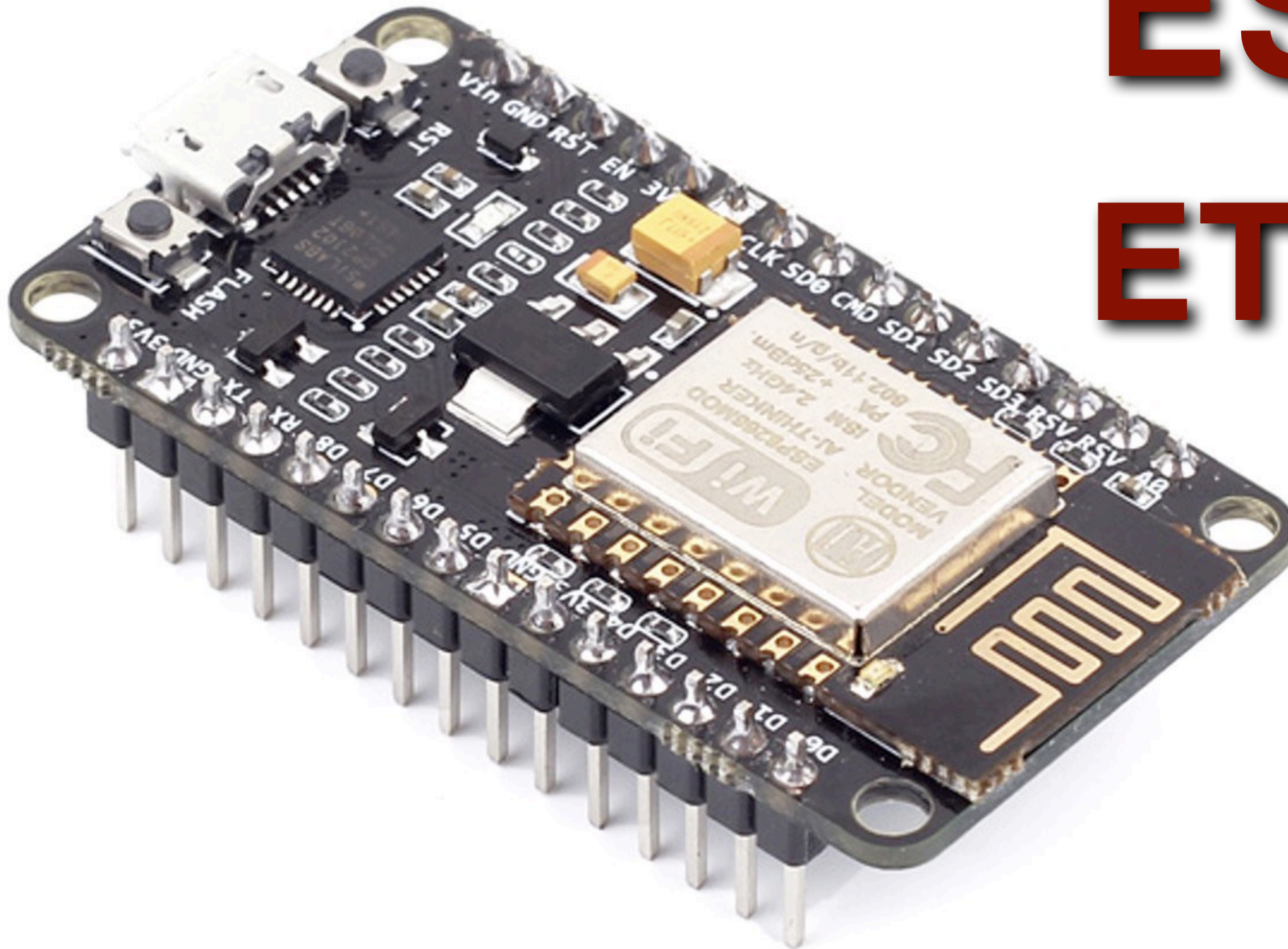


ESP32 ET WEB



Réseau, commande à distance

Jean-Marie Steinmetz

jeanmariesteinmetz@gmail.com

Qu'est-ce qu'un ingénieur ENIM ?

Mécatronique

Electronique

Informatique

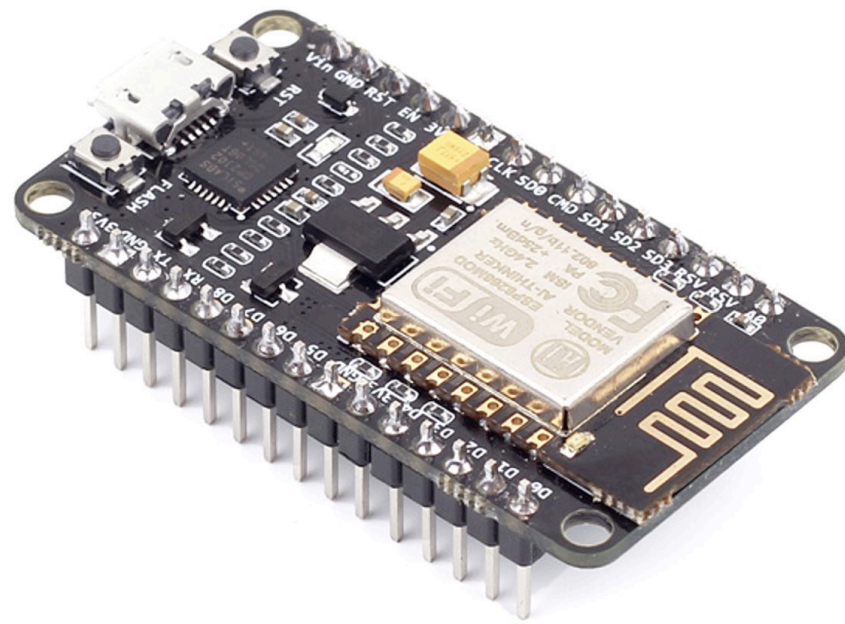
Mécanique

4ème révolution industrielle





Pour aborder la
mécatronique, nous
allons utiliser un petit
module très économique
(- de 4€ !)
qui permet de faire
beaucoup de choses !



Le module ESP32

- ESP32 est une série de microcontrôleurs intégrant la gestion du Wi-Fi.
- C'est une évolution de l'ESP8266 et il a une philosophie assez proche des modules Arduino.
- Ce module permet de développer des objets connectés : IoT Internet of Things, internet des objets.

EXEMPLES



esp32

Catégories Similaires

Electronique

Pièces de rechange et accessoires

Circuits

Sans fil Module

Capteurs de caméra

Modules de domotique

Kits domotiques

Affichage Écran

Drone GPS

Composants électroniques et fournitures

Circuits intégrés

Capteurs

Ordinateur et bureautique

Outils

[Voir plus](#)

Compatible Marque/Modèle

Toutes les Catégories > "esp32" (1,462 Résultats)

Prix:

min

- max



Livraison depuis



☐ Expédition rapide

☐ Expédi

Trier par:

Pertinence

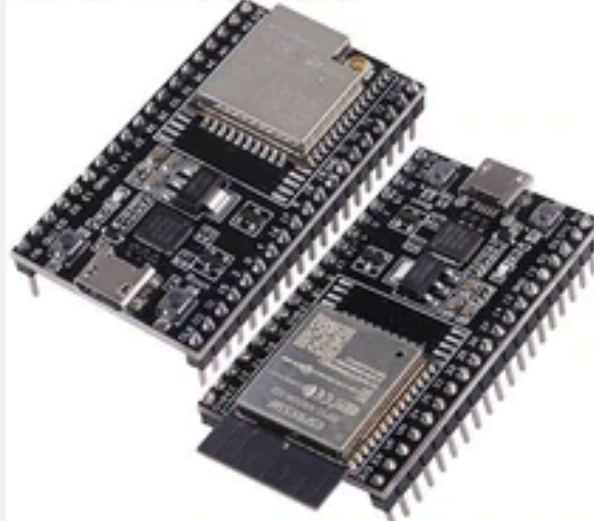
Commandes

Dates

Prix



ESP32-WROOM-32U



ESP32-WROOM-32D

ESP32-DevKitC noyau ESP32 dével...

€ 3,64

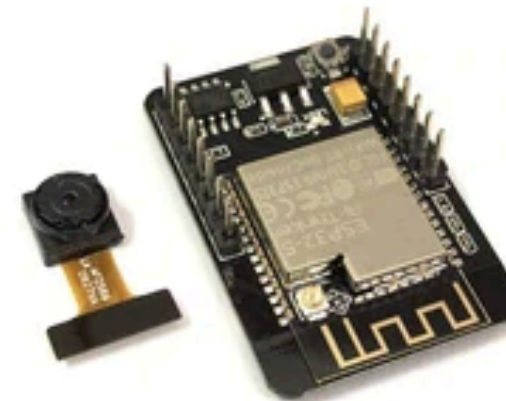
Expédition: € 2,25

★ 4.9

1570 Vendus

[Annonce](#)

LANDZO



ESP32-CAM WiFi Module WiFi ESP3...

€ 9,30

Expédition: € 0,51

★ 5.0

26 Vendus

[Annonce](#)

D1 MINI ESP32



D1 Mini ESP32 ESP-32 WiFi + Bluet...

€ 3,75

Expédition: € 0,89

Retour gratuit

34 Vendus

[Annonce](#)

Programmation ESP32

Nous allons utiliser le MicroPython qui est une variante pour l'embarqué du langage Python.

```
import esp
esp.osdebug(None)

import gc
```

SSID : Service Set Identifier, nom d'un réseau sans fil

```
ssid = 'WIFI ENIM'
password = 'votre mot de passe'

station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
    pass

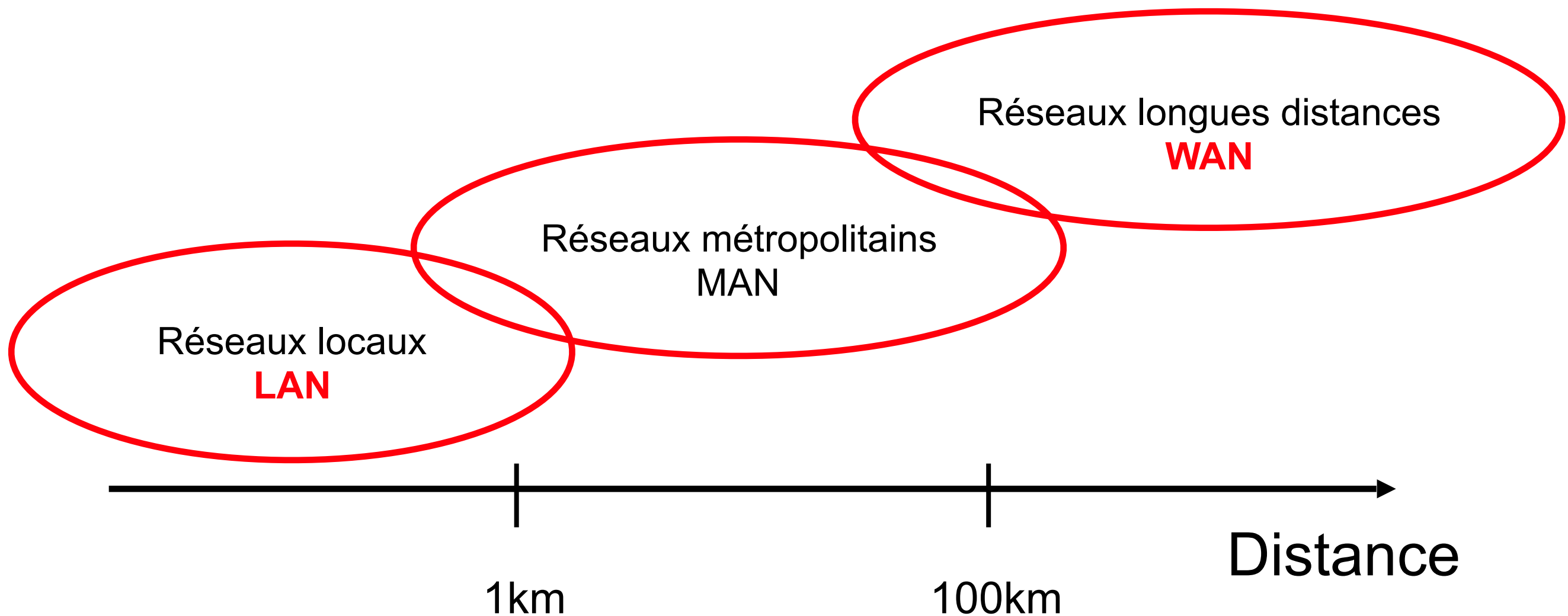
print('Connection successful')
```

