

Table des matières

1 télécharger l'EDI arduino.....	1
2 Installation de l'EDI.....	1
3 Configuration de l'EDI pour un esp32.....	4
4 Ecriture et compilation du premier programme.....	5
5 Téléversement et test du programme.....	6
6 Faire clignoter une Led sur la broche 15.....	8
7 Lire une entrée analogique.....	9
8 Scanner les Points Accès WIFI.....	10
9 Obtenir une adresse IP et l'afficher.....	11
10 Un serveur WEB tout simple.....	12
2 Créer une icône sur le bureau.....	16

1 Télécharger l'EDI arduino

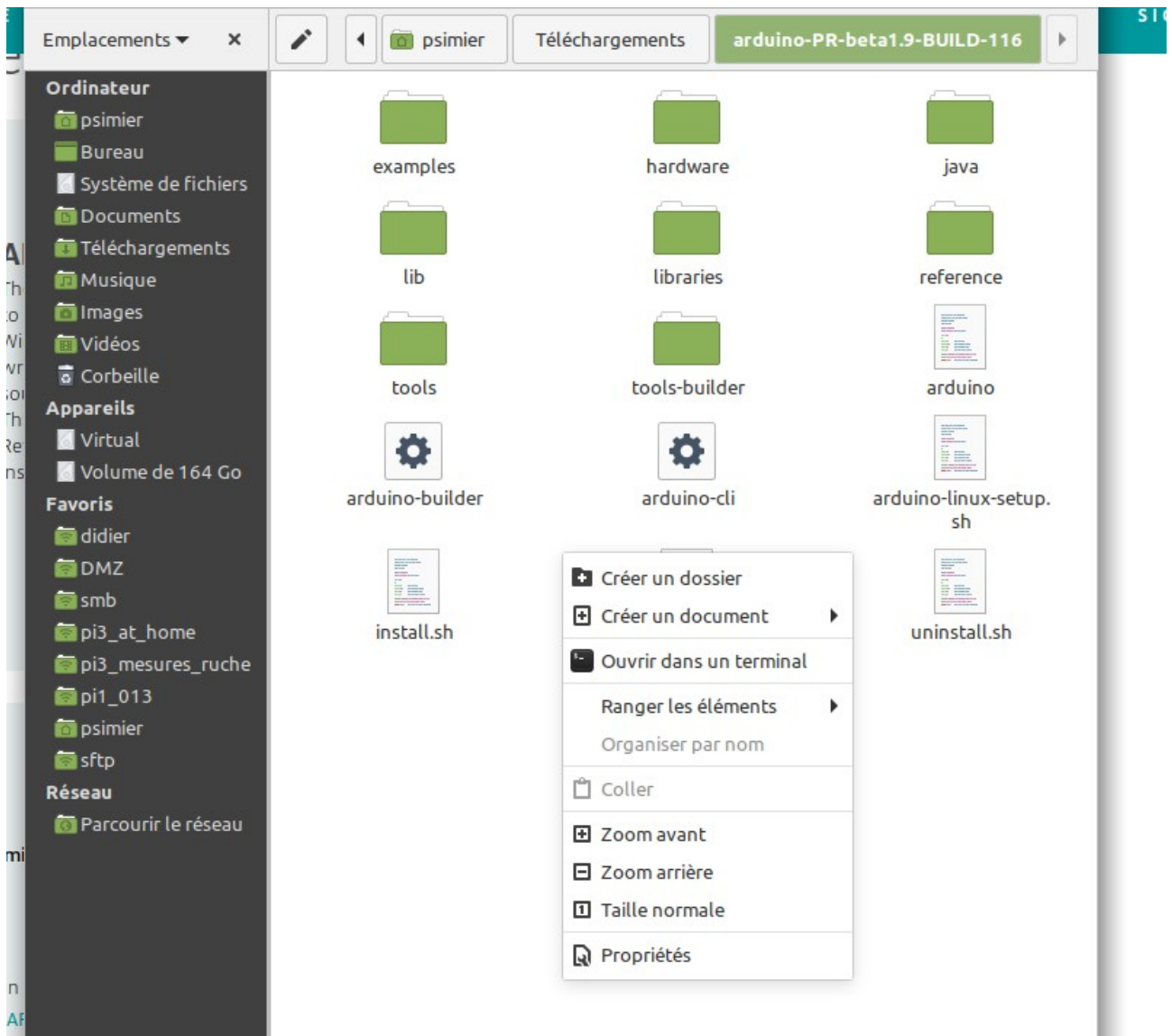
Télécharger la dernière version de l'EDI
(Version beta 1,9 BUILD-116) novembre 2019

