

	<h1 style="text-align: center;">Carte IOT</h1> <p style="text-align: center;">(Description technique pour d'espruino)</p>	
---	---	---

LE CREN Anthony

Espruino a été créé par Gordon Williams en 2012 dans le but de rendre le développement de microcontrôleurs multiplateforme. Le firmware Espruino a été rendu open source en 2013. Depuis la carte Espruino d'origine, il y a eu un certain nombre de nouvelles cartes de développement officielles dont l'ESP8266 et ESP32.



JavaScript Responsive Fully Open Source Crowdfunded

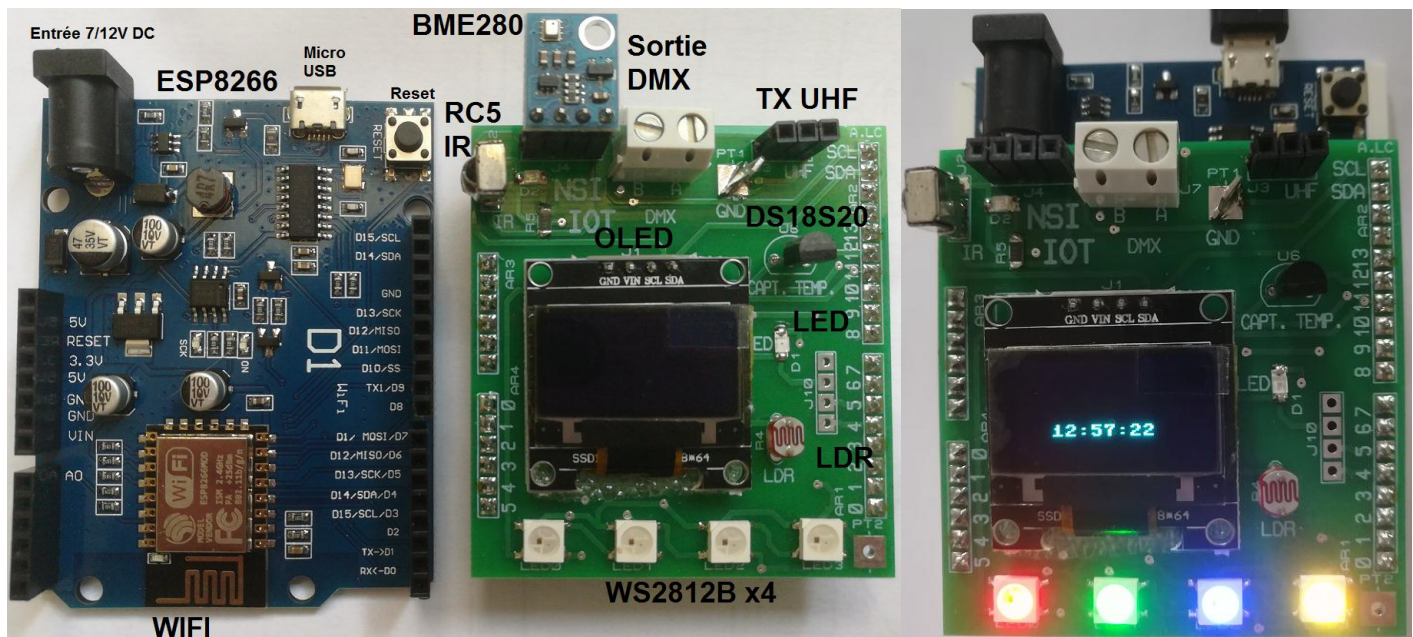
Espruino utilise le langage JavaScript. L'interpréteur JavaScript a l'avantage d'avoir un retour instantané afin de visualiser, déboguer et modifier le programme pendant son exécution.

<https://www.espruino.com/Other+Boards>

	Chip	Speed Mhz	Nb de Variables
<u>ESP8266</u>	Xtensa	80	1023
<u>ESP32</u>	Xtensa	240	5000

Il existe énormément d'exemples et de tutoriaux :

<https://www.espruino.com/Tutorials>



Description des périphériques montés sur la carte Shield

ESP32		ESP8266	Composant	Rôle
12	0	Logique (OUT)	Led rouge	Led de test
17	2	Logique (OUT)	Sortie RS485	Commande de spots DMX
21	4	Logique	SDA : Afficheur OLED	Afficheur OLED et capteur I2C divers
22	5	Logique	SCL : Afficheur OLED	
19	12	Logique	DS18S20	Capteur de température
23	13	Logique (IN)	VS1838B	Capteur infrarouge pour télécommande
18	14	Logique (OUT)	WS2812B	Ruban de 4 leds couleurs
5	15	Logique (OUT)	Emetteur UHF	Passerelle vers un réseau LPWAN
35	A0	Analogique (IN)	Capteur LDR	Capteur de lumière

DMX : ESP8266 : UART1 ou ESP32 : UART2

Configuration du shield en fonction de la carte microcontrôleur :

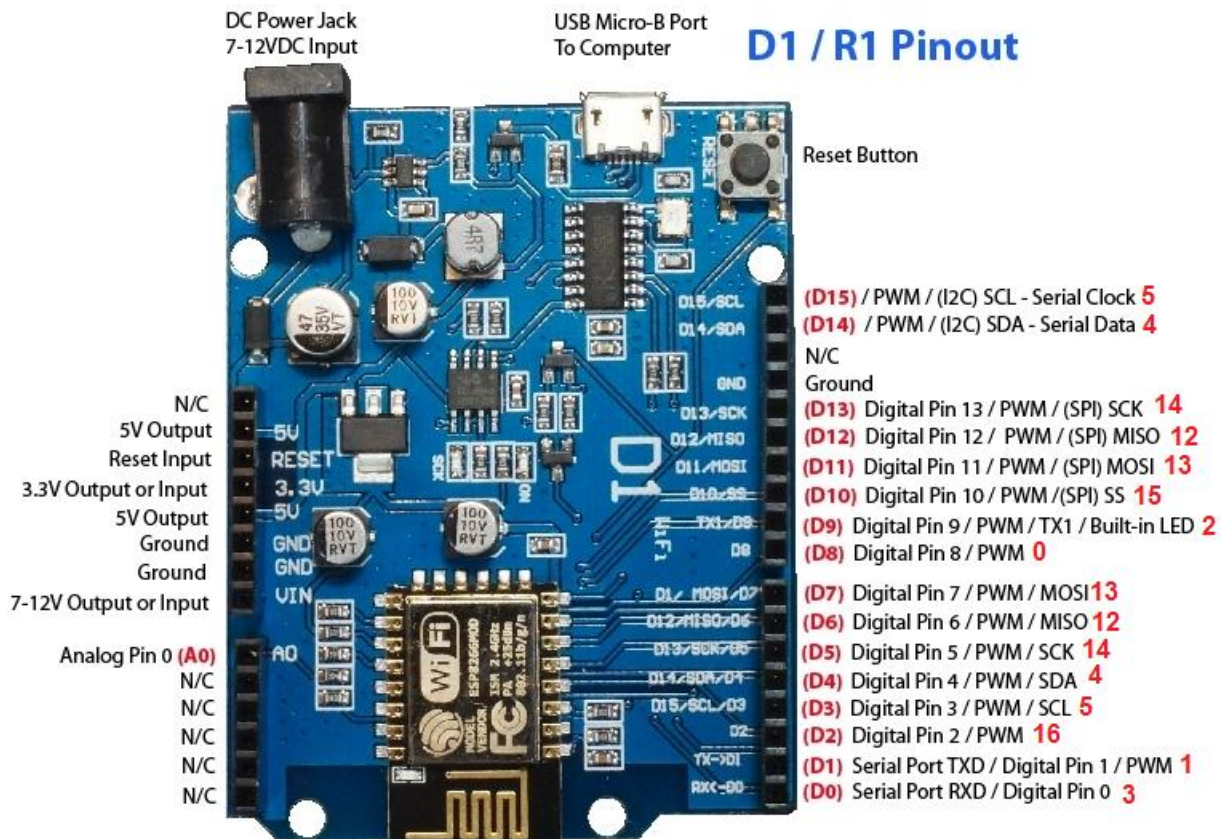
	J5	J6	J8	J9
ESP8266	Ouvert	Fermé	Ouvert	Fermé
ESP32	Fermé	Ouvert	Fermé	Ouvert