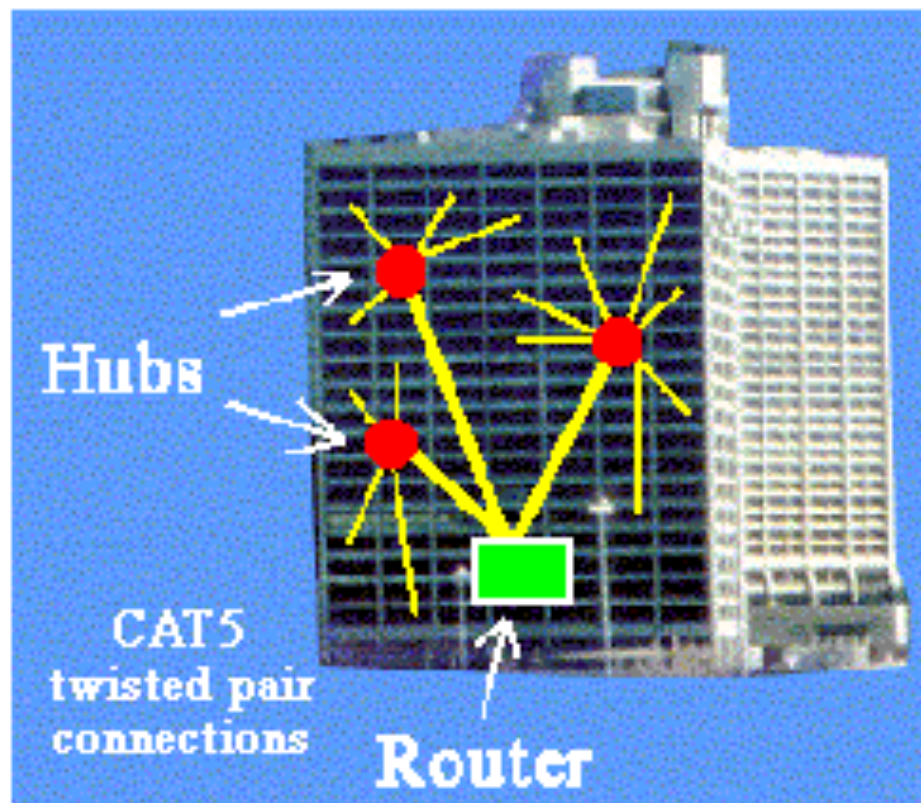
RÉSEAU

WLAN

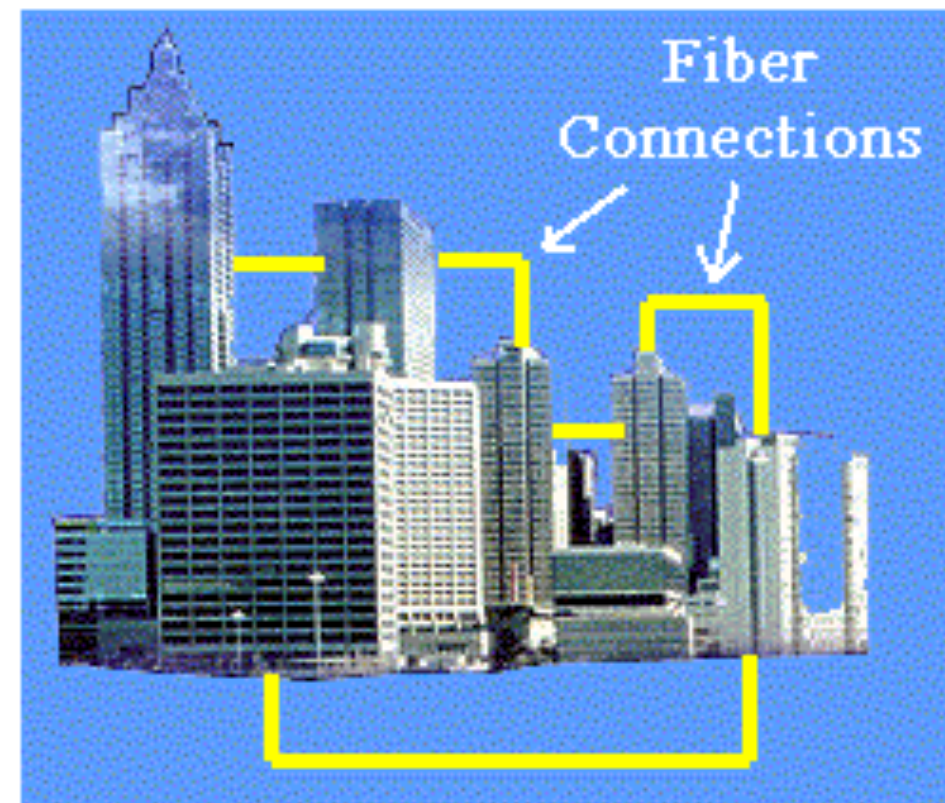
WLAN ? Echelle des réseaux

- L'échelle des réseaux est déterminée par la longueur de transmission des informations.

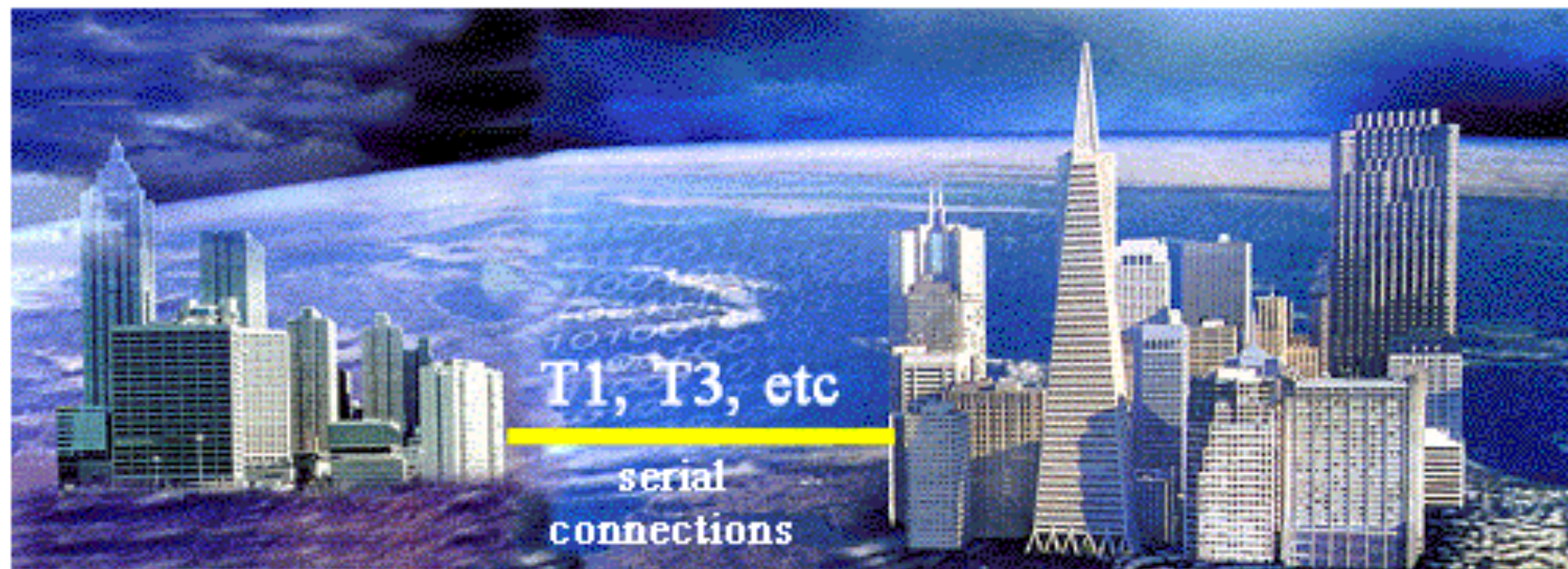




LAN



MAN



WAN

ESP32 et réseau



Hybrid Wi-Fi & Bluetooth Chip

ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

ESP32 et réseau

Table 1: ESP32-WROOM-32 Specifications

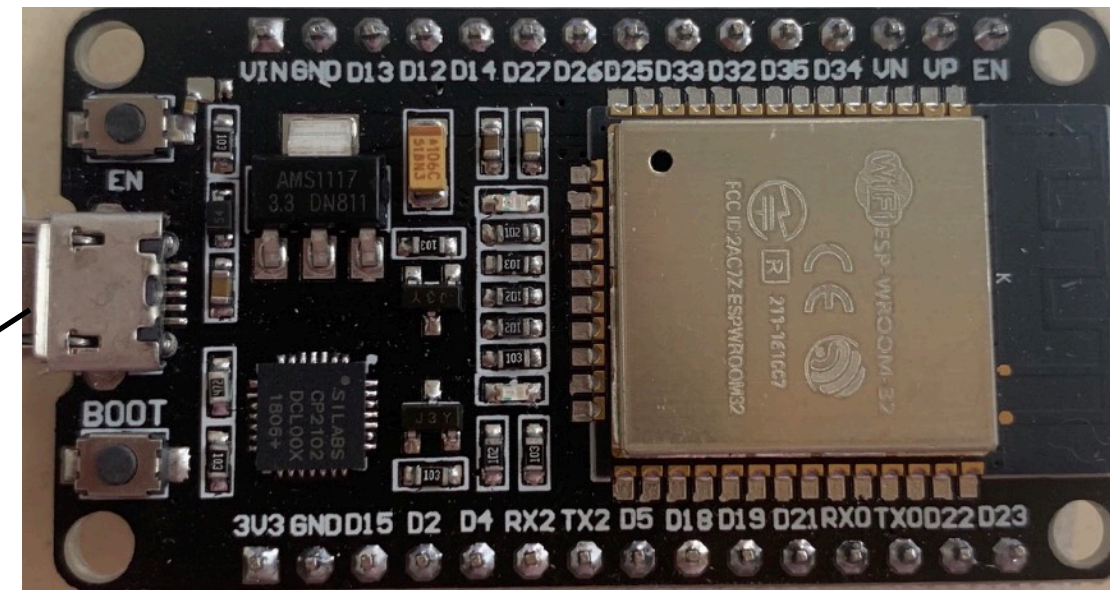
Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance 
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliability	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps) 
		A-MPDU and A-MSDU aggregation and 0.4 μ s guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz 
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH

ESP32 et réseau

Protocoles 802.11 b/g/n

Protocole	Dates	Fréquences	Taux de transfert (Typ)	Taux de transfert (Max)	Portée moyenne (intérieur)	Portée (extérieur)
Norme initiale	1997	2,4-2,5 GHz	1 Mbit/s	2 Mbit/s	?	?
802.11a	1999	5,15-5,35 GHz 5,47-5,725 / 5,725-5,875	25 Mbit/s	54 Mbit/s	~25 m	~75 m
802.11b	1999	2,4-2,5 GHz	6,5 Mbit/s	11 Mbit/s	~35 m	~100 m
802.11g	2003	2,4-2,5 GHz	25 Mbit/s	54 Mbit/s	~25 m	~75 m
802.11n	2009	2,4 GHz et/ou 5 GHz	200 Mbit/s	450 Mbit/s	~50 m	~125 m
802.11ac	jan. 2014	5,15-5,35 GHz 5,47-5,875 GHz	433 Mbit/s	1300 Mbit/s	~20 m	~50 m

SSID nom du WIFI
mot de passe



Câble USB
pour alimentation
et transfert



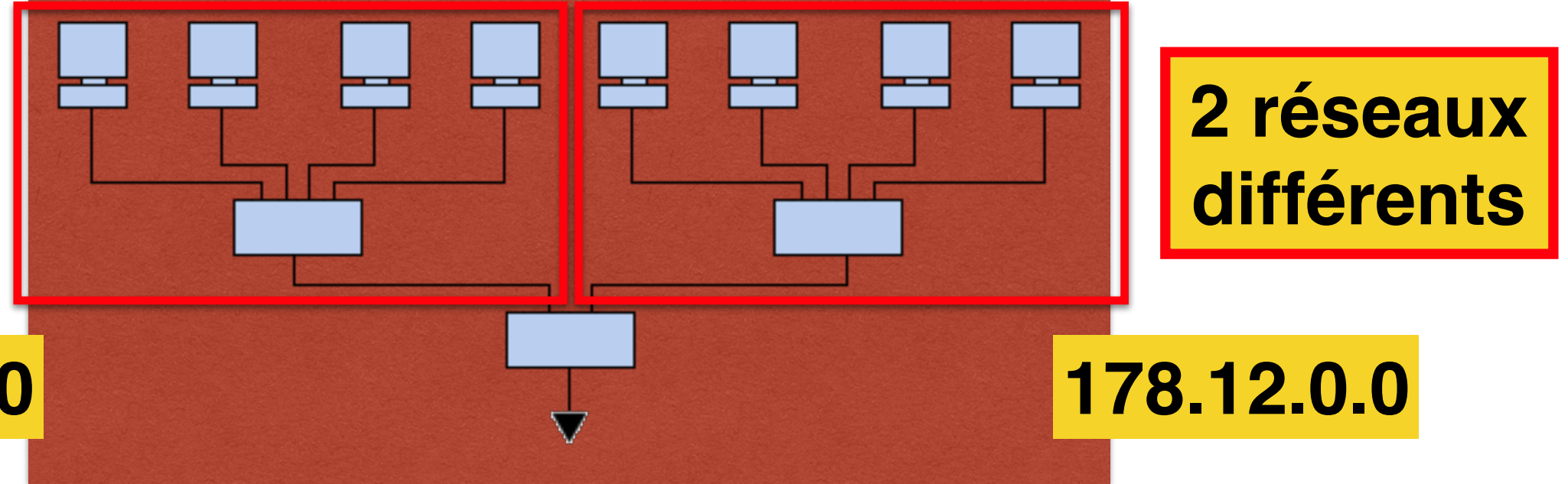
Adresse IP



Adresse IP

- Sur Internet, les ordinateurs communiquent entre eux grâce au protocole IP (Internet Protocol), qui utilise des adresses numériques, appelées adresses IP, composées de 4 nombres entiers (4 octets) entre 0 et 255 et notées sous la forme xxx.xxx.xxx.xxx.
- Par exemple, 194.153.205.26 est une adresse IP.
- Ces adresses servent aux ordinateurs du réseau pour communiquer entre-eux, ainsi chaque ordinateur d'un réseau possède une **adresse IP qui doit être unique sur ce réseau.**

- Une **adresse IP** est une adresse 32 bits, généralement notée sous forme de 4 nombres entiers séparés par des points. On distingue en fait deux parties dans l'adresse IP :
 - une partie des nombres à gauche désigne le réseau et est appelée **adresse de réseau** (en anglais *netID*),
 - Les nombres de droite désignent les ordinateurs de ce réseau et est appelée **adresse machine** (en anglais *host-ID*).