<https://www.arduino.cc/en/Main/Software>

Décompresser l'archive

2 Installation de l'EDI

Dans le navigateur de fichier, faire un clic droit puis choisir ouvrir dans un terminal



passer en super utilisateur

```
psimier@b107PSR:~/Téléchargements/arduino-PR-beta1.9-BUILD-116$ su
Mot de passe :
b107PSR:/home/USERS/PROFS/psimier/Téléchargements/arduino-PR-beta1.9-BUILD-116# ./install.sh
Adding desktop shortcut, menu item and file associations for IDE...

done !
```

Placer un lanceur sur le bureau la commande est la suivante :

```
/home/USERS/PROFS/psimier/Téléchargements/arduino-PR-beta1.9-BUILD-116/arduino
```

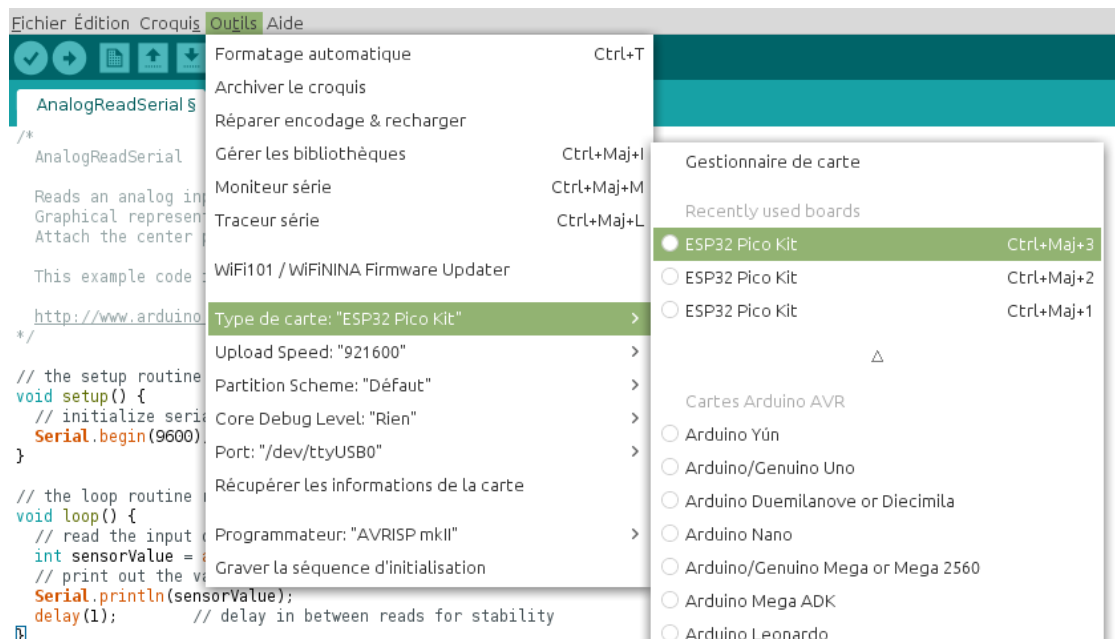
3 Configuration de l'EDI pour un esp32

Dans le menu **Fichier** choisir **Préférence**

Entrer l'URL suivante dans URL de gestionnaire de cartes supplémentaires

https://dl.espressif.com/dl/package_esp32_index.json

Ouvrez le gestionnaire de carte : depuis le menu **Outils** choisir **Type de carte** puis **Gestionnaire de carte**



Descendre l'ascenseur tout en bas et choisir **esp32**



Puis dans le menu **Outils** choisir **Port**

sélectionner Serial Port (USB) /dev/ ttyUSB0

Toujours dans le menu **Outils** choisir type de carte **ESP32 Pico kit**

4 Écriture et compilation du premier programme

```
void setup() {  
    // initialize serial communication at 9600 bits per second:  
    Serial.begin(9600);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  
    // print out the value you read:  
    Serial.println("Hello ESP32!");  
    delay(1000);           // delay in between reads for stability  
}
```

Pour compiler choisir le menu **Croquis** puis **Vérifier/Compiler**

Compilation terminée.

Le croquis utilise 214561 octets (16%) de l'espace de stockage de programmes. Le maximum est de 1310720 octets.
Les variables globales utilisent 15380 octets (4%) de mémoire dynamique, ce qui laisse 312300 octets pour les variables locales. Le maximum est de 327680 octets.