Commencer par télécharger le dossier complet sur GitHub

<https://github.com/f4goh/Carre-shield-IOT/archive/refs/heads/master.zip>

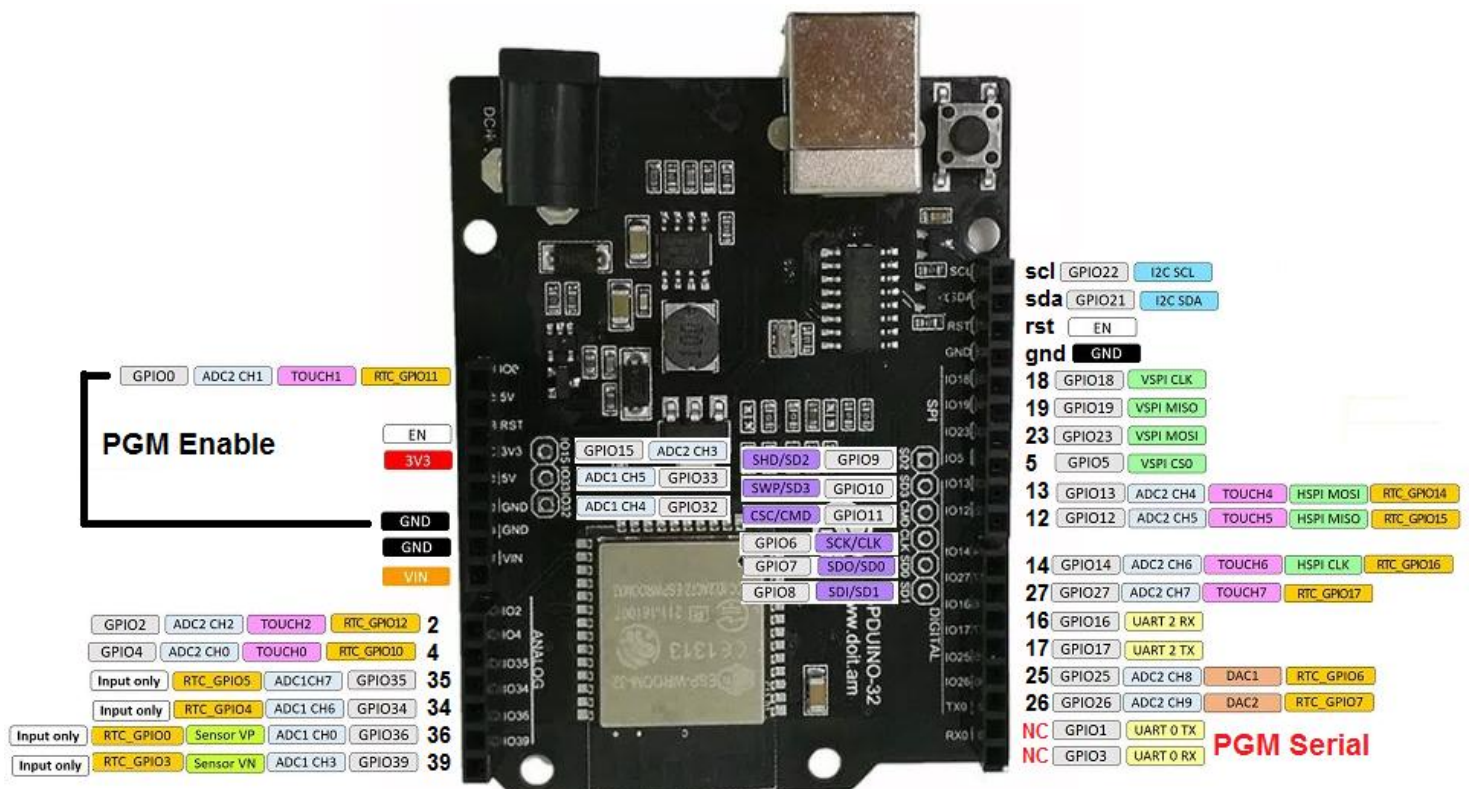
ESP8266

D1 / R1 Pinout



only pins 0, 2, 4, 5, 12, 13, 14, 15, and 16 can be used.

ESP32



Annexe 1 : Implantation du firmware Espruino dans l'ESP8266

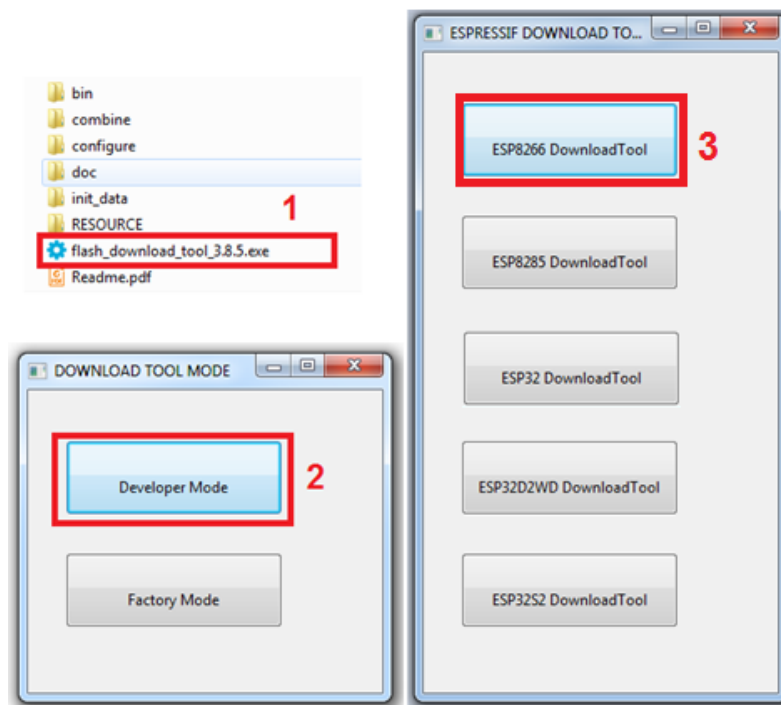
Programmation du firmware Micropython sous Windows :