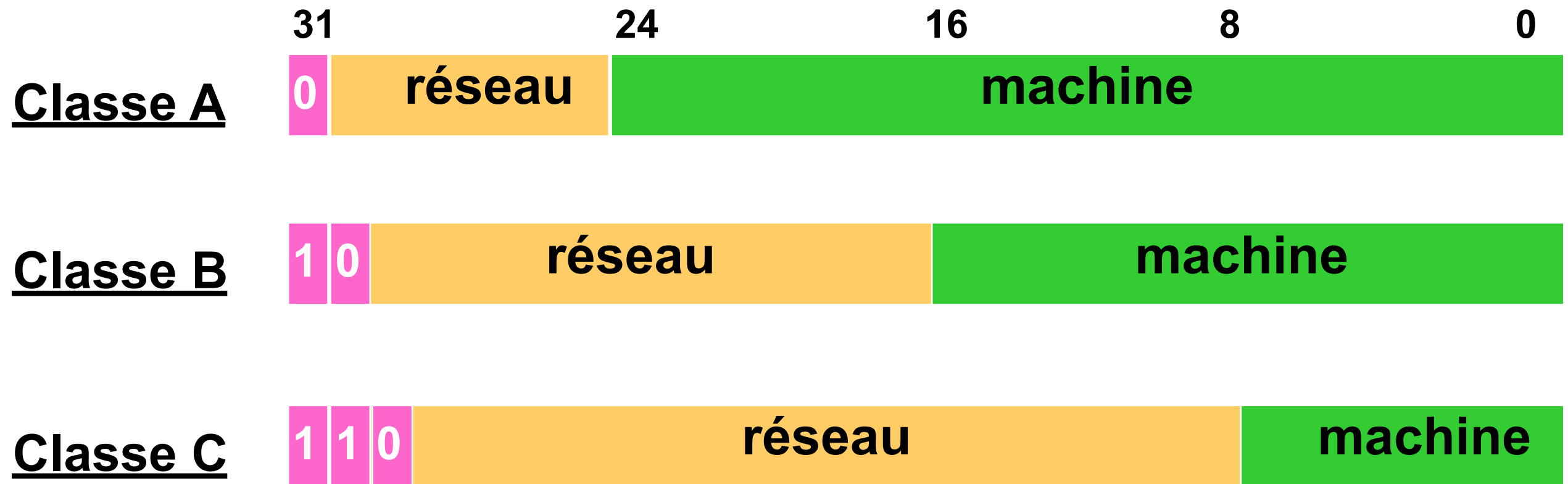
- Soit l'exemple ci-dessus :
- Notons le réseau de gauche 194.28.12.0. Il contient les ordinateurs suivants :
 - 194.28.12.1 à 194.28.12.4
- Notons celui de droite 178.12.0.0. Il comprend les ordinateurs suivants :
 - 178.12.77.1 à 178.12.77.4

- Dans le cas précédent, les réseaux sont notés 194.28.12.0 et 178.12.0.0, puis on numérote incrémentalement chacun des ordinateurs le constituant.
- Ceci peut être fait automatiquement par un serveur (DHCP, comme dans le réseau de l'ENIM) ou manuellement.
- Un serveur DHCP affecte des adresses IP différentes à chaque ordinateur qui le demande dans le réseau, mais dans notre cas, nous devons choisir une adresse fixe et manuellement.

Classes de réseaux

- Les adresses IP sont réparties en plusieurs classes suivant le nombre de machines devant s'y trouver.
- On distingue principalement 3 classes de réseaux :
A, B et C
 - A : grands réseaux
 - B : réseaux moyens
 - C : petits réseaux

Les classes IP



Les adresses binaires de la classe A
commencent toujours par le bit 0
ceux de la classe B, par les bits 1 et 0
ceux de la classe C par les bits 110

Quelles sont les plages d'adresses des classes A, B et C ?

Classe A : de 0000 0000 à 0111 1111

$$0111\ 1111 = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$$

c'est à dire de 0 à 127

Classe B : de **10**00 0000 à **10**11 1111

$$\mathbf{10}00\ 0000 = 128$$

$$\mathbf{10}11\ 1111 = 128 + 32 + 16 + 8 + 4 + 2 + 1 = 191$$

càd de 128 à 191

Classe C : de **1100** 0000 à **1101** 1111

$$\mathbf{1100}\ 0000 = 128 + 64 = 192$$

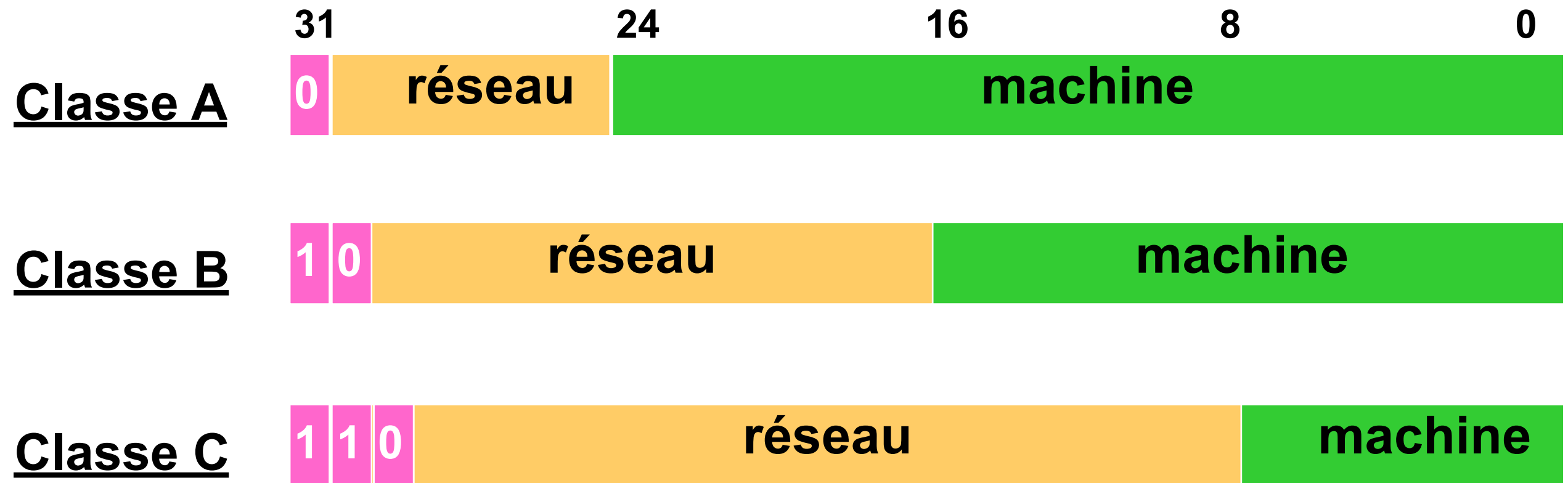
$$\mathbf{1101}\ 1111 = 128 + 64 + 16 + 8 + 4 + 2 + 1 = 223$$

càd de 192 à 223

Adresses réservées

- Il y a des adresses particulières qui sont réservées à des usages particuliers :
 - celles qui se terminent par **255**, ce sont des adresses de **Broadcast**, diffusion générale à tout le réseau
 - celles qui se termine par **0**, réservées aux **adresses du réseau**
- On ne peut donc pas les affecter à des machines particulières

Nombre d'adresses différentes



classe A : adresse machine sur 24 bits : $2^{24} - 2$
(2 adresses sont réservées : 0 et 255)

classe B : adresse machine sur 16 bits : $2^{16} - 2$

classe C : adresse machine sur 8 bits : $2^8 - 2$

- Classe A
 - Plage d'adresses :
 - Nombre de machines :
 - Taille du réseau :
- Classe B
 - Plage d'adresses :
 - Nombre de machines :
 - Taille du réseau :
- Classe C
 - Plage d'adresses :
 - Nombre de machines :
 - Taille du réseau :

Exercices

- Soit les adresses IP suivantes :
192.19.140.25
172.18.51.6
101.33.15.1
- Donnez pour chaque adresse, la classe de celle-ci, la partie adresse réseau et la partie adresse machine.
- A quel type de réseau, correspondent les différentes classes de ces adresses ?

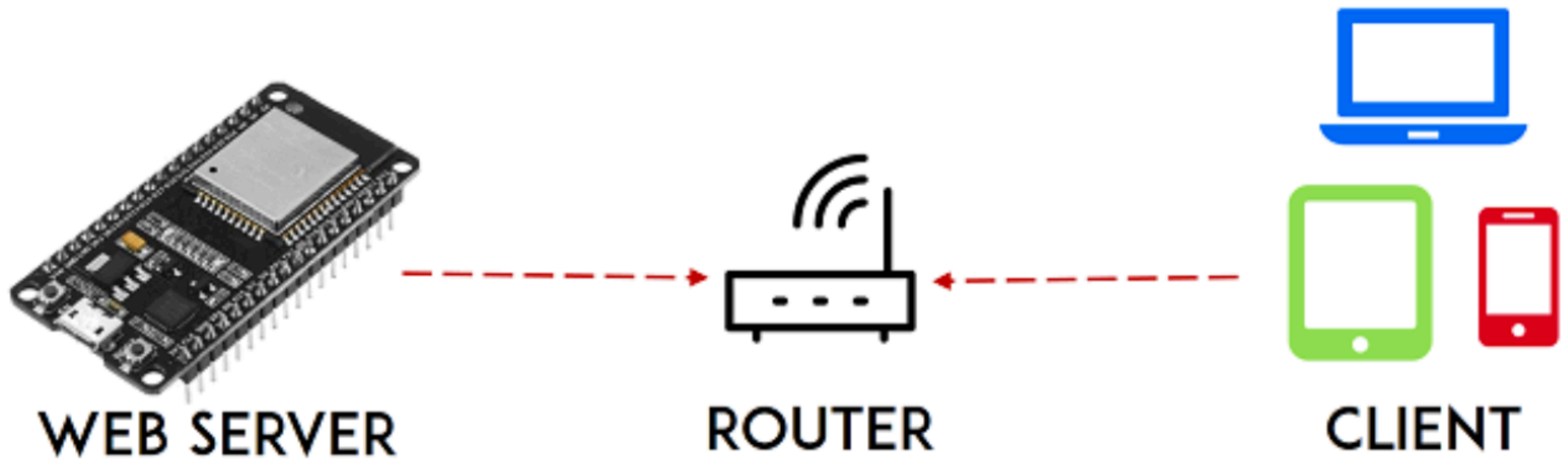
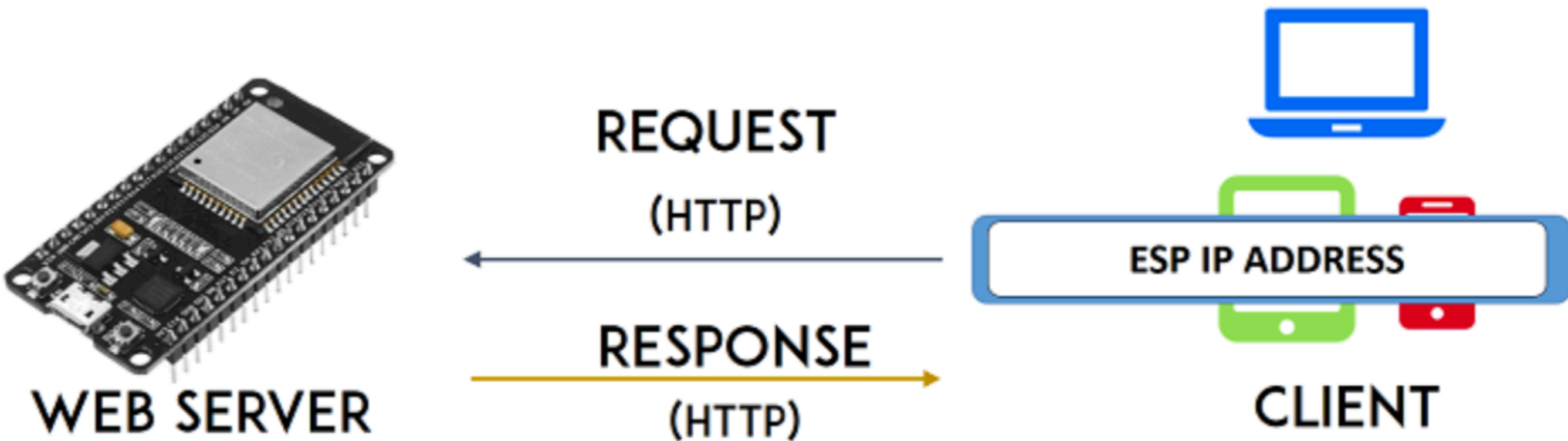
Correction

Accès en wifi



- Pour pouvoir se connecter grâce au réseau wifi qui partage les connexions aux cartes ESP32, il faut ensuite sélectionner le wifi en question et entrer l'adresse IP du ESP32 concerné.