5 Téléversement et test du programme

Pour Téléverser le programme choisir dans le menu **Croquis** puis **Téléverser**

```
Téléversement terminé
Hash of data verified.
Compressed 214672 bytes to 109569...

Writing at 0x00010000... (14 %)
Writing at 0x00014000... (28 %)
Writing at 0x00018000... (42 %)
Writing at 0x0001c000... (57 %)
Writing at 0x00020000... (71 %)
Writing at 0x00024000... (85 %)
Writing at 0x00028000... (100 %)
Wrote 214672 bytes (109569 compressed) at 0x00010000 in 1.5 seconds (effective 1116.0 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 8428.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

26
```

Pour vérifier les données envoyées sur la liaison série

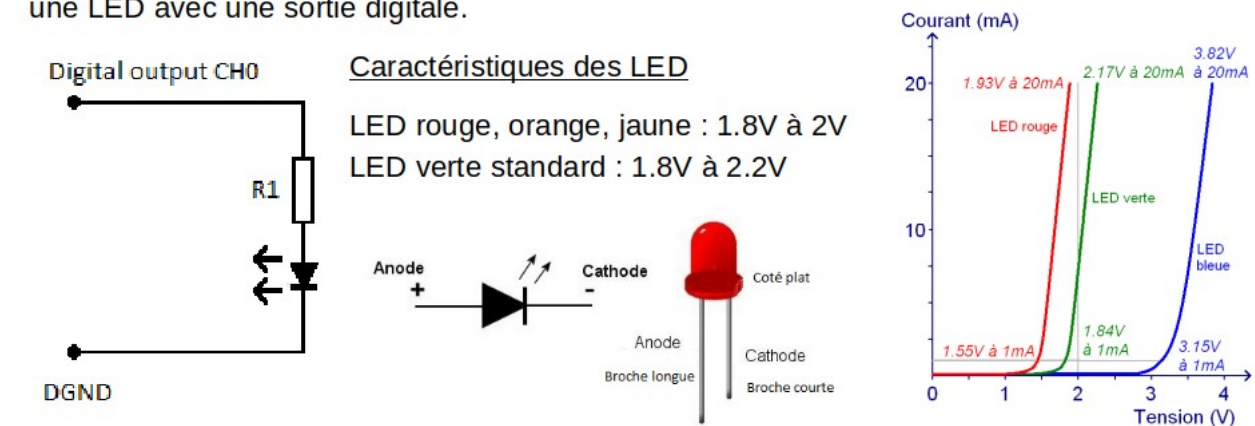
Dans le menu **Outils** choisir **Moniteur série**



Le programme écrit dans la console , chaque seconde sur une nouvelle ligne
Hello ESP32!

6 Faire clignoter une Led sur la broche 15

Pour tester le fonctionnement du module, on propose dans un premier temps d'allumer une LED avec une sortie digitale.



Vous pouvez brancher la led directement entre Gnd et D15 (la broche courte est connectée sur Gnd). Le courant de sortie de l'ESP32 est limité à 20 mA.

pinMode : Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie. Les broches analogiques peuvent être utilisées en tant que broches numériques.

DigitalWrite : Met un niveau logique HIGH ou LOW sur une broche numérique.

Le programme

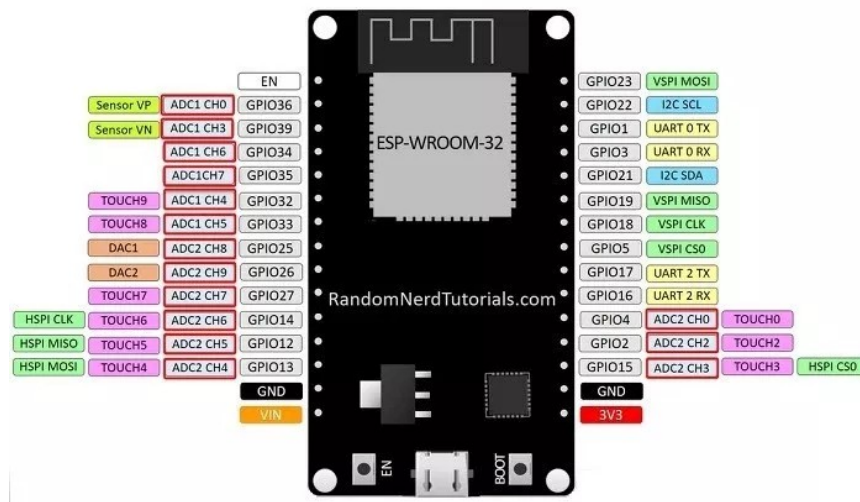
```
void setup() {  
    Serial.begin(9600);  
    pinMode(15, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(15, HIGH);  
    delay(1000);  
    digitalWrite(15, LOW);  
    delay(1000);  
}
```

7 Lire une entrée analogique

Lire une entrée analogique avec l'ESP32 est aussi simple que d'utiliser la fonction `analogRead()`. Elle accepte comme argument le GPIO que vous voulez lire.