Télécharger l'utilitaire de programmation sur le site officiel ou sur mon [github](#)

<https://www.espressif.com/en/support/download/other-tools>

Flash Download Tools					Expand all +	Download selected
<input type="checkbox"/>	Title	Platform	Version	Release Date	Download	
<input type="checkbox"/> +	Flash Download Tools	Windows PC	V3.9.2	2021.11.10		

Décompresser le fichier zip, puis exécuter l'utilitaire flash_download_tools



Télécharger les fichiers

https://github.com/f4goh/Carre-shield-IOT/tree/master/Espruino/firmware/espruino_2v11_esp8266%204M

master

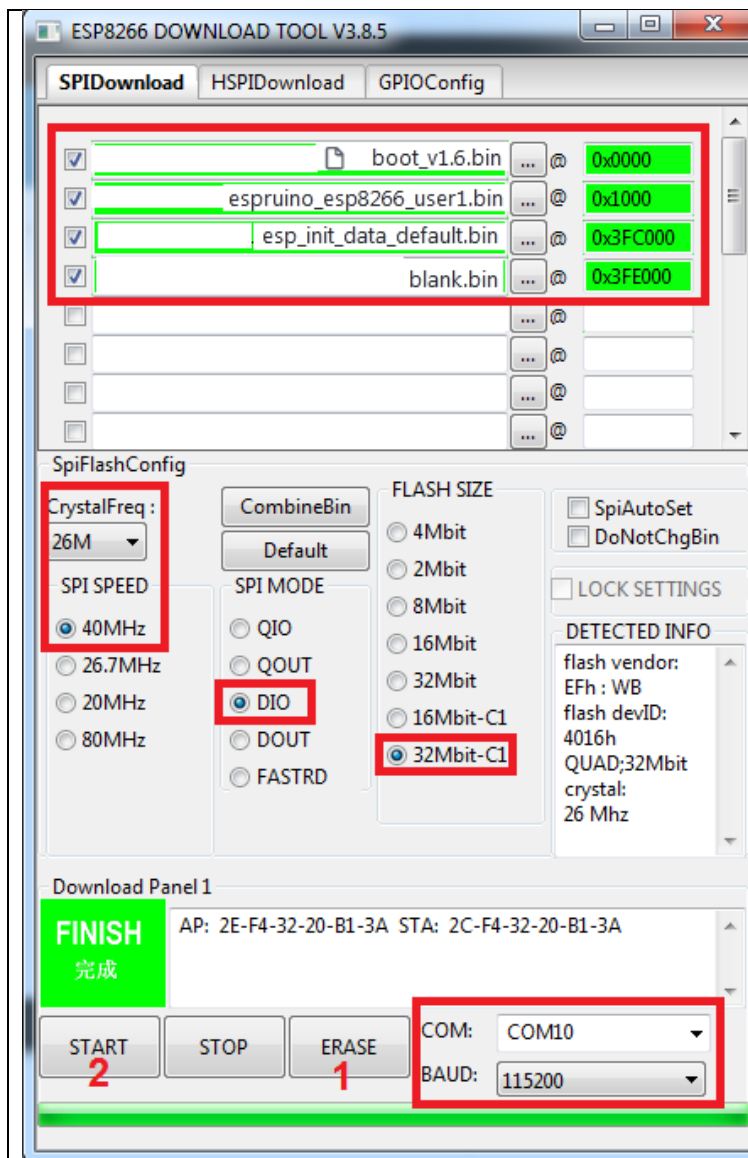
[Carte-shield-IOT](#) / [Espruino](#) / [firmware](#) / [espruino_2v11_esp8266 4M](#) /

f4goh update

..

	blank.bin	update
	boot_v1.6.bin	update
	esp_init_data_default.bin	update
	espruino_esp8266_user1.bin	update
	espruino_esp8266_user2.bin	update

Configurer flash_download_tools de la manière suivante :



1 Effectuer un effacement (erase).

Débrancher puis rebrancher la carte.

2 Programmer le firmware (Start).

Sous Linux il faut installer esptool

https://www.espruino.com/ESP8266_Flashing

```
$ /path/to/esptool/esptool.py --port /dev/ttyUSB0 --baud 115200 \
write_flash --flash_freq 80m --flash_mode qio --flash_size 32m \
0x0000 "boot_v1.6.bin" 0x1000 espruino_esp8266_user1.bin \
0x3FC000 esp_init_data_default.bin 0x3FE000 blank.bin
```

Remarque : Pour les utilisateurs de l'**esp01**, il faudra utiliser des fichiers spécifiques pour une mémoire flash de **512K**

https://github.com/f4goh/Carre-shield-IOT/tree/master/Espruino/firmware/espruino_2v11_esp8266%20512K

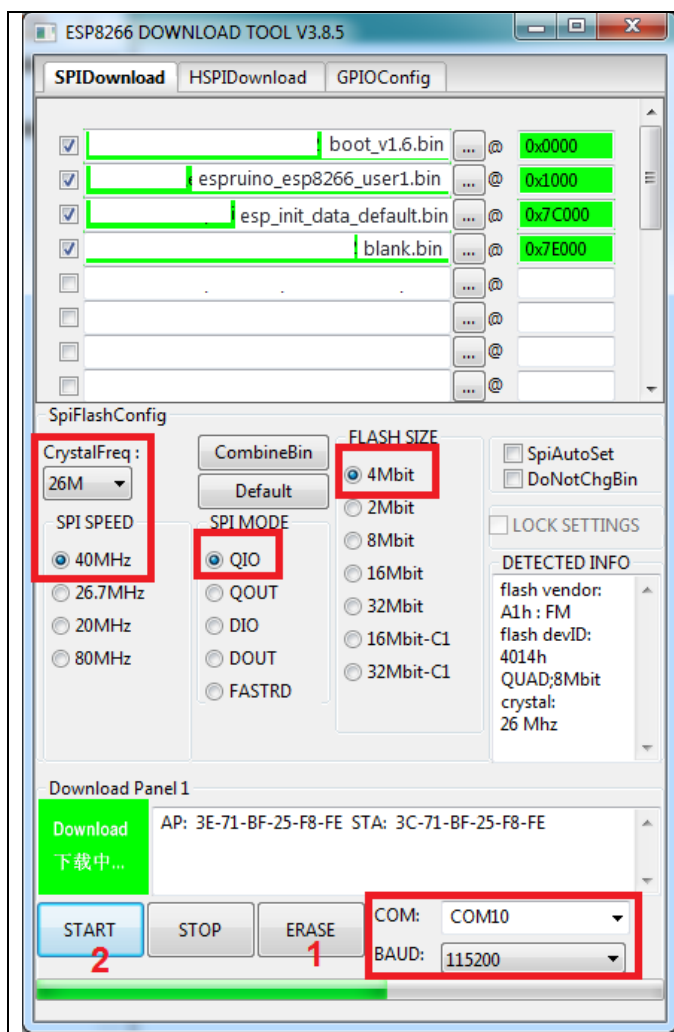
master Carte-shield-IOT / Espruino / firmware / espruino_2v11_esp8266 512K /

f4goh update

File	Action
blank.bin	update
boot_v1.6.bin	update
esp_init_data_default.bin	update
espruino_esp8266_user1.bin	update
espruino_esp8266_user2.bin	update



Configurer flash_download_tools de la manière suivante :



Positionner le commutateur du programmeur sur prog

1 Effectuer un effacement (erase).

Débrancher puis rebrancher la carte.

