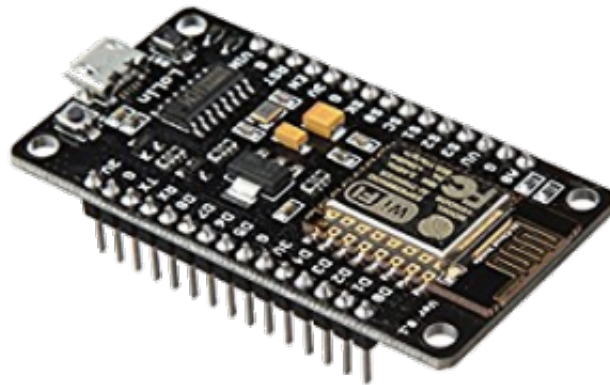


FIGURE A.1 – ESP12 NodeMCU

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

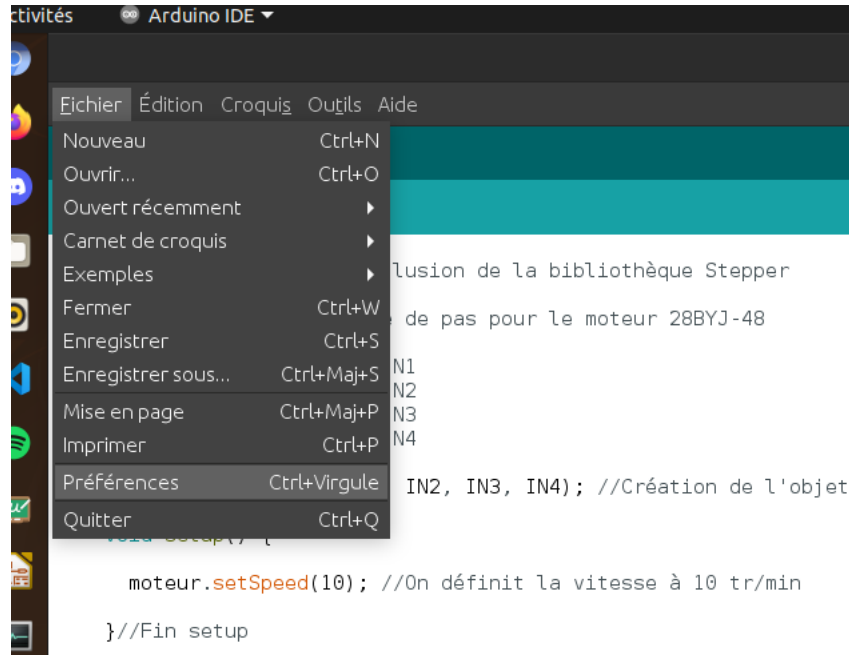


FIGURE A.2 – Préférences Arduino

- 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.