L'ESP32 prend en charge les entrées analogiques sur 18 canaux différents, cependant seuls 15 sont disponibles sur la carte (encadrés en rouge sur le schéma).

ESP32 DEVKIT V1 - DOIT



Les convertisseurs d'entrée analogique ont une résolution de 12 bits. Cela signifie que lorsque vous lisez une entrée analogique, sa valeur peut varier de 0 à 4095.

Code lecture et affichage de l'entrée analogique sur la broche 34.

```
#define LED 13
#define InAna 34

void setup() {
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}

void loop() {
    int value = analogRead(InAna);
    Serial.println(value);
    digitalWrite(LED, HIGH);
    delay(200);
    digitalWrite(LED, LOW);
    delay(1000);
}
```

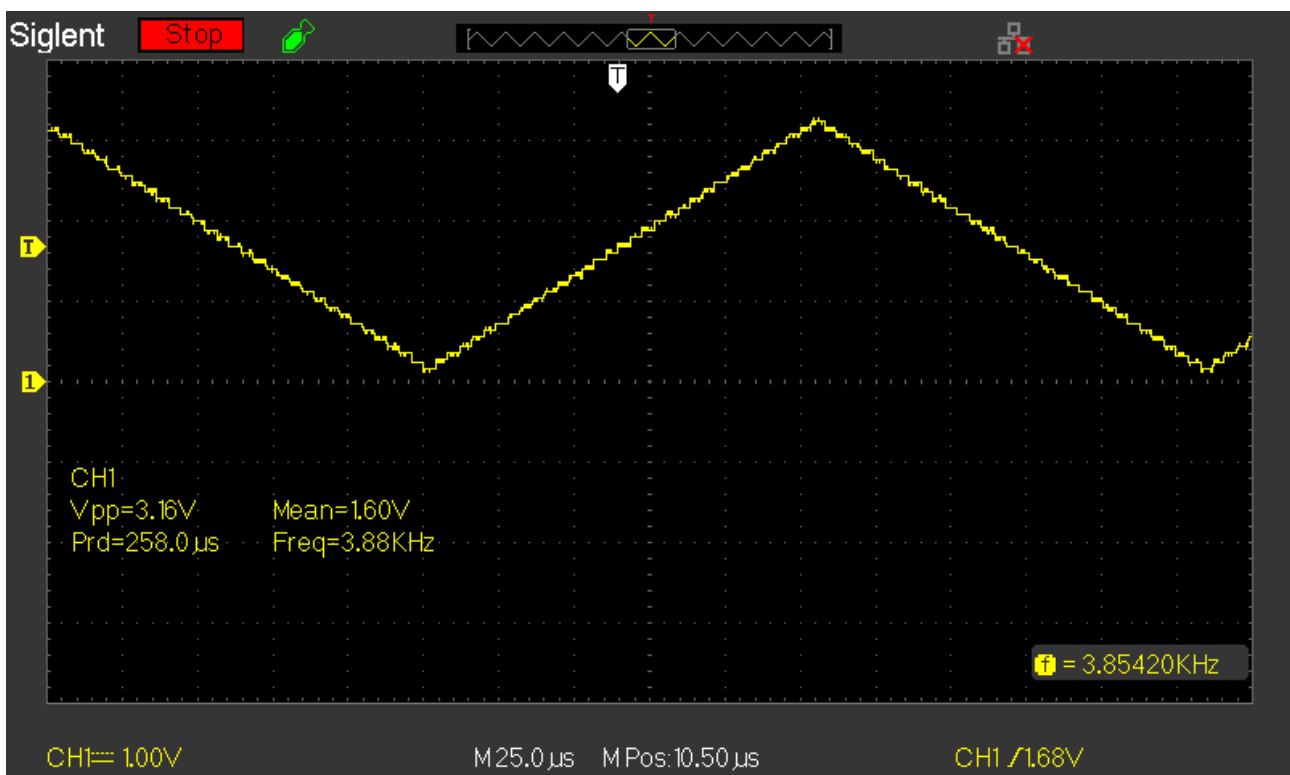
8 Ecrire une sortie analogiques

L'ESP32 possède deux convertisseurs numérique-analogique (DAC) 8 bits, connectés à **GPIO25** pour le canal 1 et à **GPIO26** pour le canal 2.

```
#include <driver/dac.h>

void setup() {
  Serial.begin(9600);
  Serial.print("Sortie analogique ");
  dac_output_enable(DAC_CHANNEL_1);
}

void loop() {
  for(int i=0; i<255; i+=8){
    dac_output_voltage(DAC_CHANNEL_1, i);
  }
  for(int i=255; i>0; i-=8){
    dac_output_voltage(DAC_CHANNEL_1, i);
  }
}
```



On observe un signal en dent de scie de 0V à 3,16V période 258 μs

9 Scanner les Points Accès WIFI

Le premier programme WiFiScan permet de lister tous les WiFi que le module ESP 32 capte.