2 Programmer le firmware (Start).

Sous Linux il faut utiliser esptool.

https://www.espruino.com/ESP8266_Flashing

```
$ /path/to/esptool/esptool.py --port
/dev/ttyUSB0 --baud 115200 \
  write_flash --flash_freq 40m --flash_mode
qio --flash_size 4m \
  0x0000 "boot_v1.6.bin" 0x1000
espruino_esp8266_user1.bin \
  0x7C000 esp_init_data_default.bin 0x7E000
blank.bin
```

Annexe 2 : Implantation du firmware Micropython dans l'ESP32

Programmation du firmware Micropython sous Windows :

Télécharger l'utilitaire de [programmation expressif](https://www.espressif.com/en/support/download/other-tools)

<https://www.espressif.com/en/support/download/other-tools>

Flash Download Tools

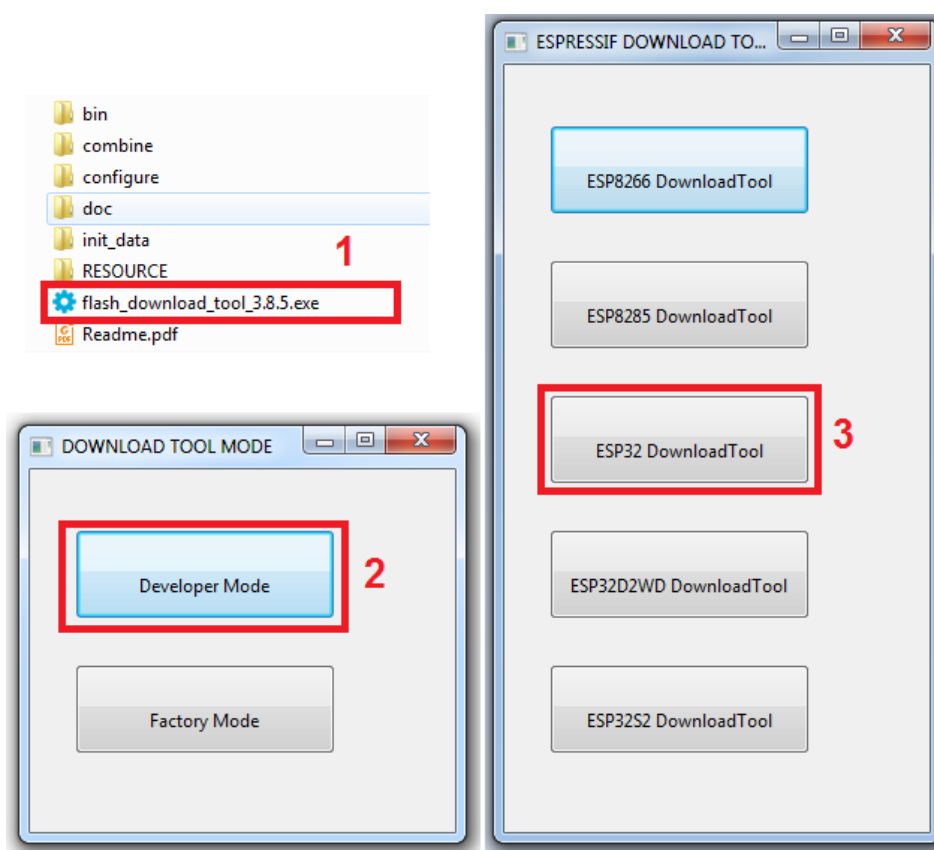
Expand all +



Download selected

<input type="checkbox"/>	Title	Platform	Version	Release Date ▾	Download
<input type="checkbox"/>	+ Flash Download Tools	Windows PC	V3.9.2	2021.11.10	

Décompresser le fichier zip, puis exécuter l'utilitaire flash_download_tools.