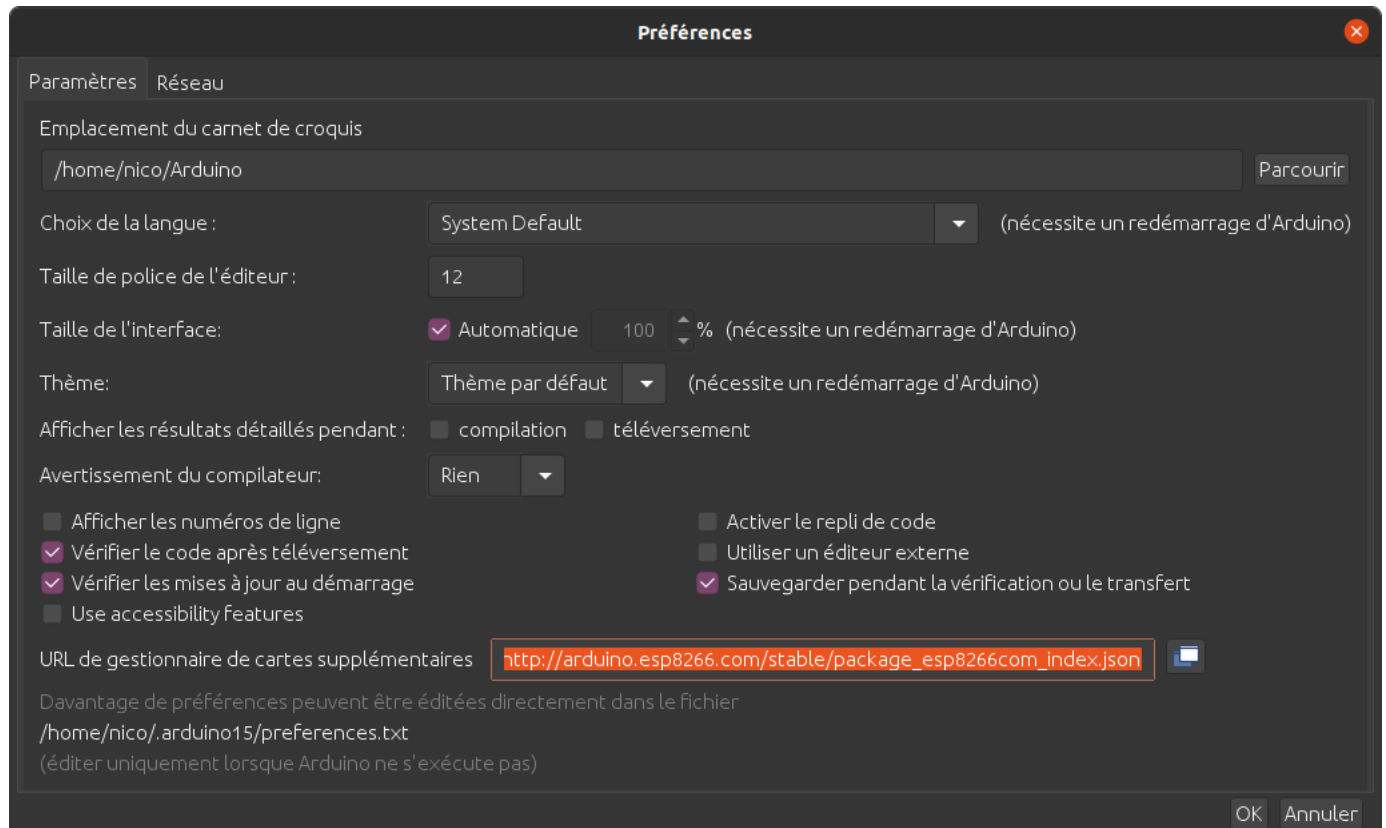



FIGURE A.3 – Lien pour les cartes ESP8266

Puis faire  OK

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

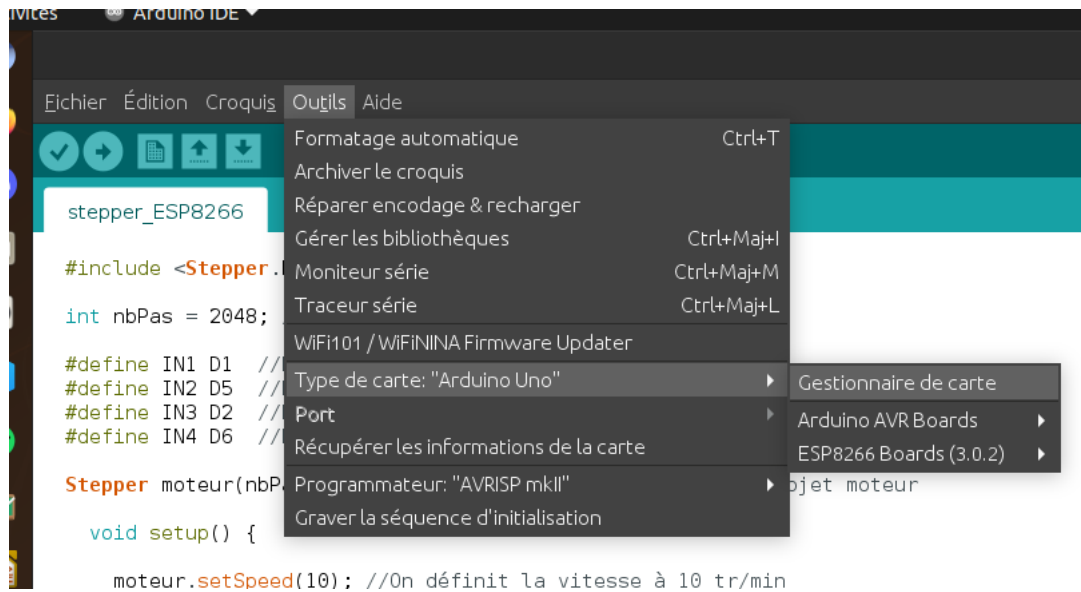


FIGURE A.4 – Gestionnaire des cartes ESP8266