Affichage de l'adresse IP

```
try:
    import usocket as socket
except:
    import socket
import network
import esp
esp.osdebug(None)
import gc
gc.collect()
ssid = 'NOM DU WIFI'
password = 'Mot de passe du WIFI'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
    pass
print('Connection successful')
print(station.ifconfig())
```

Socket : outil de connexion réseau

importation de la bibliothèque socket

importation de la bibliothèque réseau

Garbage Collector : nettoyage mémoire

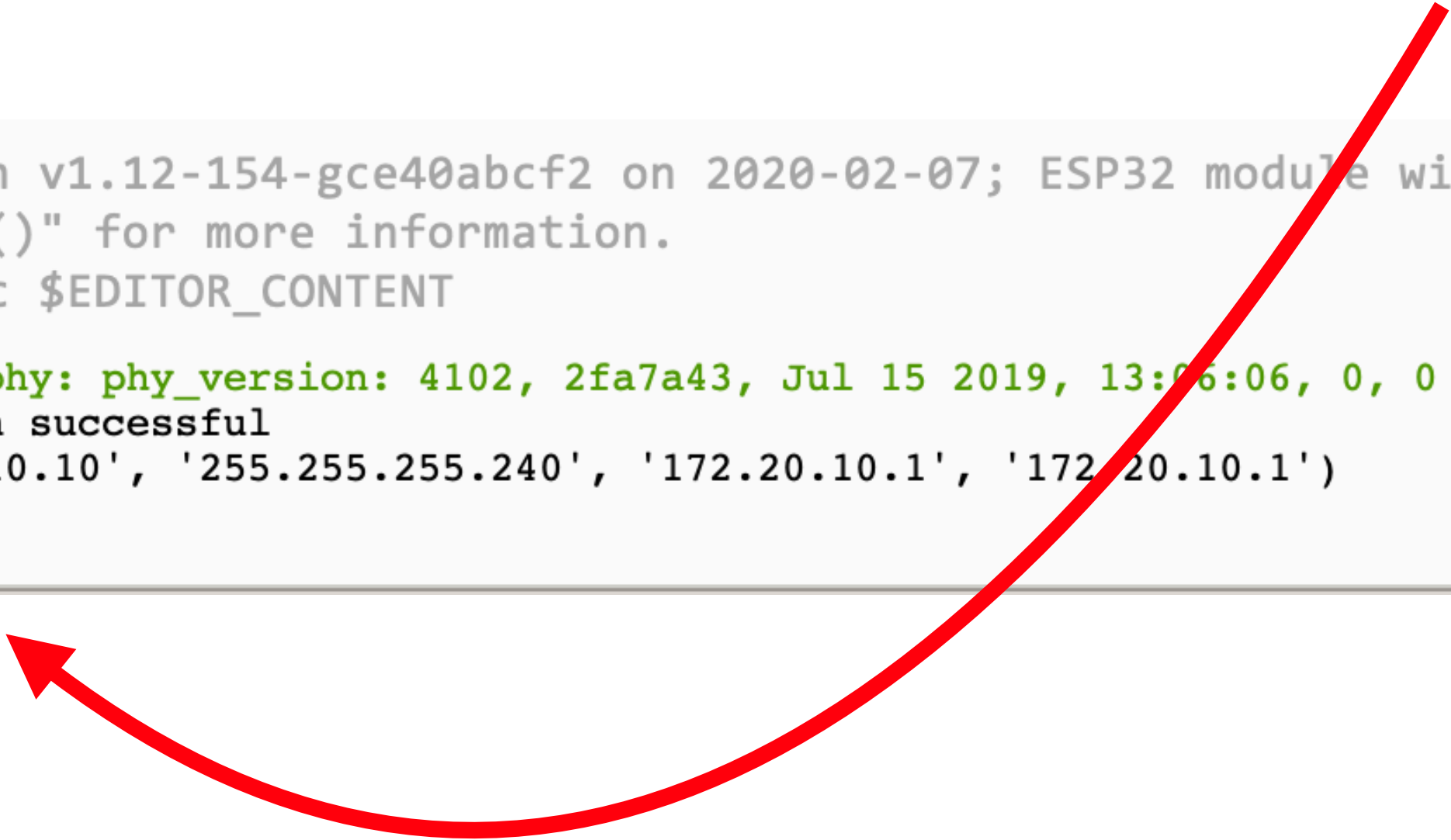
paramètre d'un WIFI existant

Affichage de l'adresse IP

```
MicroPython v1.12-154-gce40abcf2 on 2020-02-07; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

I (7347) phy: phy_version: 4102, 2fa7a43, Jul 15 2019, 13:06:06, 0, 0
Connection successful
('172.20.10.10', '255.255.255.240', '172.20.10.1', '172.20.10.1')

>>>
```



Serveur WEB

Contenu de la page WEB

```
def web_page():  
    html = """<html><head><meta name="viewport" content="width=device-  
width, initial-scale=1"></head>  
<body><h1>Bonjour eleves de l'ENIM !</h1></body></html>"""  
    return html
```

Utilisation de la socket pour se connecter au réseau

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
s.bind(('', 80))
```

Utilisation de l'adresse IP locale et du port d'écoute

```
s.listen(5)
```

Ecoute de la socket

```
while True:
```

```
    conn, addr = s.accept()  
    print( 'Connection de %s' % str(addr))  
    request = conn.recv(1024)  
    print( 'Contenu = %s ' % str(request))  
    response = web_page()  
    conn.send('HTTP/1.1 200 OK\n')  
    conn.send('Content-Type: text/html\n')  
    conn.send('Connection: close\n\n')  
    conn.sendall(response)
```

Affichage des connexions

Serveur WEB

Affichage des connexions

```
MicroPython v1.12-154-gce40abcf2 on 2020-02-07; ESP32 module with ESP32
Type "help()" for more information.
```

```
>>> %Run -c $EDITOR_CONTENT
```

```
I (4457) phy: phy_version: 4102, 2fa7a43, Jul 15 2019, 13:06:06, 0, 0
```

```
Connection successful
```

```
('172.20.10.10', '255.255.255.240', '172.20.10.1', '172.20.10.1')
```

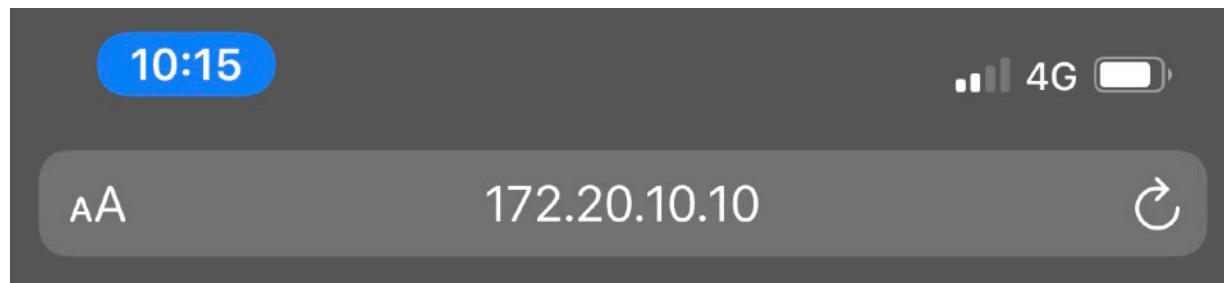
```
Connection de ('172.20.10.1', 49493)
```

```
Contenu = b'GET / HTTP/1.1\r\nHost: 172.20.10.10\r\nUpgrade-Insecure-Requests: 1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nUser-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Mobile/15E148 Safari/604.1\r\nAccept-Language: fr-fr\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n'
```

Adresse IP ESP32

Adresse IP Client smartphone

Accès au serveur web



Bonjour élèves de l'ENIM !

Comme actuellement un serveur web tourne sur la carte ESP, on peut lancer un navigateur web et taper l'adresse IP de l'ESP32 pour s'en assurer.

Un serveur web permet de lire des données à distance, mais également de lancer des actions comme nous le verrons.

Serveur web + LED

```
from machine import Pin
from time import sleep
led = Pin(2, Pin.OUT)
```

Au début du code

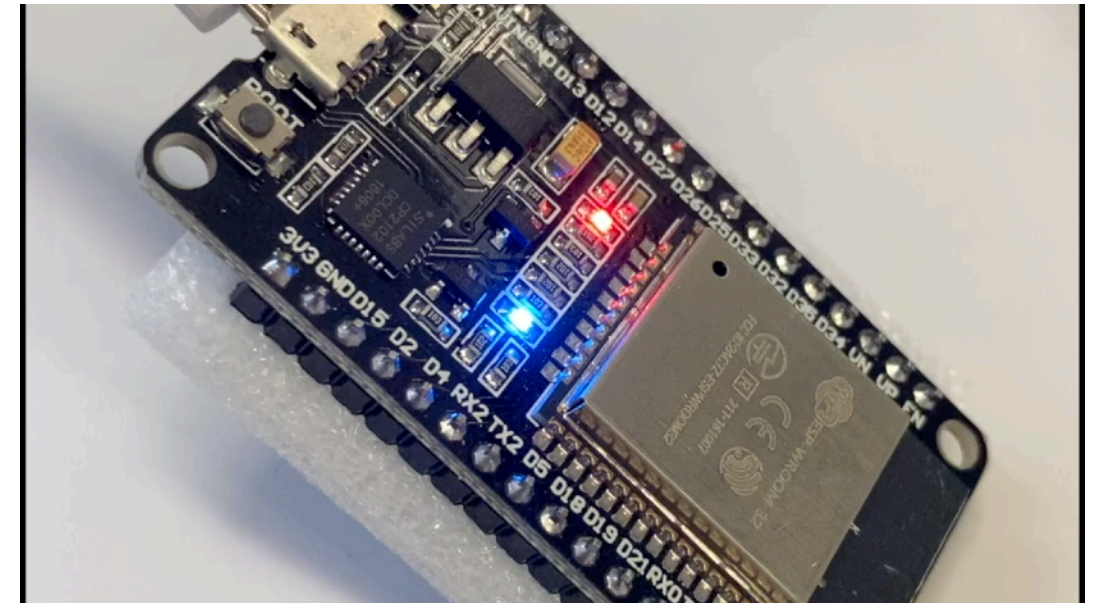
```
i=0
while (i<10):
```

À la fin du code

```
    led.value(not led.value())
```

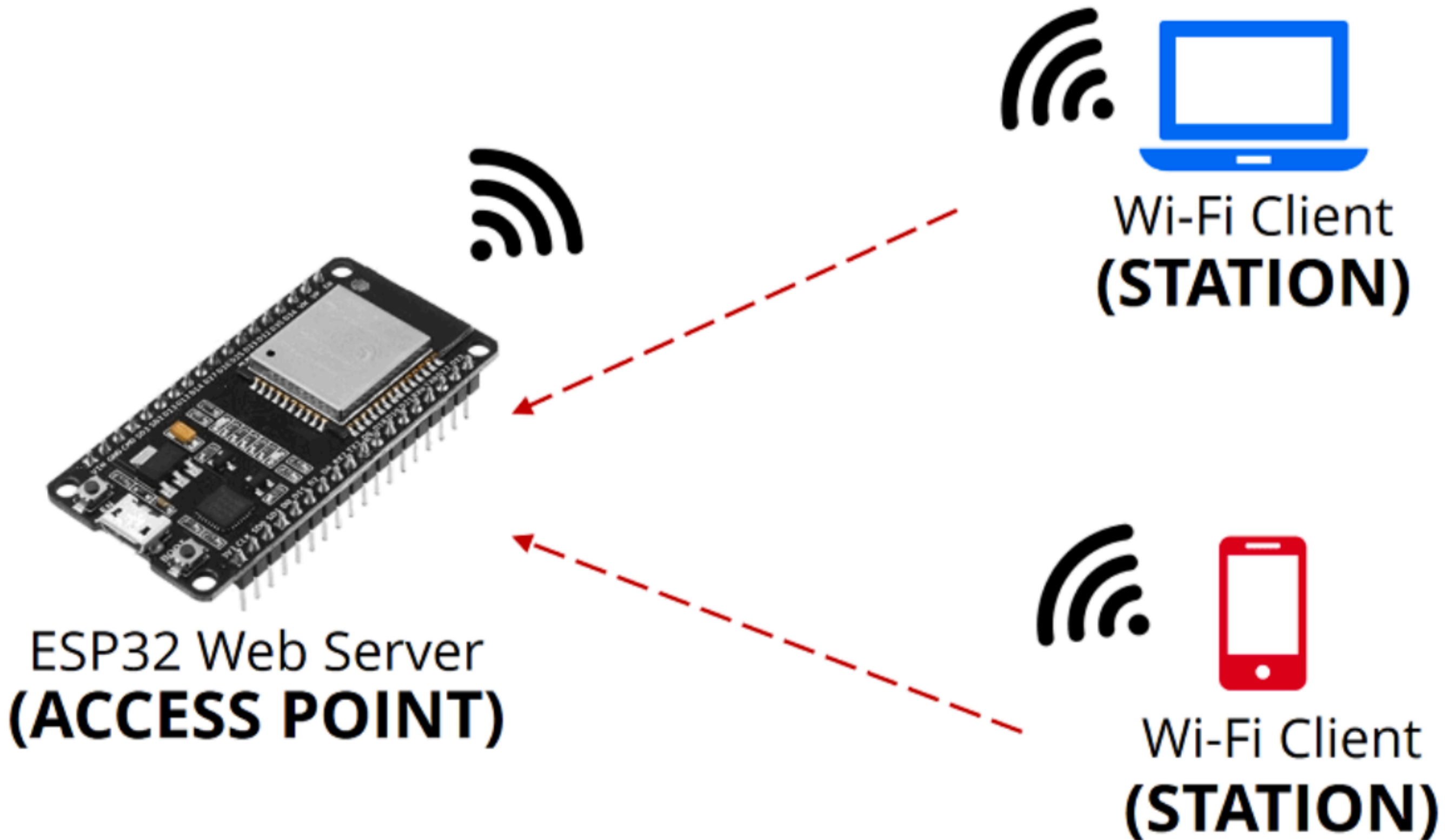
```
    sleep(1.0)
```