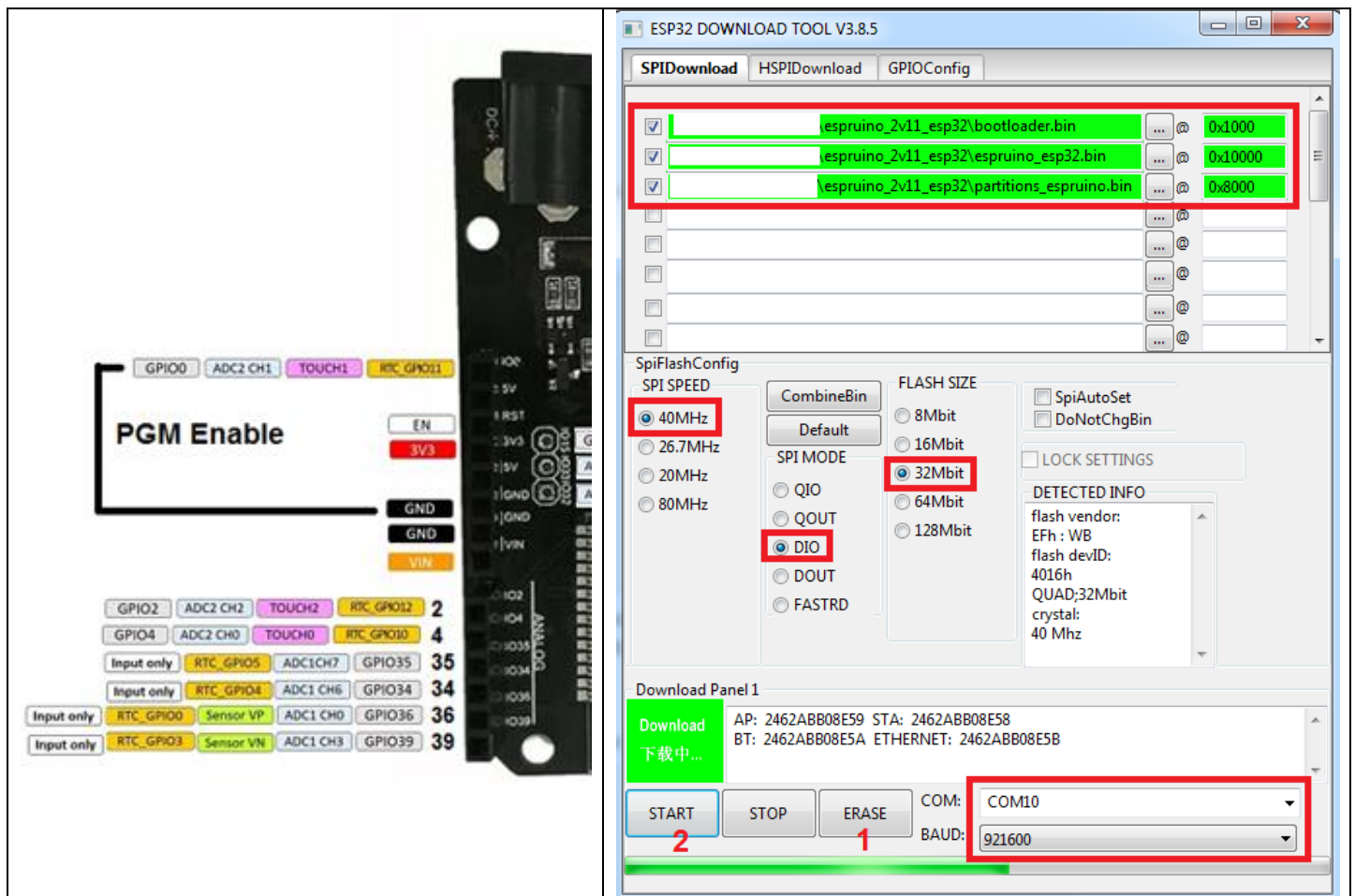


Télécharger le fichier firmware pour l'esp32.

https://github.com/f4goh/Carre-shield-IOT/tree/master/Espruino/firmware/espruino_2v11_esp32



Sélectionner les fichiers binaires, préciser les adresses de programmation, sélectionner le bon port de communication, relier un fil entre la masse GND et l'entrée GPIO 0 (PGM Enable) ensuite effacer (erase) et programmer le firmware : start.



Où trouver les derniers firmwares ?

<http://www.espruino.com/Download> ou <http://www.espruino.com/binaries/>

Annexe 3 : Programmation du firmware Espruino sous Linux

Pour programmer le firmware de l'esp32, il faudra installer esptool sous linux en ligne de commande

<https://github.com/espressif/esptool>

```
sudo apt install python-pip
sudo pip install --upgrade pip
sudo pip install esptool
pip install pyserial
sudo pip install pyserial
```

Repérer le port série de la carte esp32

```
nsi@nsi-LIFEB00K-A555:~$ ls /dev/tty*
/dev/tty19  /dev/tty34  /dev/tty5   /dev/tty8   /dev/ttyS21  /dev/ttyS9
/dev/tty2   /dev/tty35  /dev/tty50  /dev/tty9   /dev/ttyS22  /dev/ttyUSB0
```

Exécuter les 2 commandes suivantes :

Esp8266 4M (ESP12)

```
$ python esptool.py --port /dev/ttyUSB0 erase_flash
$ /path/to/esptool/esptool.py --port /dev/ttyUSB0 --baud 115200 \
  write_flash --flash_freq 80m --flash_mode qio --flash_size 32m \
    0x0000 "boot_v1.6.bin" 0x1000 espruino_esp8266_user1.bin \
    0x3FC000 esp_init_data_default.bin 0x3FE000 blank.bin
```

Esp8266 512K (ESP01)

```
$ python esptool.py --port /dev/ttyUSB0 erase_flash
$ /path/to/esptool/esptool.py --port /dev/ttyUSB0 --baud 115200 \
  write_flash --flash_freq 40m --flash_mode qio --flash_size 4m \
    0x0000 "boot_v1.6.bin" 0x1000 espruino_esp8266_user1.bin \
    0x7C000 esp_init_data_default.bin 0x7E000 blank.bin
```

Esp32 <https://www.espruino.com/ESP32>

```
esp-idf/components/esptool_py/esptool/esptool.py \
  --chip esp32 \
  --port /dev/ttyUSB0 \
  --baud 921600 \
  --after hard_reset write_flash \
  -z \
  --flash_mode dio \
  --flash_freq 40m \
  --flash_size detect \
  0x1000 bootloader.bin \
  0x8000 partitions_espruino.bin \
  0x10000 espruino_esp32.bin
```

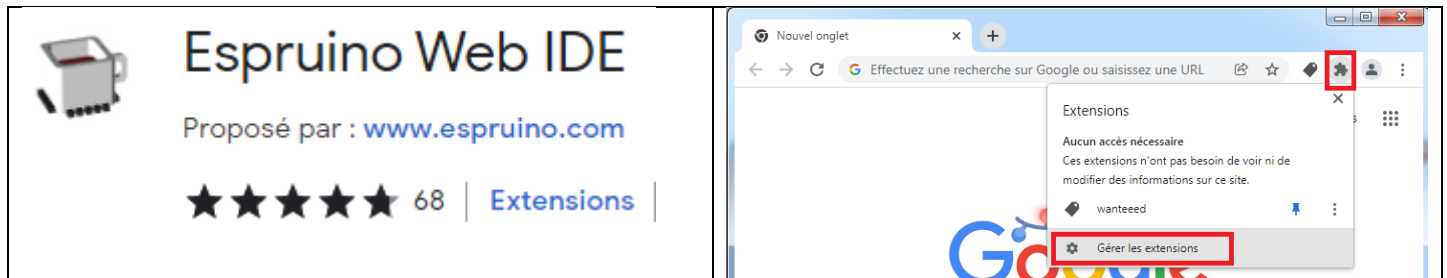
Commandes afin de changer l'adresse MAC de l'ESP32 si nécessaire

```
python espfuse.py --port /dev/ttyUSB0 summary
python espfuse.py --port /dev/ttyUSB0 dump
python espfuse.py --port /dev/ttyUSB0 mac
python espfuse.py --port /dev/ttyUSB0 get_custom_mac
python espfuse.py --port /dev/ttyUSB0 burn_custom_mac de:ad:be:ef:fe:00
python espfuse.py --port /dev/ttyUSB0 get_custom_mac
```