Dans la fonction de configuration **setup**, on met le module WiFi en mode *Station* (on aurait pu le mettre aussi en point d'accès). Ne cherchez pas la méthode **mode** dans la doc Arduino, il s'agit d'une commande spécifique au module ESP 32.

On retrouve ensuite les instructions qui permettent d'afficher les points d'accès WiFi captés par le module ESP. La méthode **scanNetwork** permet, comme son nom l'indique, de scanner les différents canaux (fréquences) dédiés au WiFi et retourner le nombre de réseaux trouvés. Elle écoute toutes les fréquences et récupère leurs noms. si ce nombre n'est pas nul, ils vont être affichés un par un grâce à une boucle **for**.

La méthode **SSID** permet d'afficher le nom du réseau WiFi. Dans le vocabulaire Wi-Fi, *SSID* veut dire *Service Set Identifier*.

La méthode **RSSI** (*Received Signal Strength Indication*) affiche la puissance du signal reçu. Les points d'accès WiFi émettent 10 fois par seconde un message donnant le nom du réseaux Wi-Fi.

```
#include "WiFi.h"
#include <WebServer.h>

#define ssid "SNIR03" // le nom (SSID) du réseau WiFi
#define password "totototo" // Le password associé

WebServer server(80);

String construirePage() {
    String page = "<html><head>";
    page += "<title>ESP32</title></head>";
    page += "<body><h1>Site Web ESP32</h1>";
    page += "</body></html>";
    return page;
}

void pageIndex() {
    server.send ( 200, "text/html", construirePage() );
}

void setup() {

    Serial.begin ( 9600 );
```

```

Serial.println("Setup");

WiFi.mode(WIFI_STA);
int n = WiFi.scanNetworks();

for (int i = 0; i < n; ++i) {
    // Print SSID and RSSI for each network found
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(WiFi.SSID(i));
    Serial.print(" (");
    Serial.print(WiFi.RSSI(i));
    Serial.println(")");
    delay(10);
}
}

void loop() {
}

```

10 Obtenir une adresse IP et l'afficher

On retrouve la connexion au réseau WiFi avec la méthode **begin** :

La boucle **while** permet d'attendre que la connexion soit effective et que le serveur DHCP du réseau ait fourni les paramètres nécessaires pour se connecter au réseau Internet, qui sont affichés avec le code suivant :

```

#include <WiFi.h>

const char* ssid      = "SNIR03";          // Nom du réseau
const char* password  = "totototo";        // clé

void setup() {
    Serial.begin(9600);
    delay(10);

    Serial.print("Connexion au WiFi ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);    // On se connecte
    while (WiFi.status() != WL_CONNECTED) { // On attend
        delay(500);
        Serial.print(".");
    }

    Serial.println(""); // on affiche les paramètres
}

```

```

Serial.println("WiFi connecté");
Serial.print("Adresse IP du module EPC: ");
Serial.println(WiFi.localIP());
Serial.print("Adresse IP de la gateway : ");
Serial.println(WiFi.gatewayIP());
}
void loop() {
}

```

On obtient dans la console au lycée le message suivant

```

Connexion au WiFi SNIR03
.
WiFi connecte
Adresse IP du module EPC: 172.18.58.31
Adresse IP de la gateway : 172.18.58.9

```

11 Un serveur WEB tout simple

```

#include <WiFi.h>
#include <WebServer.h>
#define InAna 34

const char* ssid      = "SNIR03";          // Nom du réseau
const char* password = "totototo";         // clé

WebServer server(80);

String construirePage() {
    int value = analogRead(InAna);
    String page = "<html><head>";
    page += "<title>Site ESP32</title></head>";
    page += "<body><h1>Valeur : ";
    page += value;
    page += "</h1>";
    page += "</body></html>";
    return page;
}

void pageIndex() {
    Serial.println("requete GET /");
    server.send ( 200, "text/html", construirePage() );
}

void setup() {
    Serial.begin(9600);
}