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

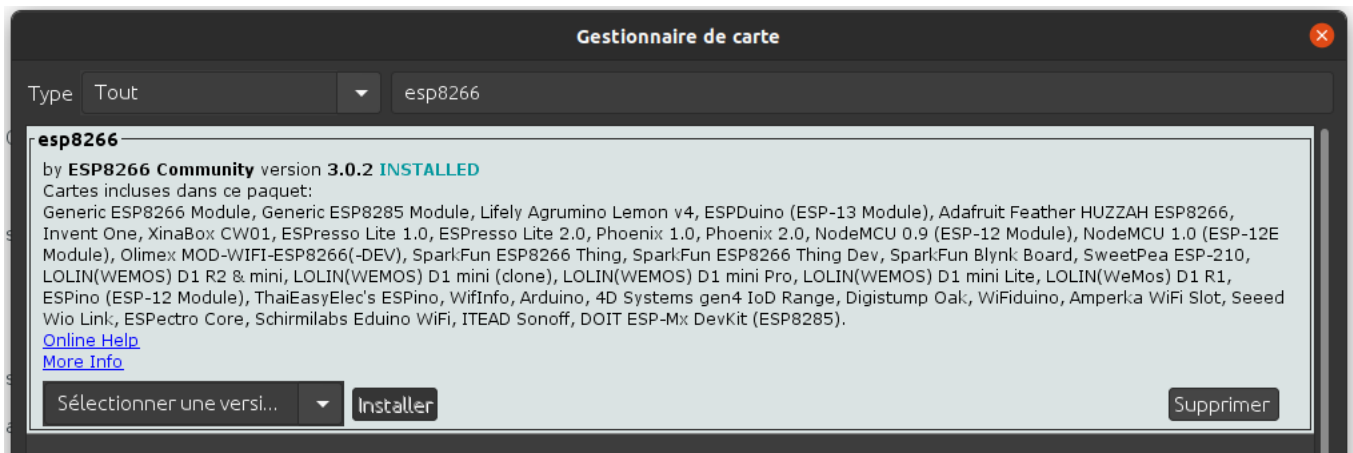


FIGURE A.5 – Installation des bibliothèques ESP8266

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)