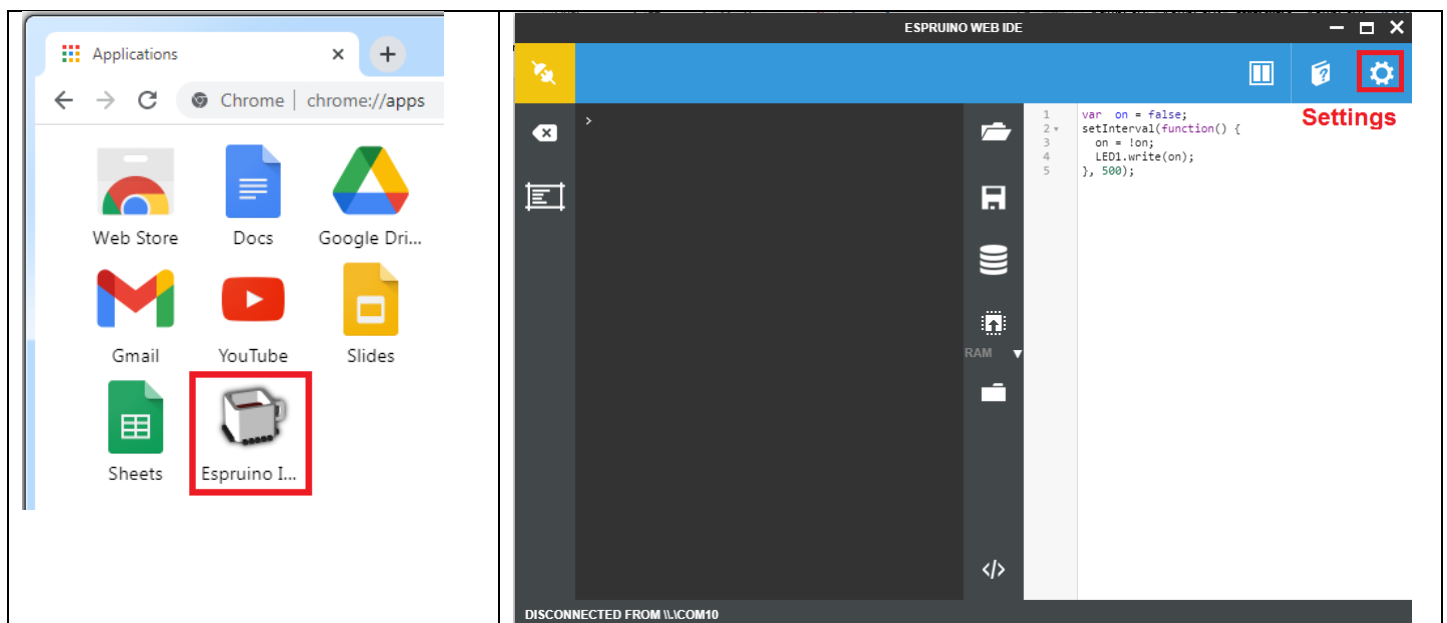
Annexe 4 : Installation de l'utilitaire de programmation Espruino WEB IDE

<https://chrome.google.com/webstore/detail/espruino-web-ide/bleoifhkdbjfbobjackfdifdneehpo>

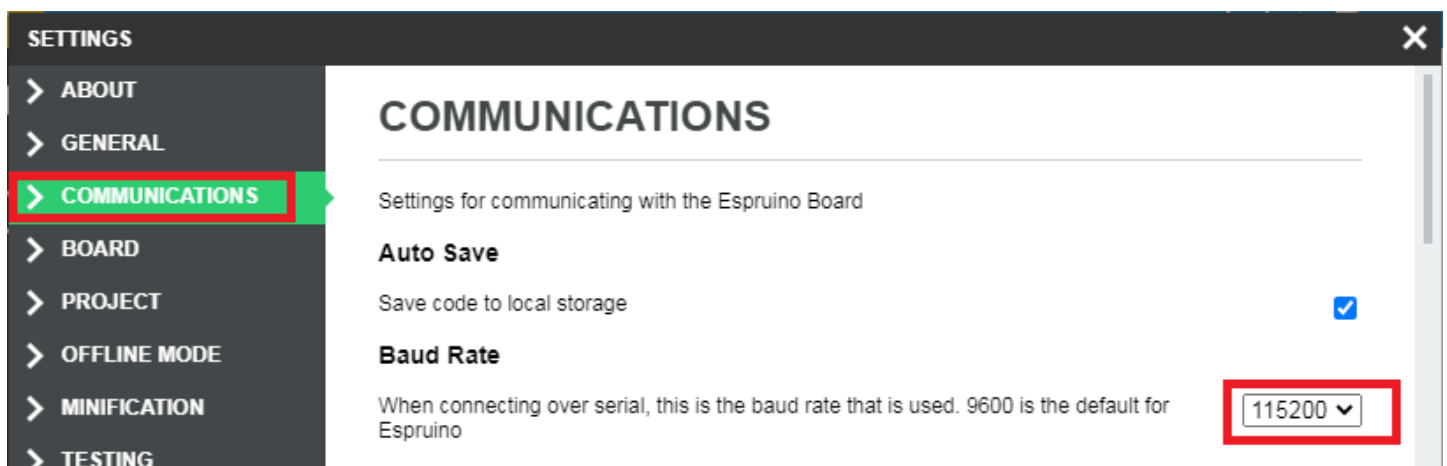
Utilitaire est un module d'extension du navigateur Chrome.



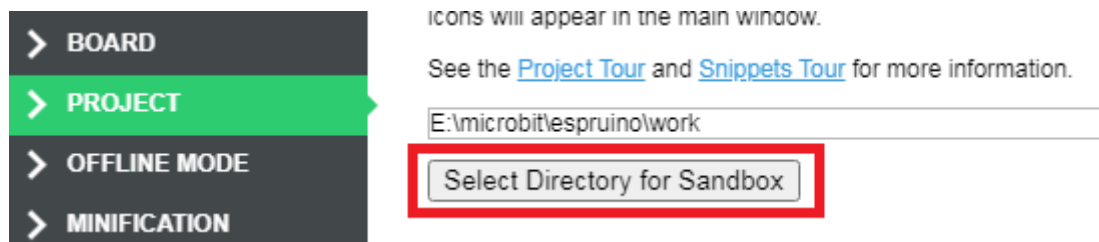
Vérifier l'installation du logiciel, puis l'exécuter.



Aller dans le menu settings, puis modifier la vitesse de communication à 115200.



Sélectionner un répertoire de travail :