```

```

delay(10);

Serial.print("Connexion au WiFi ");
Serial.println(ssid);

WiFi.begin(ssid, password);    // On se connecte

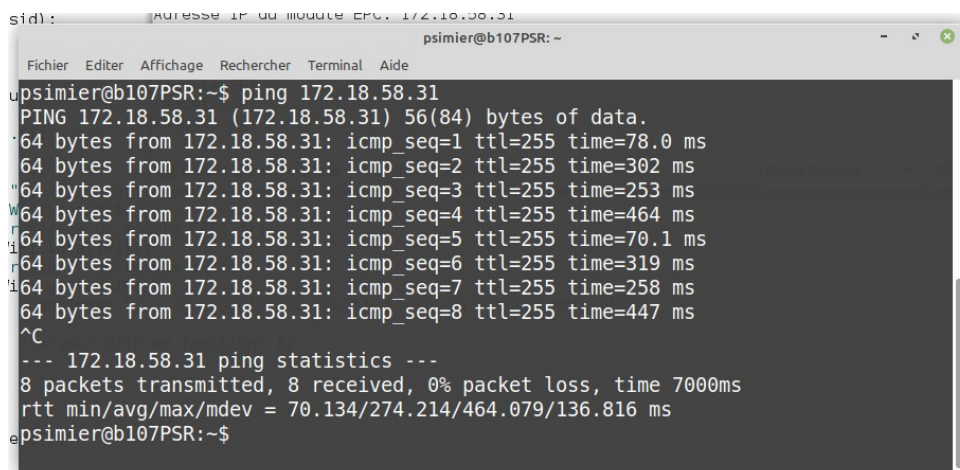
while (WiFi.status() != WL_CONNECTED) { // On attend
    delay(500);
    Serial.print(".");
}

Serial.println(""); // on affiche les paramètres
Serial.println("WiFi connecté");
Serial.print("Adresse IP du module EPC: ");
Serial.println(WiFi.localIP());
Serial.print("Adresse IP de la box : ");
Serial.println(WiFi.gatewayIP());
server.on ( "/", pageIndex ); // associe une fonction pour la méthode http GET /
server.begin();
Serial.println ( "Serveur HTTP en fonction" );
}

void loop() {
    server.handleClient();
}

```

Il est possible de vérifier la connexion réseau avec la commande ping



```

psimier@b107PSR:~$ ping 172.18.58.31
PING 172.18.58.31 (172.18.58.31) 56(84) bytes of data.
64 bytes from 172.18.58.31: icmp_seq=1 ttl=255 time=78.0 ms
64 bytes from 172.18.58.31: icmp_seq=2 ttl=255 time=302 ms
64 bytes from 172.18.58.31: icmp_seq=3 ttl=255 time=253 ms
64 bytes from 172.18.58.31: icmp_seq=4 ttl=255 time=464 ms
64 bytes from 172.18.58.31: icmp_seq=5 ttl=255 time=70.1 ms
64 bytes from 172.18.58.31: icmp_seq=6 ttl=255 time=319 ms
64 bytes from 172.18.58.31: icmp_seq=7 ttl=255 time=258 ms
64 bytes from 172.18.58.31: icmp_seq=8 ttl=255 time=447 ms
^C
--- 172.18.58.31 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7000ms
rtt min/avg/max/mdev = 70.134/274.214/464.079/136.816 ms
psimier@b107PSR:~$

```

12 Un client web

On doit donc commencer par établir une connexion avec le réseau WiFi. C'est la méthode `begin()` de la classe `WiFi` qui s'en charge.

```
WiFi.begin(ssid, password);
```

On continue le déroulement du programme que lorsqu'une connexion a été établie. La méthode `status()` retourne l'état de la connexion.

```
while (WiFi.status() != WL_CONNECTED) { // On attend
    delay(500);
    Serial.print(".");
}
```

Maintenant qu'on dispose d'une connexion WiFi, on va envoyer à intervalle régulier une requête au serveur. Pour cela, on doit créer un objet client qui se connectera au serveur.

```
WiFiClient client;
const char* host    = "Example.com";
const int httpPort = 80;

if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    client.stop() ;
    delay(1000);
    return;
}
```

Il ne reste plus qu'à envoyer la requête au serveur avec la méthode `print()` de `WiFiClient`.

```
String url = String("/");
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
```

Puis on attends la réponse du serveur

```
unsigned long timeout = millis();
while (client.available() == 0) {
    if (millis() - timeout > 5000) {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return;
    }
}
```

On lit les données reçues, s'il y en a

```
while(client.available()){
    String line = client.readStringUntil('\r'); // découpe ligne par ligne
    Serial.print(line);
}
// plus de données
Serial.println();
Serial.println("connexion fermée");
client.stop();
```