```
    i=i+1
```



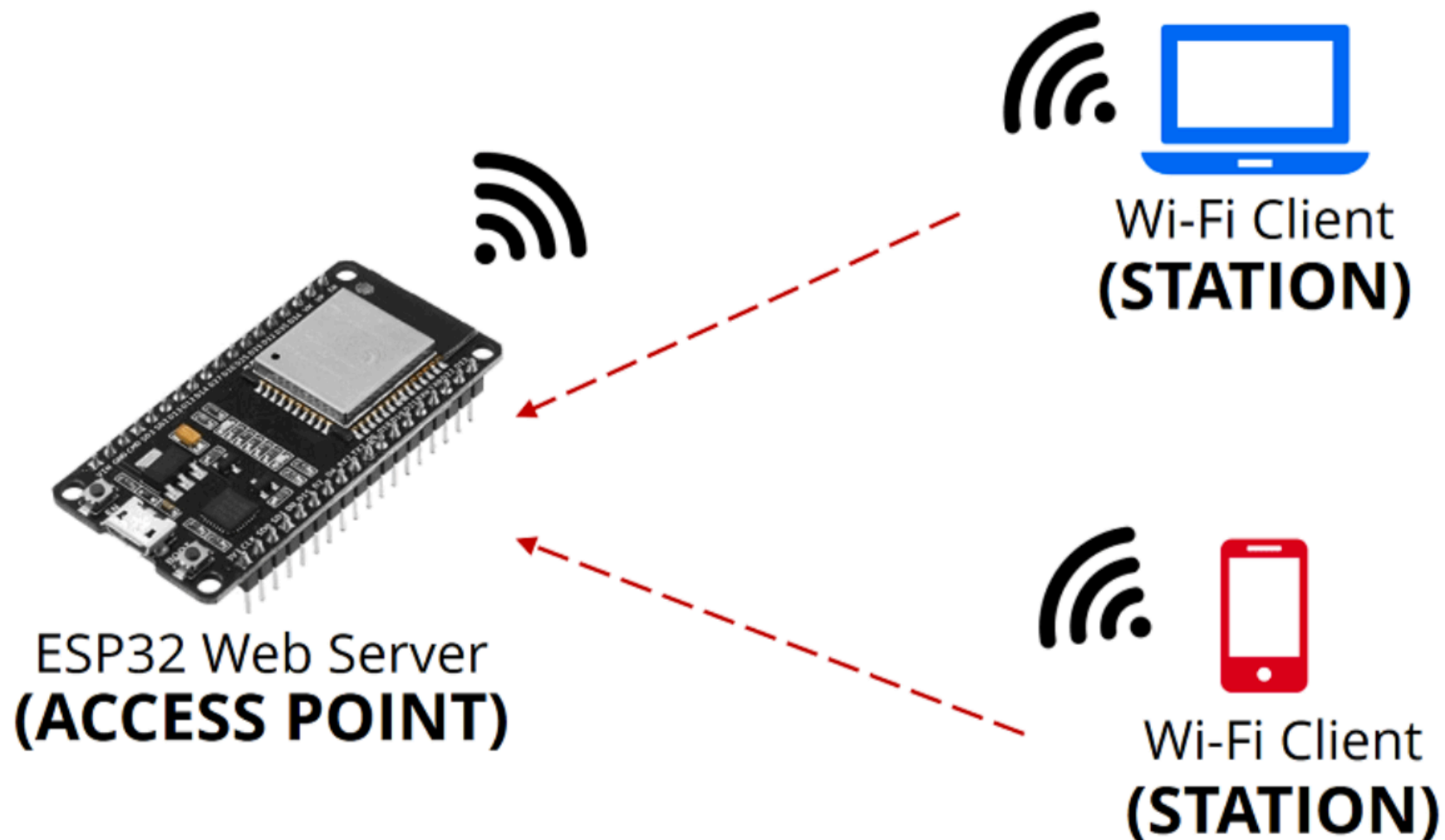
**La LED va clignoter
combien de fois ?**

Point d'accès



Point d'accès WIFI

L'ESP32 est capable de devenir un point d'accès WIFI c'est à dire de créer une zone WIFI dont il déterminera son nom (SSID) et le mot de passe.



Point d'accès WIFI

```
ssid = 'ESP32ENIM'
password = '123456789'
ap = network.WLAN(network.AP_IF)
ap.active(True)
ap.config(essid=ssid, password=password)
while ap.active() == False:
    pass
print('Connection successful')
print(ap.ifconfig())
def web_page():
    html = """<html><head><meta name="viewport"
content="width=device-width, initial-scale=1"></head>
<body><h1>ESP 32 ENIM Access Point !</h1></body></
html>"""
    return html
```

paramètre du point d'accès WIFI

Création du point d'accès

Affichage des connexions

Contenu de la page WEB

Point d'accès WIFI

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
while True:
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    print('Content = %s' % str(request))
    response = web_page()+ '<h1>Connecte a : <br>Adresse  
IP, numero de port : <br>' + str(addr)+ '</h1>'
    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.send(response)
    conn.close()
```

Ecoute de la socket

Affichage des connexions

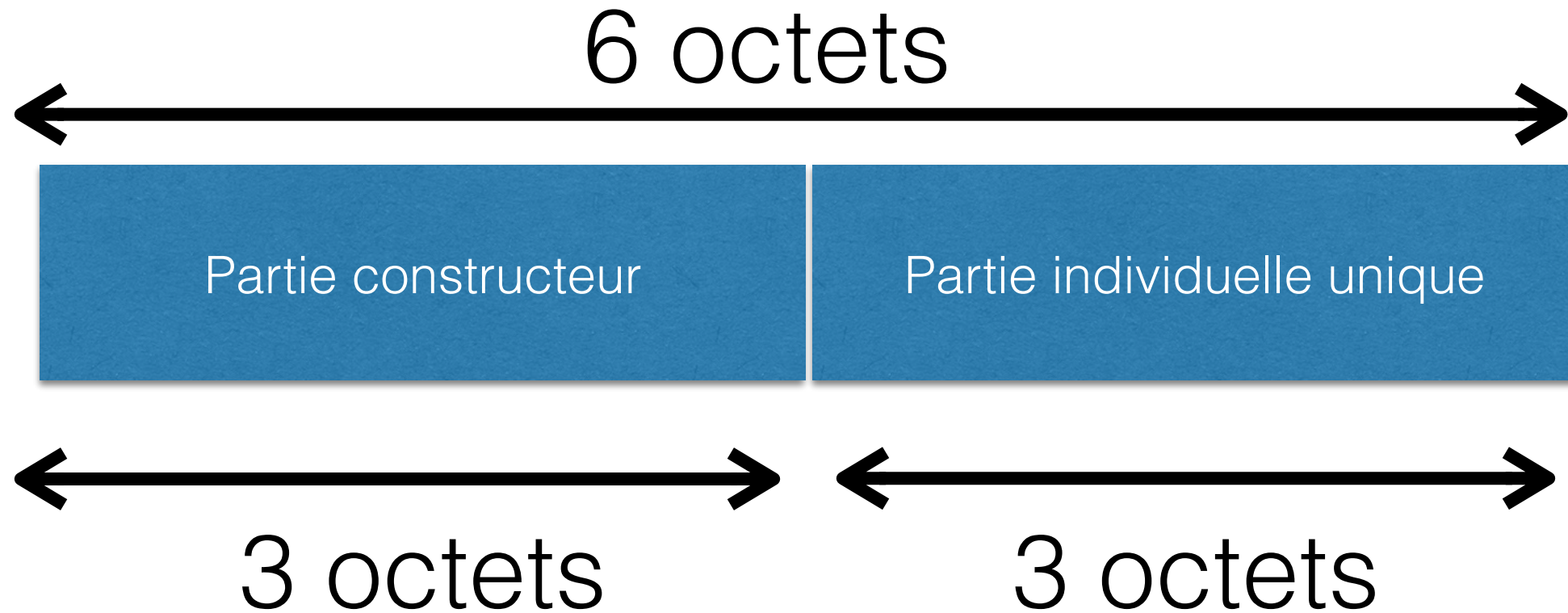
Adresse MAC

- Nous allons à présent encore approfondir nos connaissances
- A côté de l'adresse IP il y a une autre adresse importante : c'est l'adresse physique, encore appelée adresse MAC.

Adresse MAC

- Une adresse MAC (Media Access Control), aussi appelée adresse physique, est stockée dans l'interface réseau.
- Elle est unique au monde.
- Toutes les interfaces réseau ont une adresse MAC, même celles contenues dans les PC et autres appareils connectés (tablette tactile, smartphone, consoles de jeux, ...).

Adresse MAC

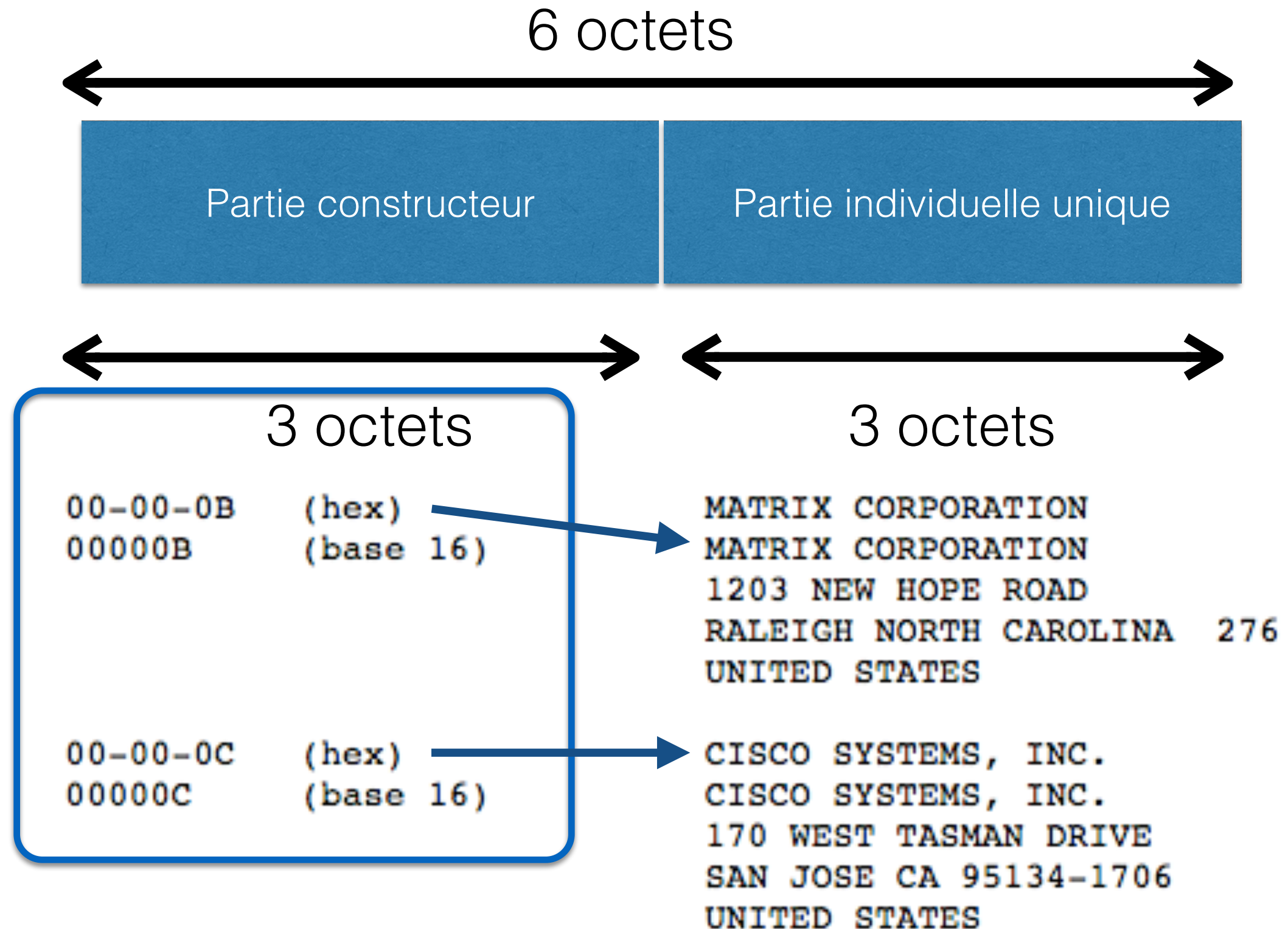


Une adresse MAC est constituée de 48 bits (6 octets) et est généralement représentée sous la forme hexadécimale en séparant les octets par un double point ou un tiret.

Par exemple : **3 octets** **3 octets**
5E:FF:56:A2:AF:15.

The example MAC address "5E:FF:56:A2:AF:15." is shown with two blue double-headed arrows below it. The first arrow is under the first three octets "5E:FF:56" and the second arrow is under the last three octets "A2:AF:15", visually demonstrating the 3-octet halves mentioned in the text above.