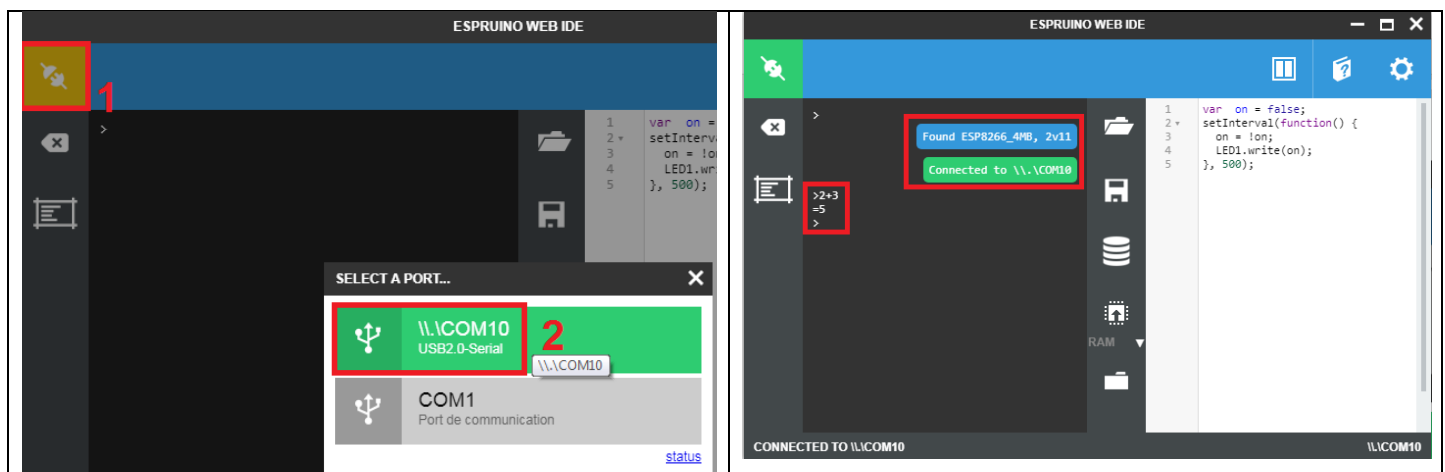


Les répertoires suivants seront créés automatiquement

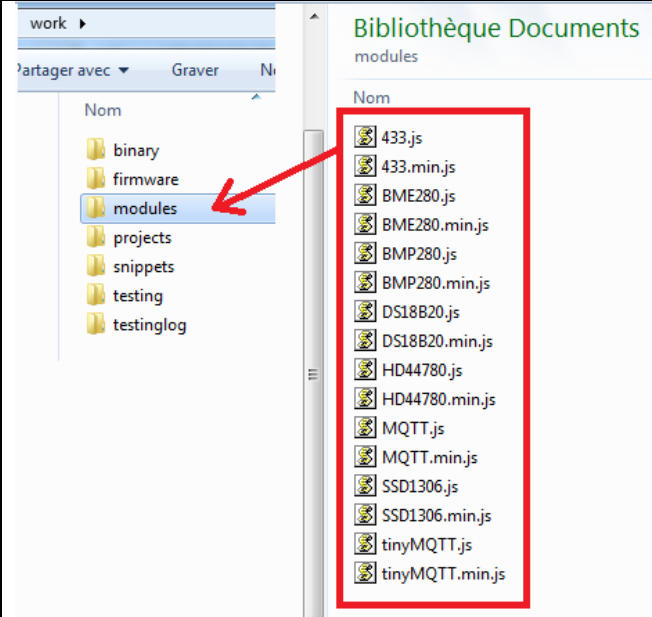
	<p>Le répertoire projects contient les programmes en javascript Exemple :</p> <ul style="list-style-type: none"> blink_oninit.js blink_oninit_V2.js ds18s20_8266.js io file.js lcd.js lcd_8266.js neopixels_8266.js scan_i2c.js scan_i2c_V2.js ssd1306_8266.js wifi_8266.js wifi_8266_oninit.js 	<p>Le répertoire module contient les bibliothèques Exemple :</p> <ul style="list-style-type: none"> DS18B20.js DS18B20.min.js HD44780.js HD44780.min.js MQTT.js MQTT.min.js SSD1306.js SSD1306.min.js tinyMQTT.js tinyMQTT.min.js
--	---	---

Tester la connexion avec la carte Esp8266.
Vérifier que la carte est bien reconnue.

Saisir dans la console un calcul simple pour vérifier que l'ESP8266 répond correctement.



Installation des modules

	<p>Copier le contenu du répertoire module de Github dans le répertoire module de votre espace de travail créé précédemment. (Select Directroy for Sandbox)</p> <p>Davantage de modules sont téléchargeables ici :</p> <p>http://www.espruino.com/modules/</p>
--	--

Pour la documentation des modules, utiliser l'URL <http://www.espruino.com/> en ajoutant le nom du module.

Exemples :

<https://www.espruino.com/SSD1306>

<https://www.espruino.com/DS18B20>

<https://www.espruino.com/MQTT>

Installation de Espruino WEB IDE sous linux

1 Commencer par installer google chrome

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo dpkg -i google-chrome-stable_current_amd64.deb
sudo apt-get install -f
```

2 Puis installer Espruino WEB IDE comme expliqué précédemment sous Windows.

L'icône Espruino WEB IDE doit apparaitre dans le menu de démarrage.

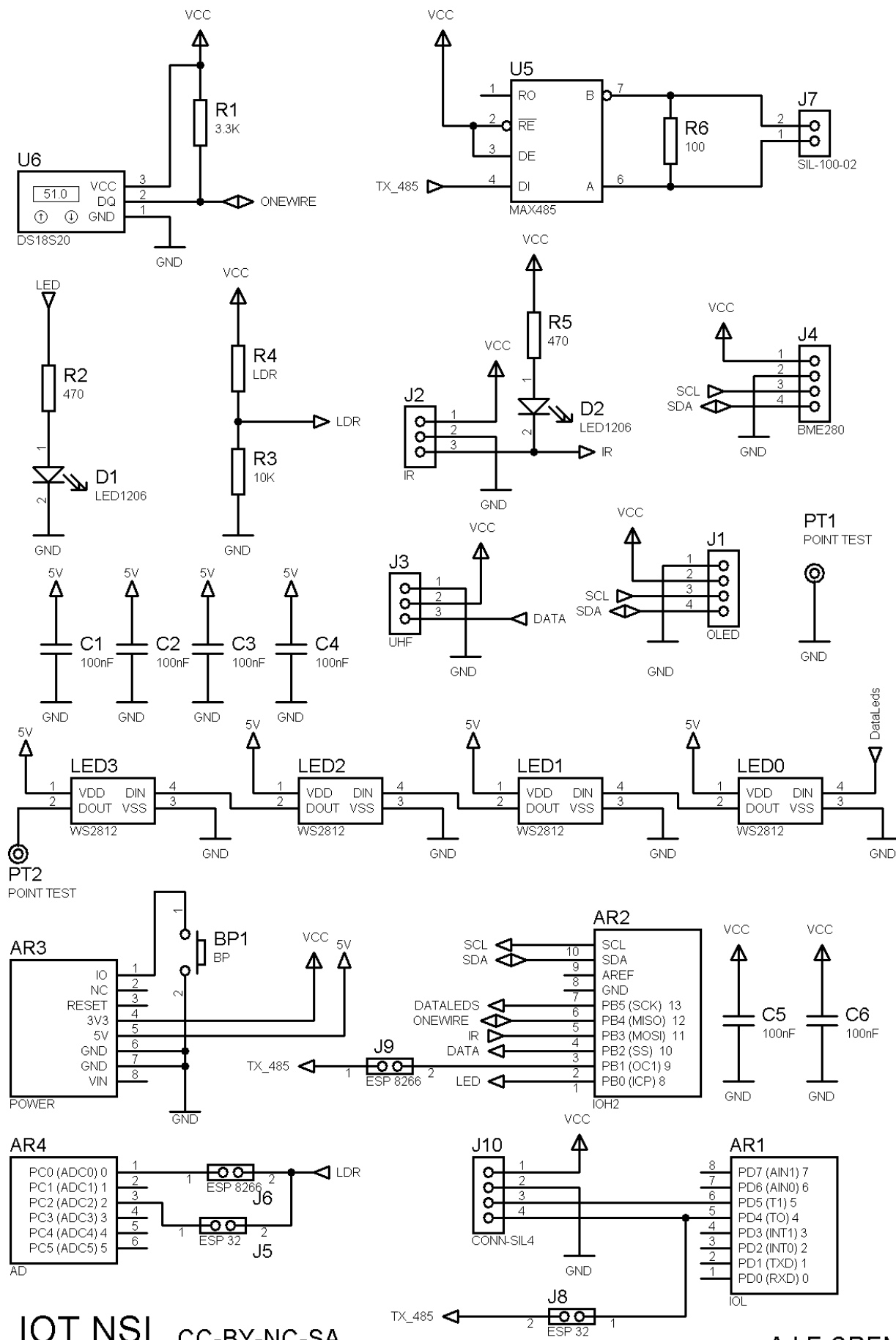
3 En cas de problème de droits sur port /dev/ttyUSB0, ajouter l'utilisateur au groupe dialout.