La sortie du programme obtenue avec Example.com

```
WiFi connecte
Adresse IP du module EPC: 172.18.58.31
Adresse IP de la gateway : 172.18.58.9
Connexion au serveur : Example.com
demande URL: /
HTTP/1.1 200 OK
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Wed, 04 Dec 2019 10:00:09 GMT
Etag: "3147526947+gzip+ident"
Expires: Wed, 11 Dec 2019 10:00:09 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (bsa/EB15)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close

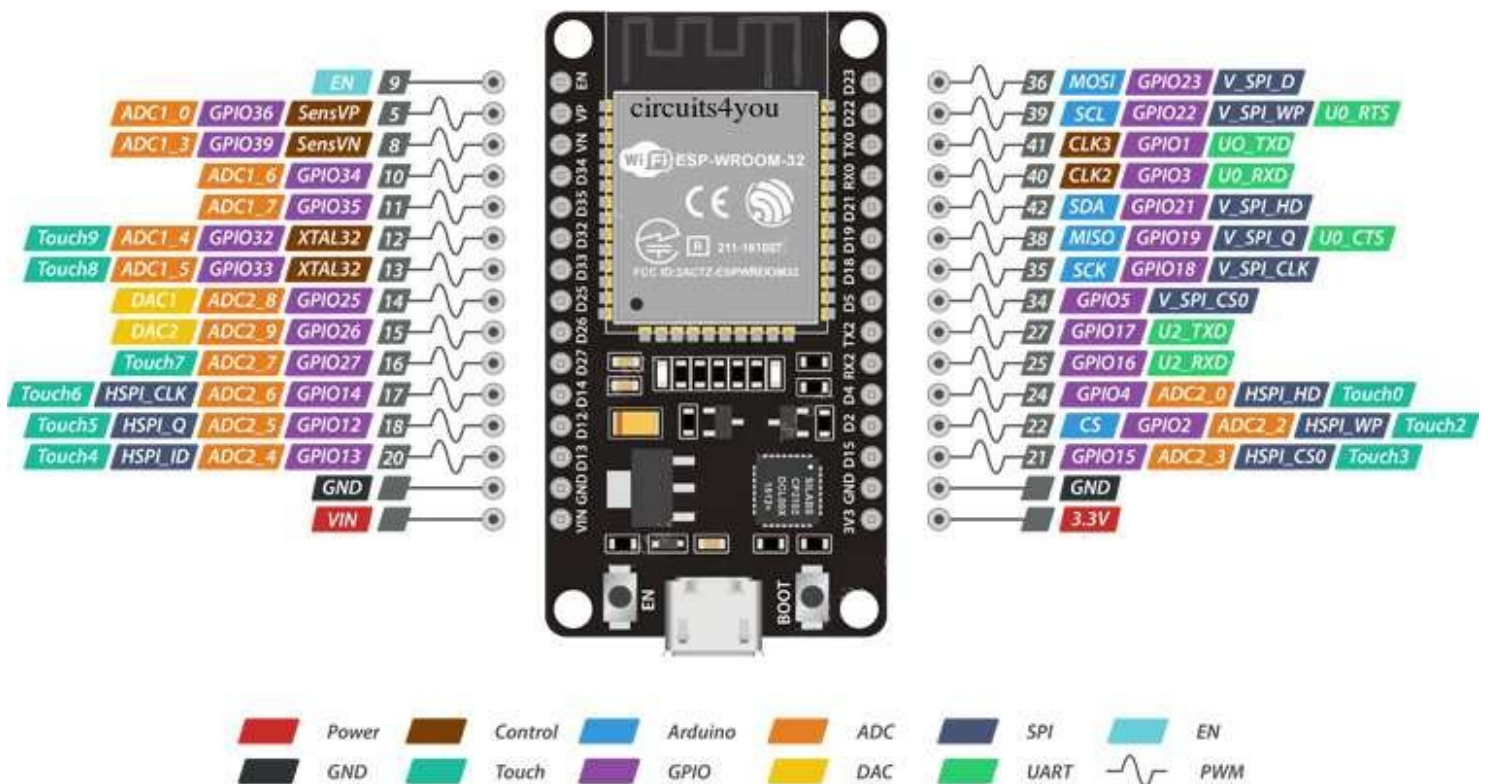
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont,
    } ...
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
```