Adresse MAC



Affichage de l'adresse MAC

```
try:
    import usocket as socket
except:
    import socket
import network
import esp
esp.osdebug(None)
import gc
gc.collect()
ssid = 'SSID'
password = 'mot de passe'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
    pass
print('Connection successful')
print(station.ifconfig())
mac = station.config('mac')
smac = ""
for ot in list(mac): h = hex(ot); smac += h[2] + h[3]
print("Adresse MAC : " + smac)
```



Partie commune

Résultat : Adresse MAC : a4cf1244a414

Affichage de l'adresse IP sur PC

- Si on cherche l'adresse l'adresse IPv4 de la carte Ethernet du PC Windows :

```
C:\Users\jms>ipconfig
```

```
Configuration IP de Windows
```

```
Carte Ethernet Ethernet :
```

```
Suffixe DNS propre à la connexion. . . : localdomain
Adresse IPv6. . . . . : fdb2:2c26:f4e4:0:f868:3cd9
Adresse IPv6 temporaire . . . . . : fdb2:2c26:f4e4:0:1019:66ef
Adresse IPv4 de liaison local. . . . . : 10.211.55.5
Adresse IPv4. . . . . : 10.211.55.5
Masque de sous-réseau. . . . . : 255.255.255.0
Passerelle par défaut. . . . . : 10.211.55.1
```

Ici 10.221.55.5

```
Carte Tunnel isatap.localdomain :
```

Affichage de l'adresse IP sur PC

- Sur MacOS on peut regarder dans le menu « Réseau » des « Préférences Système »
- Sur Linux ou Raspbian, on ouvre une fenêtre de terminal et on utilise la commande « **sudo ifconfig** » ou "**ip addr show**" et on entre le mot de passe lorsqu'il est demandé



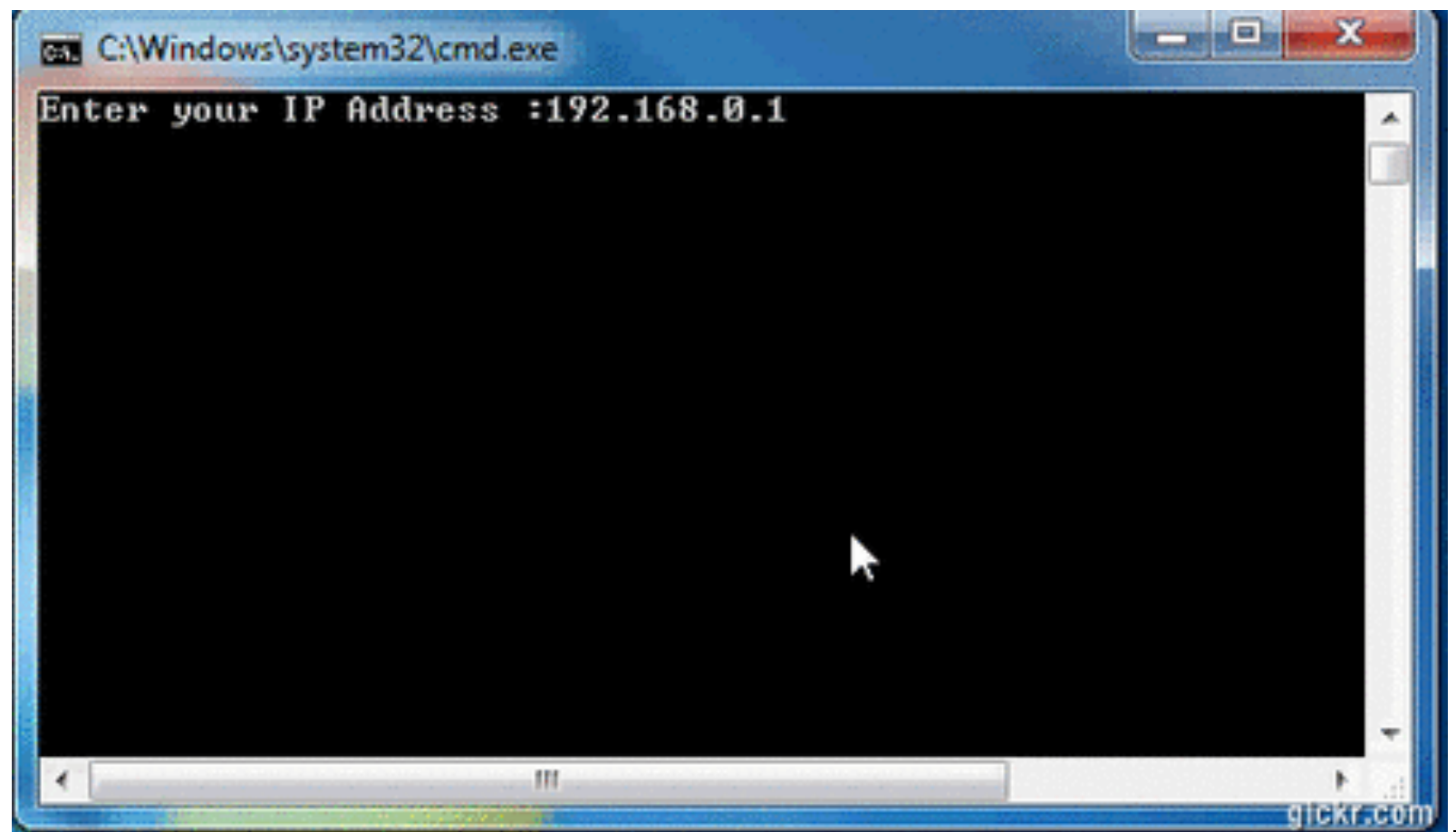
```
tecmint@tecmint ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr 28:d2:44:eb:bd:98
          inet addr:192.168.0.104 Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2ad2:44ff:feeb:bd98/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:342087 errors:0 dropped:0 overruns:0 frame:0
          TX packets:233764 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:406375041 (406.3 MB)  TX bytes:25096967 (25.0 MB)
```

Commande Ping



- La commande « ping » permet de savoir si le module Arduino est joignable à travers l'adresse réseau choisie, si oui elle répondra à cette commande.

C'est une
commande de
test importante
pour les réseaux



Commande Ping

- Exemple de commande ping avec les réponses :

```
C:\>ping 91.198.174.225
```

```
Ping 213.248.93.86 avec 32 octets de données :
```

```
Réponse de 213.248.93.86 : octets=32 temps<10ms TTL=53
```

```
Réponse de 213.248.93.86 : octets=32 temps<10ms TTL=53
```

```
Réponse de 213.248.93.86 : octets=32 temps<10ms TTL=53
```

```
Réponse de 213.248.93.86 : octets=32 temps<10ms TTL=53
```



Utilisation de boutons



Utilisation de boutons

- L'utilisation de boutons à l'aide de formulaires HTML permet de commander facilement la LED. Voici comment nous pouvons les utiliser :
- **`<a href=\"?led=on\"><button>ON</button></a>`**
- va créer un bouton « ON »

Utilisation de boutons

- **`<a href=\"?led=on\"><button>ON</button>`**
- a : balise HTML de lien
- href : paramètre passé au lien, ici led=on, ce qui va allumer la LED
- button : on crée un bouton ON

Utilisation de boutons

- **`<a href=\"?led=off\"><button>OFF</button></a>`**
- va créer un bouton « OFF »
- Ici on passe le paramètre led à off ce qui va éteindre la LED

CODE BOUTONS

```
request = conn.recv(1024)
```

```
request = str(request) On lit les paramètres reçus
```

```
print('Content = %s' % request)
```

```
led_on = request.find('/?led=on') On recherche
```

```
led_off = request.find('/?led=off') 'on' ou 'off'
```

```
if led_on == 6:
```

```
    print('LED ON')
```

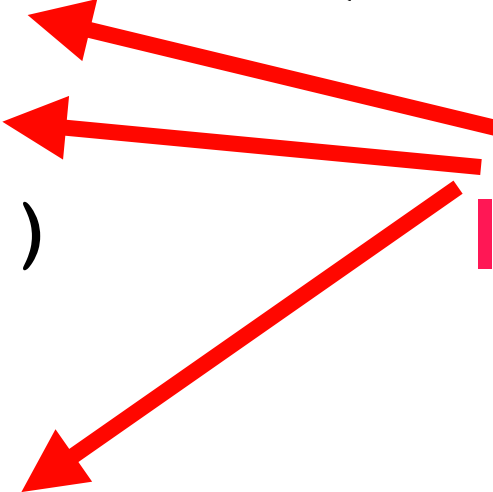
```
    led.value(1)
```

```
if led_off == 6:
```

```
    print('LED OFF')
```

```
    led.value(0)
```

**fonction qui retourne
la position de la chaîne**



**On allume ou éteint la LED
en fonction du bouton appuyé**

RÉSULTAT

```
I (4538) phy: phy_version: 4102, 2fa7a43, Jul 15 2019, 13:06:06, 0, 0
```

```
Connection successful
```

```
('172.20.10.10', '255.255.255.240', '172.20.10.1', '172.20.10.1')
```

```
Got a connection from ('172.20.10.1', 50644)
```

```
Content = b'GET / HTTP/1.1\r\nHost: 172.20.10.10\r\nUpgrade-Insecure-Request: 1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nUser-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Mobile/15E148 Safari/604.1\r\nAccept-Language: fr-fr\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n'
```

```
Got a connection from ('172.20.10.1', 50645)
```

```
Content = b'GET /?led=on HTTP/1.1\r\nHost: 172.20.10.10\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nUser-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Mobile/15E148 Safari/604.1\r\nReferer: http://172.20.10.10/\r\nAccept-Language: fr-fr\r\nAccept-Encoding: gzip, deflate\r\n\r\n'
```

```
LED ON
```

```
Got a connection from ('172.20.10.1', 50646)
```

```
Content = b'GET /favicon.ico HTTP/1.1\r\nHost: 172.20.10.10\r\nConnection: keep-alive\r\nAccept: */*\r\nUser-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Mobile/15E148 Safari/604.1\r\nAccept-Language: fr-fr\r\nReferer: http://172.20.10.10/?led=on\r\nAccept-Encoding: gzip, deflate\r\n\r\n'
```

```
Got a connection from ('172.20.10.1', 50649)
```

```
Content = b'GET /?led=off HTTP/1.1\r\nHost: 172.20.10.10\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nUser-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) V
```

LECTURE PAGE

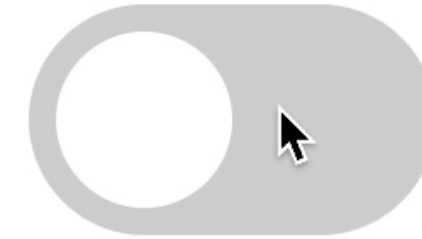
APPUI SUR ON

APPUI SUR OFF

Slider

ESP ENIM Web Server

```
def web_page():
    if led.value() == 1:
        led_state = 'checked'
    else:
        led_state = ""
    html = """<html><head><meta name="viewport" content="width=device-width, initial-
scale=1"><style>
    body{max-width: 300px; margin: 0px auto;}
    .switch{position:relative;display:inline-block;width:120px;height:68px}.switch
input{display:none}
    .slider{position:absolute;top:0;left:0;right:0;bottom:0;background-color:#ccc;border-
radius:34px}
    .slider:before{position:absolute;content:"";height:52px;width:52px;left:8px;bottom:
8px;background-color:#fff;-webkit-transition:.4s;transition:.4s;border-radius:68px}
    input:checked+.slider{background-color:#2196F3}
    input:checked+.slider:before{-webkit-transform:translateX(52px);-ms-
transform:translateX(52px);transform:translateX(52px)}
    </style><script>function toggleCheckbox(element) { var xhr = new XMLHttpRequest();
if(element.checked){ xhr.open("GET", "/?led=on", true); }
    else { xhr.open("GET", "/?led=off", true); } xhr.send(); }</script></head><body>
    <h1>ESP ENIM Web Server</h1><label class="switch"><input type="checkbox"
onchange= toggleCheckbox(this) %s><span class="slider">
    </span></label></body></html>""" % (led_state)
```



Résultat



**On allume ou éteint la LED
en fonction de la position du SLIDER**

Sauvegarde des données dans les fichiers



Fichiers avec Python

Les fichiers sur ESP32 avec MicroPython fonctionnent comme sur ordinateur avec Python :

```
f = open("fichier.txt", "a")  
f.write("\nExemple de contenu dans un fichier \n")  
f.close()
```

Saut de ligne



```
# Ouverture et lecture du fichier après ouverture :  
f = open("fichier.txt", "r")  
print(f.read())
```

Résultat :

```
Exemple de contenu dans un fichier
```

Fichiers avec Python

Liste des fichiers présents sur l'ESP32 :

```
import os  
print(os.listdir())
```

Résultat :

```
['boot.py', 'test_blink.py', 'serverLed_2.py', 'blink led.py', 'blink led2.py']
```

IFTTT

« IF This Then That »



IFTTT « IF This Then That »

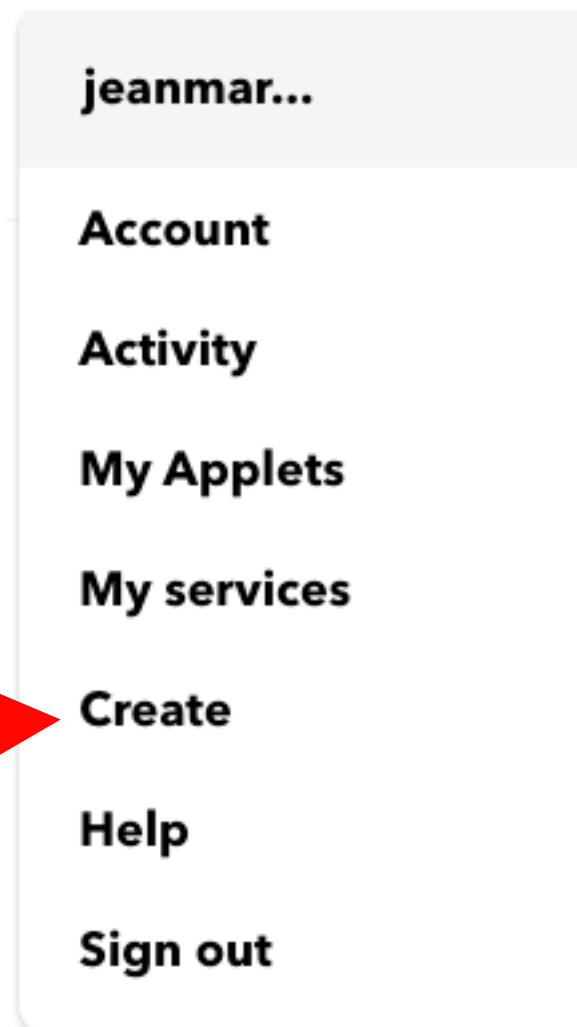


- IFTTT est un service web gratuit permettant à ses utilisateurs de créer des chaînes d'instruction simples appelées applets.
- Déclenchée par des services web
- Par exemple une applet peut envoyer un email en cas d'événement : température anormale détectée, franchissement de porte, etc...