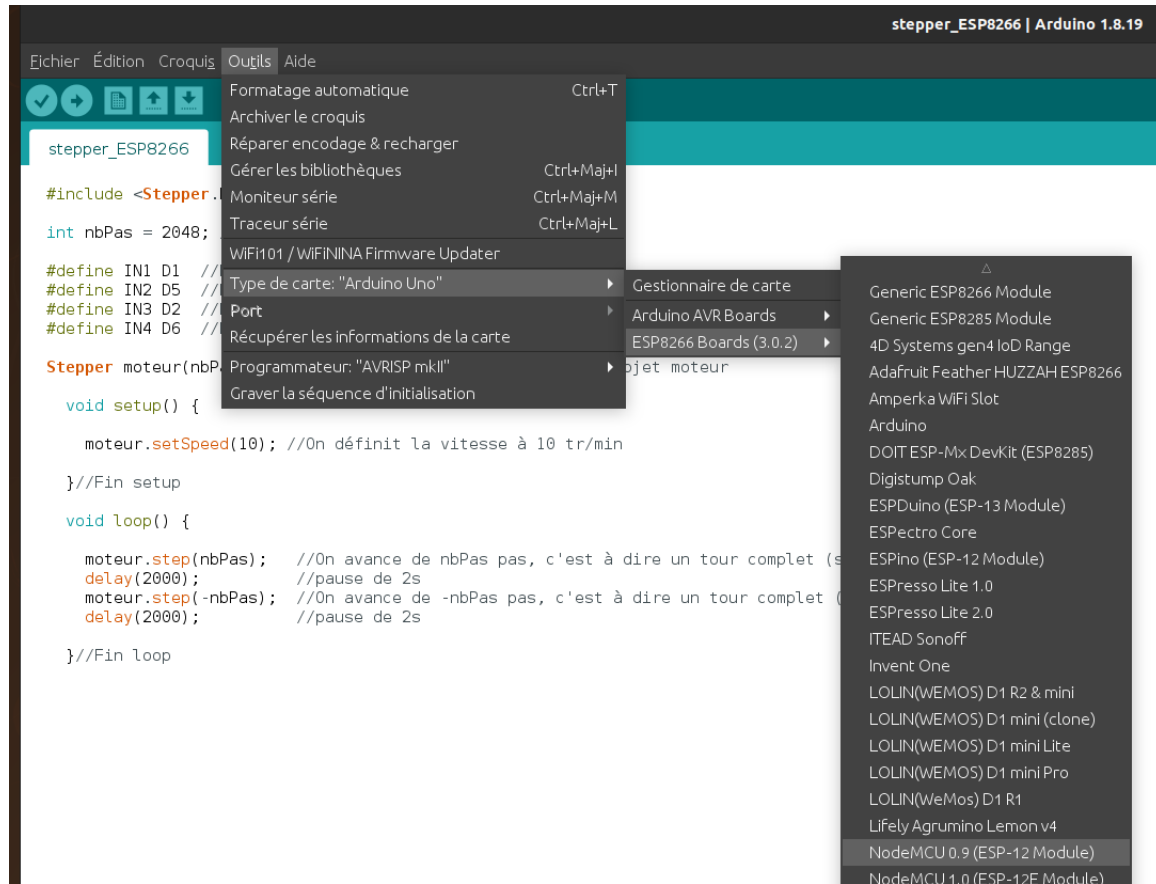


FIGURE A.6 – Sélection de la carte ESP12

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

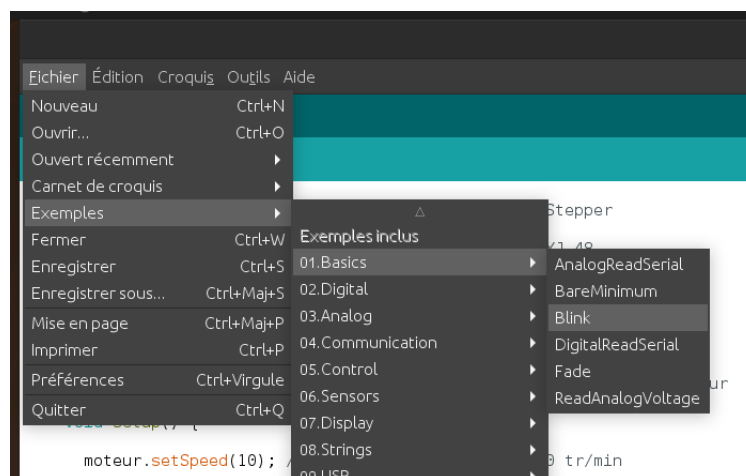


FIGURE A.7 – Emplacement de l'exemple Blink

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 connaitre 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 connaitre 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.