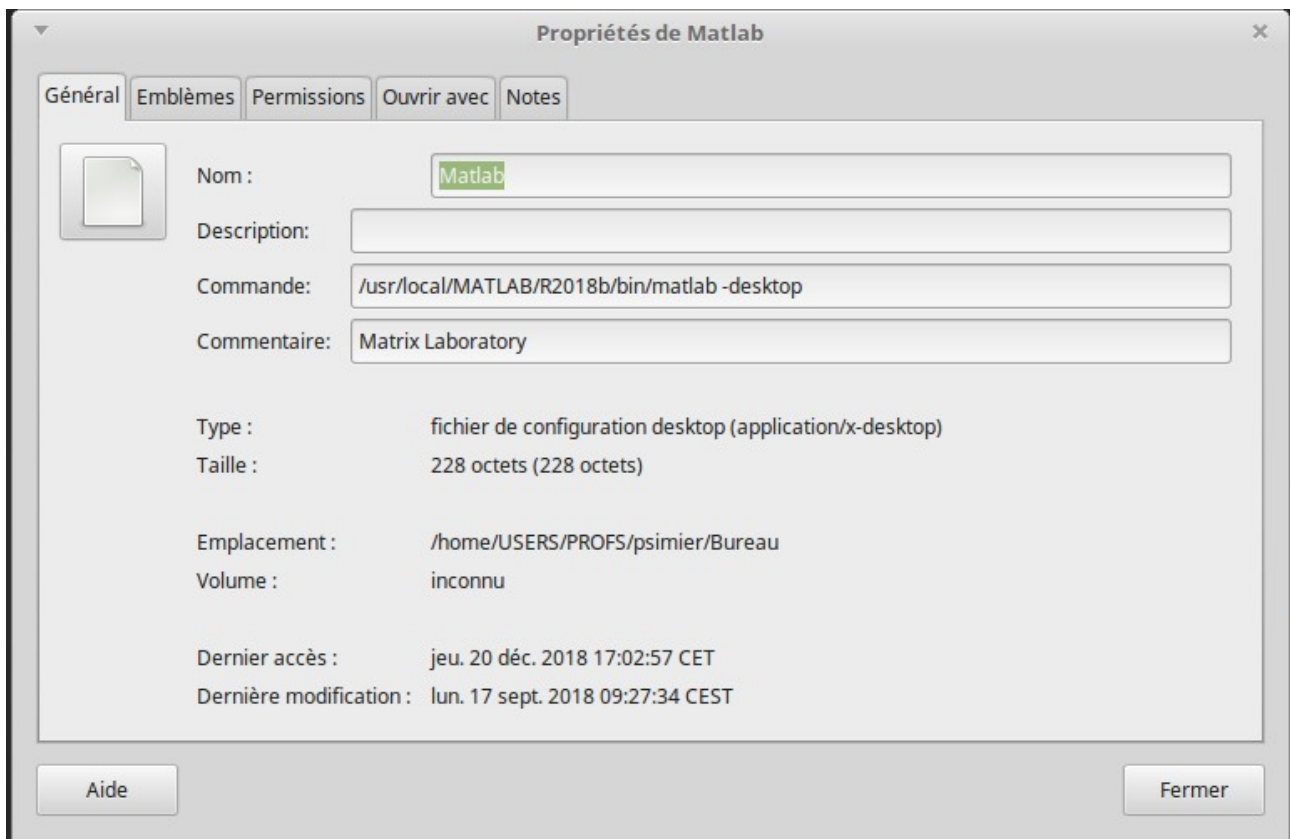
```
domain in literature without prior coordination or asking for permission.</p>
<p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

ESP32-WROOM-32 Espressif Systems

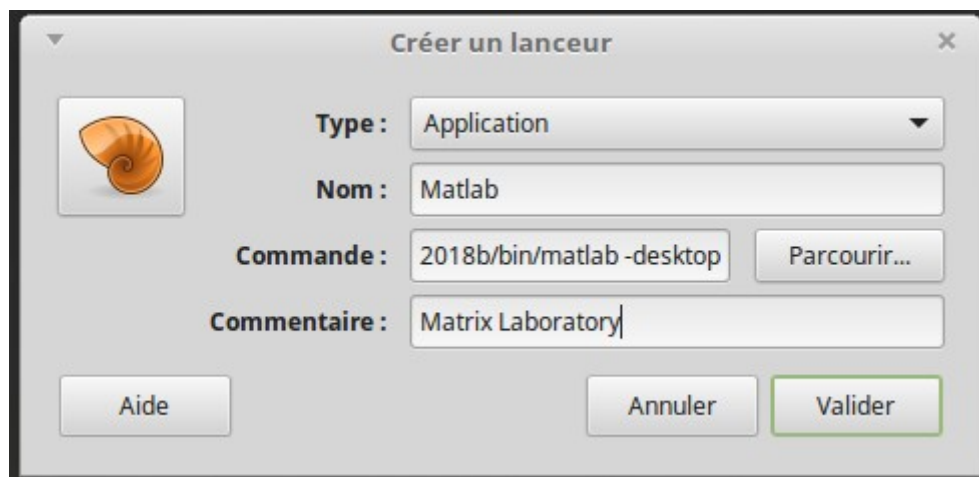


<https://electroniqueamateur.blogspot.com/2019/08/piloter-des-leds-par-wifi-esp32-esp8266.html>

2 Créer une icône sur le bureau



Sur le bureau faire un clique droit puis créer un lanceur



Nom : Matlab

Commande : /usr/local/MATLAB/R2018b/bin/matlab -desktop

Commentaire : Matrix Laboratory