IFTTT « IF This Then That »



- Nous allons envoyer un email sur événement depuis l'ESP32.
- Pour cela il faut créer un compte sur IFTTT avec l'email qui devra recevoir le message.
- A partir du compte, aller sur "Create" →



Create your own

If  **This Then That**



- Cliquer sur + : création de la condition déclenchante

Choose a service

Step 1 of 6

Q webhook



Webhooks

Webhook est un service permettant d'exécuter une action précise par appel d'une page web (URL)

< Back



Choose trigger

Step 2 of 6

Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Cliquez dans ce cadre

< Back



Complete trigger fields

Step 2 of 6

Event Name

ESP32 ENIM

The name of the event, like "button_pressed" or "front_door_opened"

Create trigger

Ecrire le nom
de l'événement

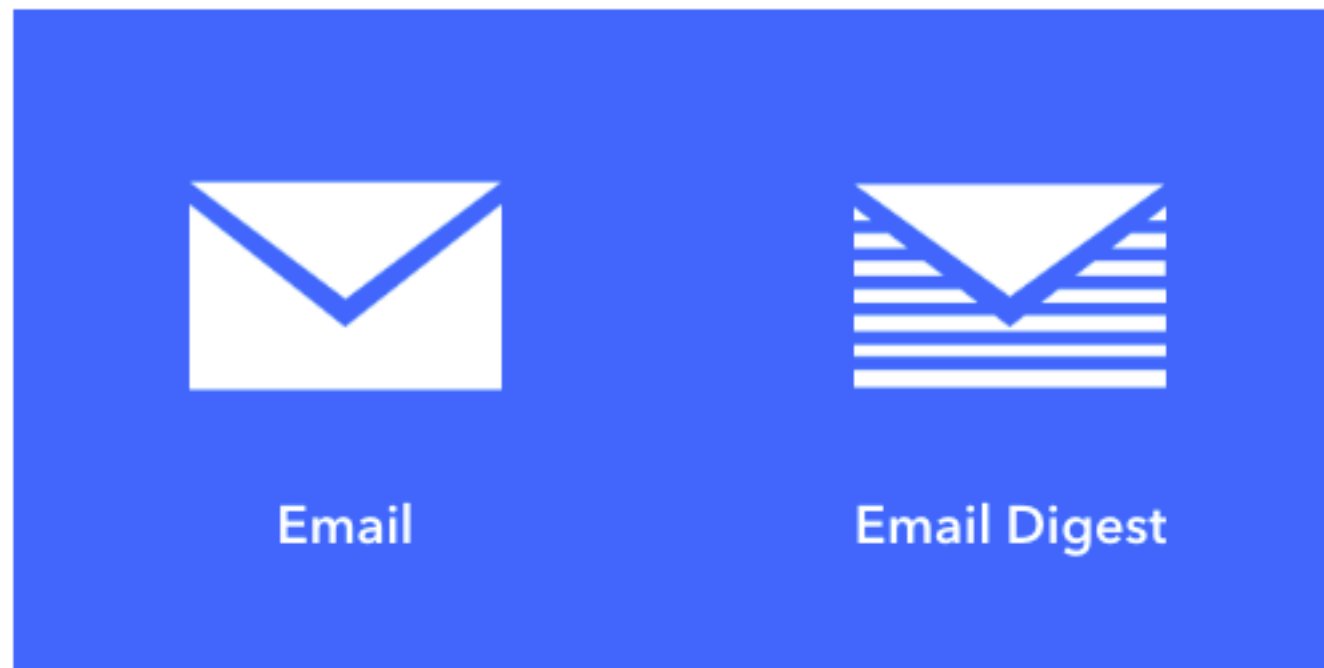
If  Then  That

Cliquez sur +
pour définir
l'action à effectuer
en cas d'événement



Choose action service

Step 3 of 6



**Ecrire email
pour sélectionner
le service Email**

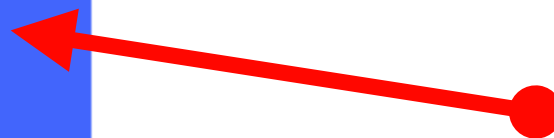


Choose action

Step 4 of 6

Send me an email

This Action will send you an HTML based email. Images and links are supported.



Cliquez dans ce cadre



Complete action field

Step 5 of 6

Subject

The event named
" **EventName** " occurred on
the Maker Webhooks service

Add ingredient

Body

What: **EventName**

When: **OccurredAt**

Extra Data: **Value1** ,
Value2 , **Value3** ,

Add ingredient

Create action

Review and finish

Step 6 of 6



If Maker Event "ESP32
ENIM", then Send me an
email at
jeanmariesteinmetz@gm
ail.com

82/140

by jeanmariesteinmetz

Receive notifications
when this Applet runs



Finish

Explore

🔍 webhooks

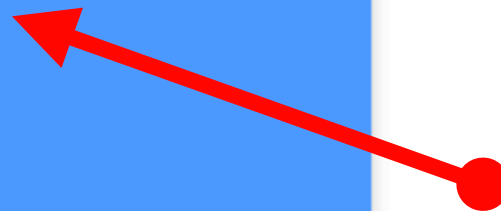


Connections

Services

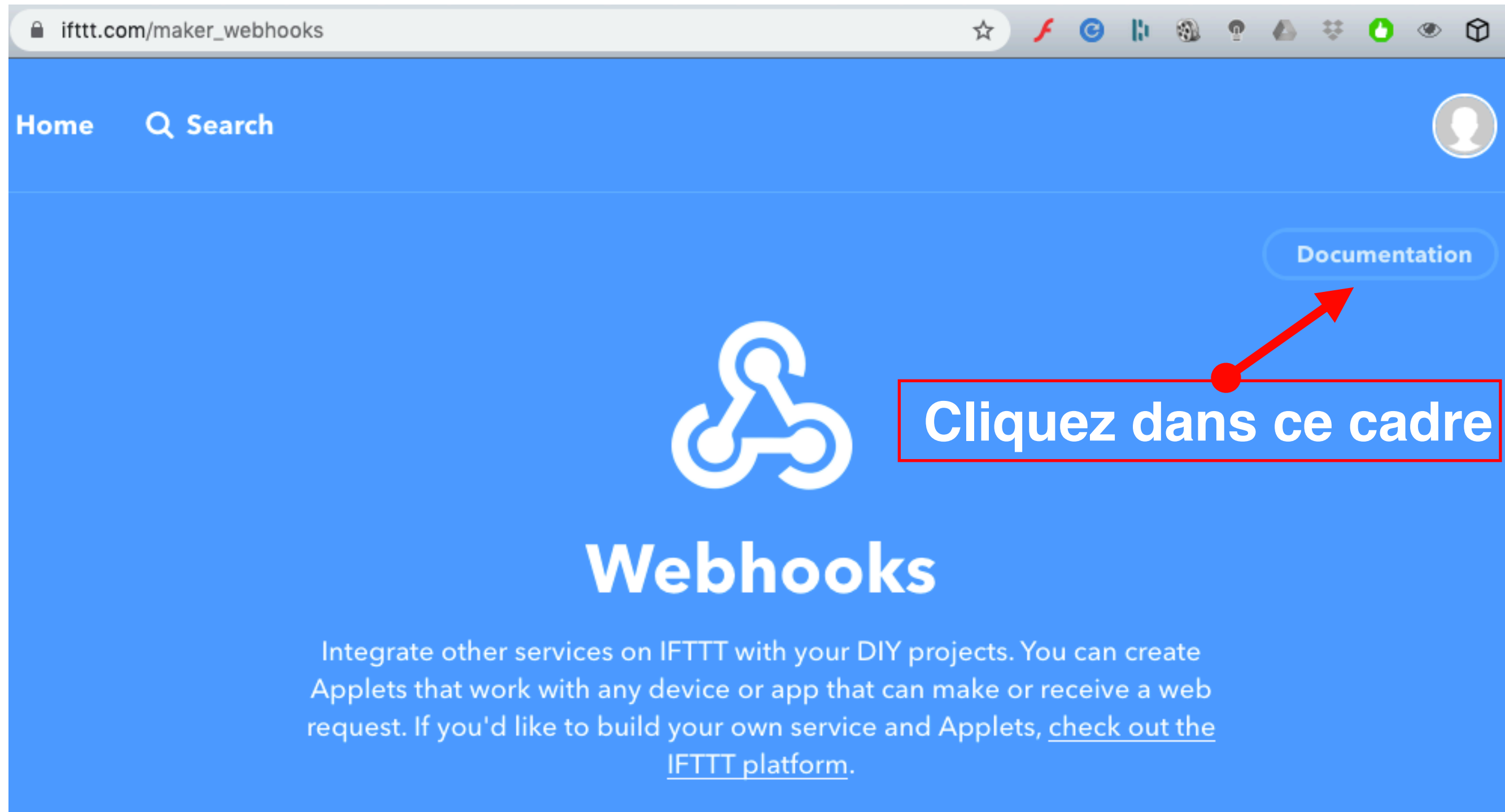


Webhooks



**Cliquez dans ce cadre
pour définir
l'URL de la requête**

Pour récupérer la clé : KEY



KEY : clé personnelle

Your key is:

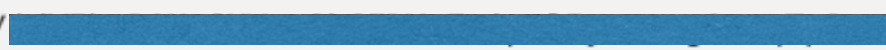


La requête est ici, mettre le nom de l'événement :

To trigger an Event

Make a POST or GET web request to:

`https://maker.ifttt.com/trigger/{event}/with/key/`



With an optional JSON body of:

```
{ "value1" : "[ ]", "value2" : "[ ]", "value3" : "[ ]" }
```

The data is completely optional, and you can also pass `value1`, `value2`, and `value3` as query parameters or form variables. This content will be passed on to the Action in your Recipe.

You can also try it with `curl` from a command line.

```
curl -X POST https://maker.ifttt.com/trigger/{event}/with/key/
```



Test It

CODE IFTTT

```
try:
    import usocket as socket
except:
    import socket
import network
import esp
import urequests
esp.osdebug(None)
import gc
gc.collect()
ssid = 'SSID'
password = 'motdepasse'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
    pass
print('Connection successful')
print(station.ifconfig())

r = urequests.post('https://maker.ifttt.com/trigger/ESP/with/
key/[REDACTED]?
value1=ESP&value2=ENIM&value3=Cours')
print(r.status_code)
print(r.text)
```

Partie commune

KEY : clé personnelle

**Paramètres passés :
informations reçues dans l'email**

Activity



Applet ran

Feb 11 - 4:36 PM



If Maker Event "ESP", then Send me an email at jeanmariesteinmetz@gmail.com

The event named "ESP" occurred on the Maker Webhooks service

^ Hide details

Ingredients

EventName

ESP

OccurredAt

February 11, 2020 at 04:36PM

Value1

ESP

Value2

ENIM

Value3

Cours

RÉSVLTAT

The event named "ESP" occurred on the Maker Webhooks service



Webhooks via IFTTT <action@ifttt.com>

À moi ▾

🌐 anglais ▾ > français ▾ [Traduire le message](#)

What: ESP

When: February 12, 2020 at 10:37AM

Extra Data: ESP, ENIM, Cours,



If Maker Event "ESP", then Send me an email at jeanmariesteinmetz@gmail.com



IFTTT

Domotique

- Si l'on désire piloter sa maison ou son appartement à l'aide de l'ESP32, il suffit de l'intégrer au réseau du domicile derrière la box du fournisseur d'accès.
- A l'aide d'une redirection de port, on peut alors accéder à la carte ESP32 de l'extérieur de la maison.
- Pour éviter que n'importe qui pilote « sa » maison, il est alors préférable de mettre un mot de passe pour valider l'accès.

Extérieur

IP Publique:port/motdepasse=ENIM
exemple de port : 20000

INTERNET

Détection du mot de
passe comme la commande de
la LED

Domicile

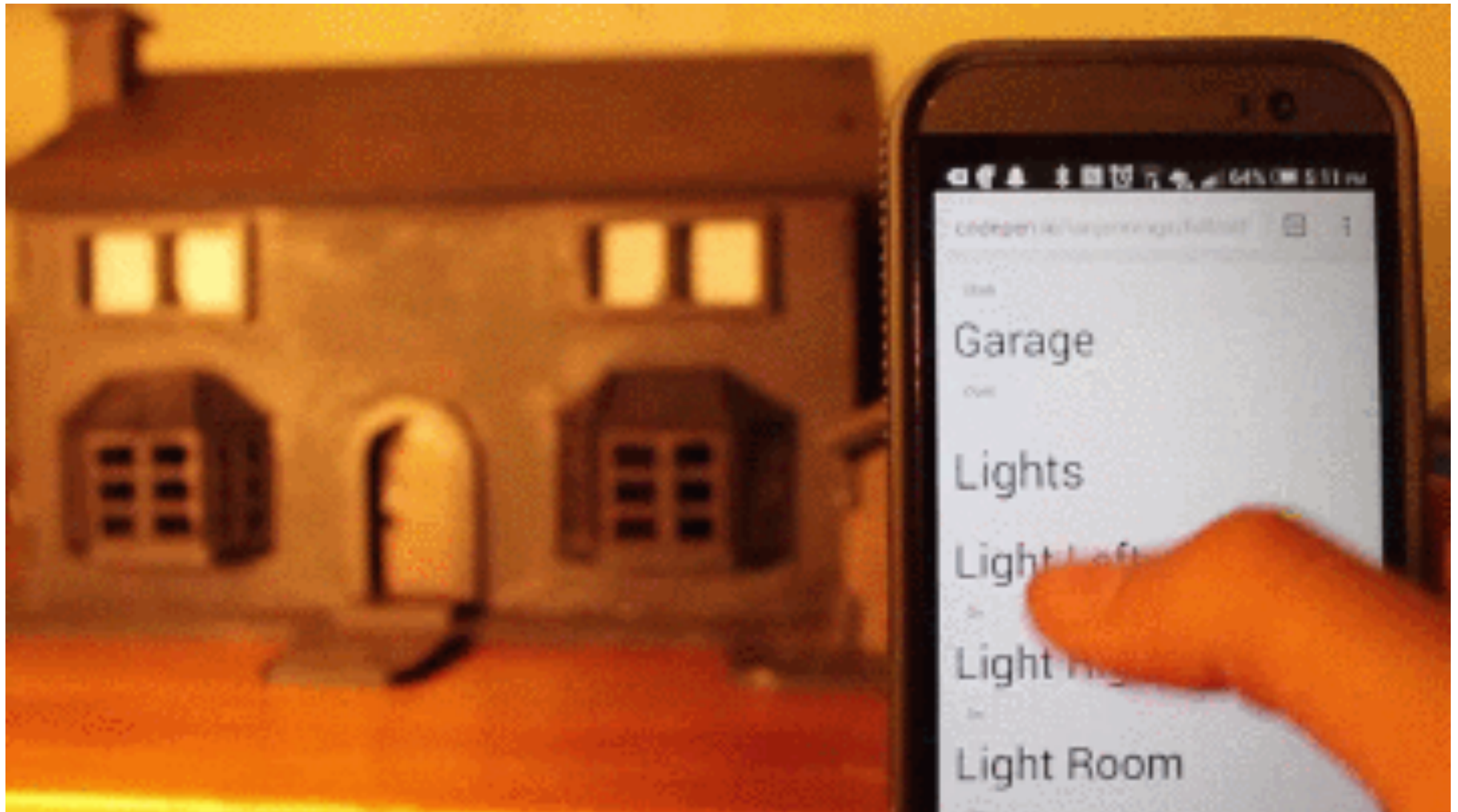
BOX

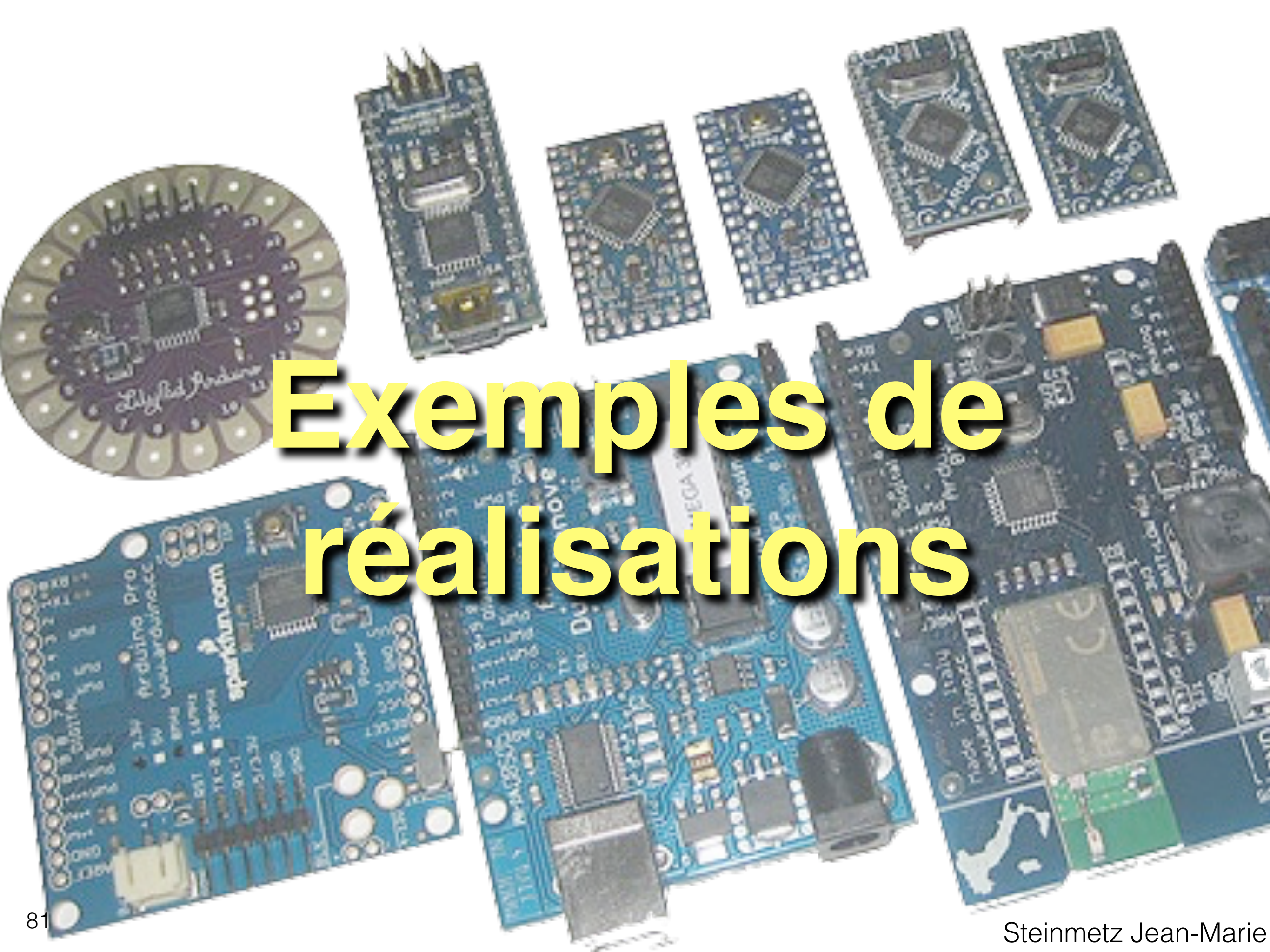
Réseau

ESP32

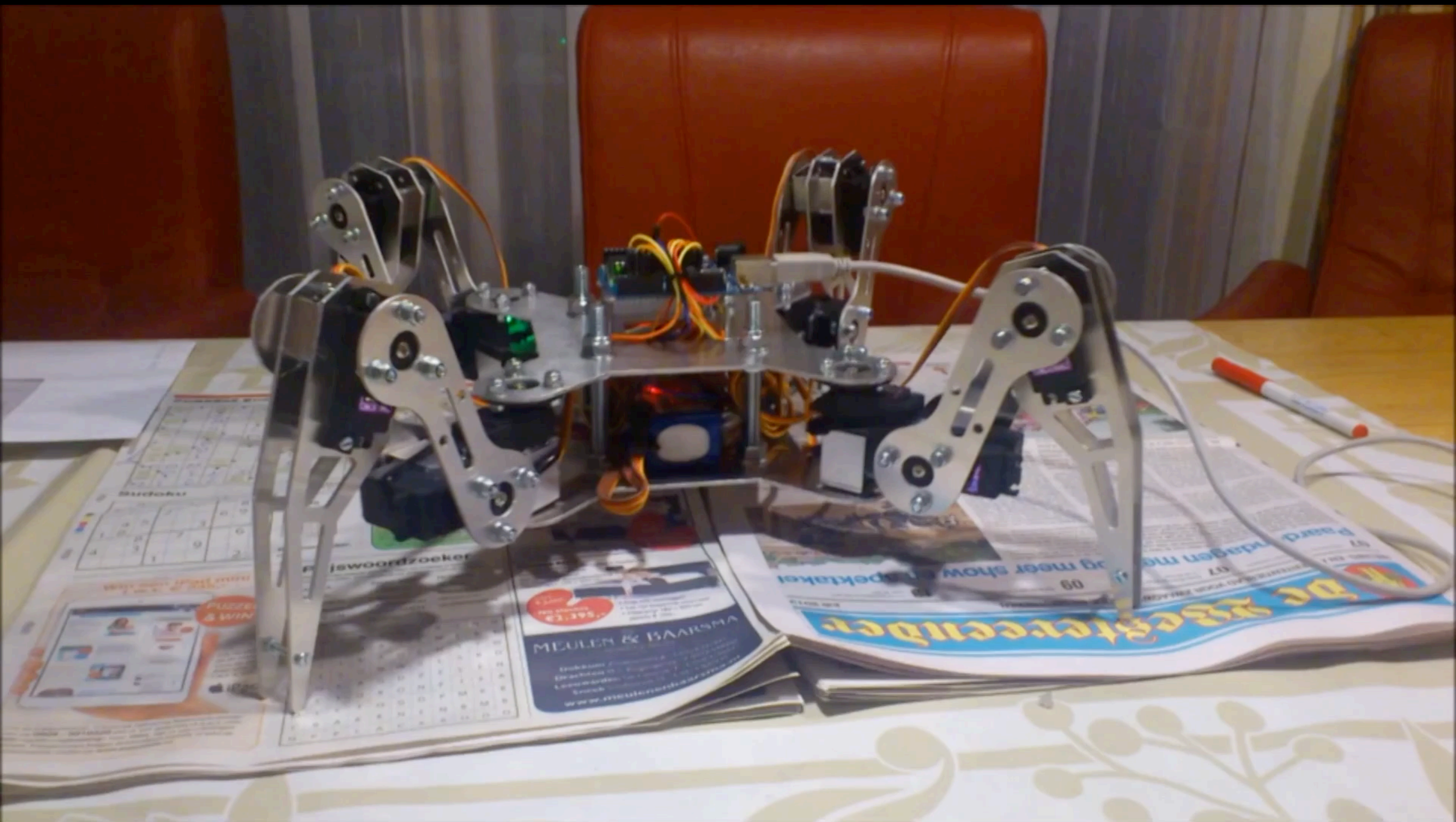
Adresse IP ESP32:80/motdepasse=ENIM

Exemple : domotique avec smartphone et commande à distance





Exemples de réalisations

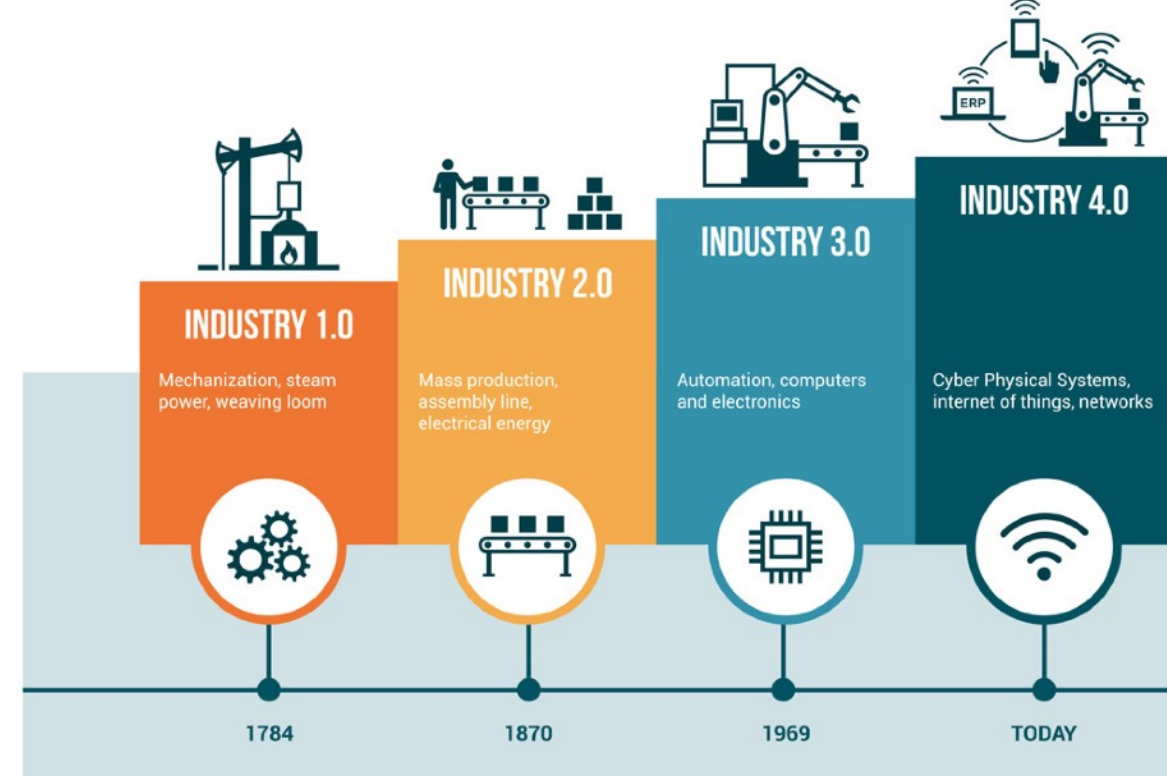


Exemples de réalisations mécatronique



Contrôle d'une voiture

Dans vos projets



Actuellement, tous les industriels travaillent sur des développements liés à l'industrie du futur. Cette 4ème révolution industrielle (1ère mécanisation, 2ème production de masse, 3ème automatisation) repose sur l'usine intelligente et une interconnexion des machines et des systèmes.

Un petit film
pour présenter
la 4ème
révolution technologique

Vous avez les outils
pour être acteur de
cette révolution !

Merci de votre
attention