```
sudo adduser nom_de_la_session_utilisateur dialout
```

Voir ensuite le document prise en main de l'environnement de programmation

IOT-TP1 prise en main.pdf

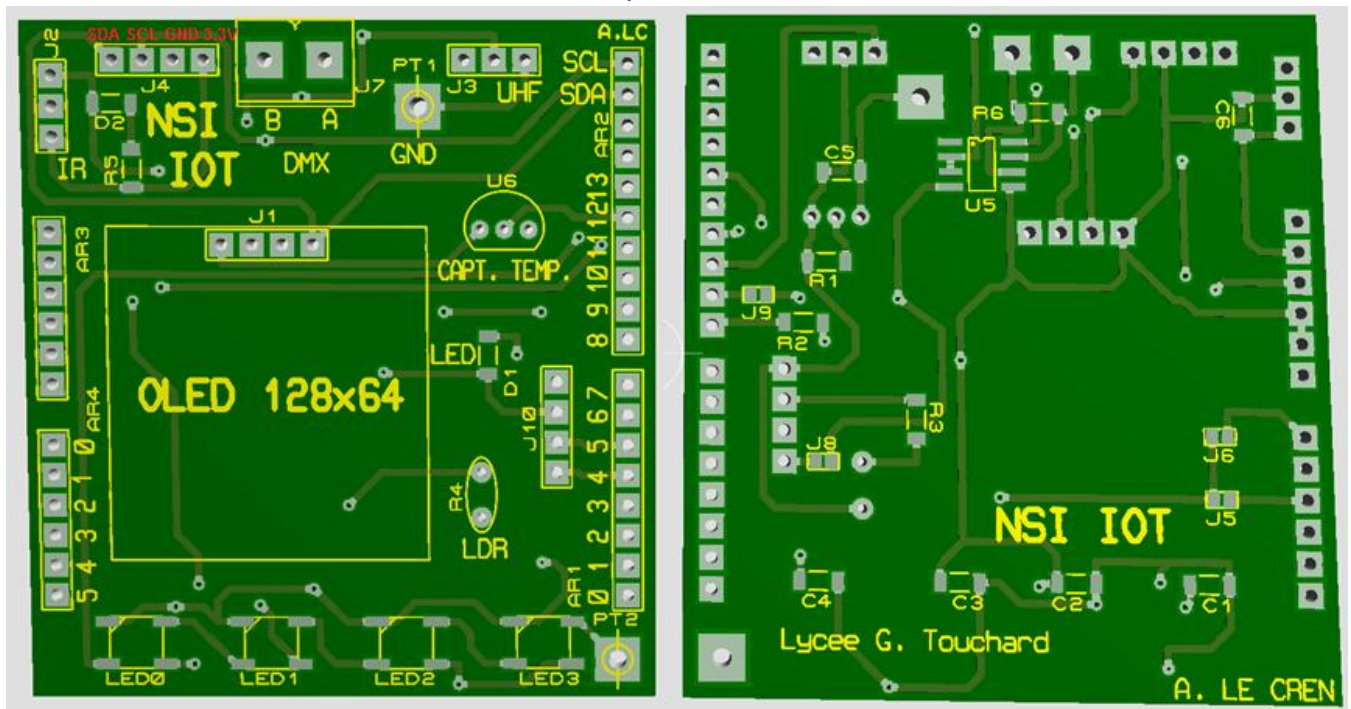
Annexe 3 : Schéma structurel



IOT NSI CC-BY-NC-SA

A.LE CREN

Circuit imprimé shield IOT



Nomenclature

Résistances

- | | | |
|-----------|---------|--|
| 1 R1,3.3K | 2 R2,R5 | 470 cms1206 |
| 1 R3 | 10K | cms1206 |
| 1 R4 | LDR | ldr classique trou traversant petit modèle |
| 1 R6 | 100 | cms1206 |

Condensateurs

- | | | |
|---------|-------|---------|
| 6 C1-C6 | 100nF | cms1206 |
|---------|-------|---------|

Circuits intégrés

<https://www.ebay.fr/itm/100Pcs-MAX485-MAX485CSA-Txrx-RS485-RS422-Lowpwr-SOP-8-Date-CODE-12-wv/262963276497>

- | | | |
|------|--------|-------|
| 1 U5 | MAX485 | sop-8 |
|------|--------|-------|

<https://www.ebay.fr/itm/10PCS-IC-DS18S20-DS1820-Digital-Thermometer-IC-GOOD-QUALITY-Li2/132256267252>

- | | | |
|------|---------|-------|
| 1 U6 | DS18S20 | to-92 |
|------|---------|-------|

Diodes : 2 D1,D2 LED cms1206

Divers :

- | | | |
|-------------|------------|------------------------------------|
| 1 AR1 | IOL | barette male sécable 8pts |
| 1 AR2 | IOH2 | barette male sécable 10pts |
| 1 AR3 | POWER | barette male sécable 6pts |
| 1 AR4 | AD | barette male sécable 6pts |
| 1 J1 | OLED | oled ssd1306 128x64 I2C |
| 1 J2 | IR | VS1838B recepteur IR |
| 1 J3 | | barette femelle sécable 3pts |
| 1 J4 | | barette femelle sécable 4pts |
| 1 J7 | SIL-100-02 | bornier gris 2pts |
| 1 J10 | | barette femelle sécable 4pts |
| 4 LED0-LED3 | WS2812b | led RGB boîtier blanc cms a souder |

<https://www.ebay.fr/itm/0-96-I2C-IIC-SPI-Serial-128X64-OLED-LCD-LED-Display-Module-for-Arduino-SSD1306/223119333626>

<https://www.ebay.fr/itm/WS2812B-5050-SMD-Addressable-Digital-RGB-LED-4-pin-Chip-5V-Black-or-White/183460578312>

<https://www.ebay.fr/itm/20PCS-VS1838-TL1838-VS1838B-Universal-Infrared-Receiving-Head-For-Remote-control/201249361916>

Ajouter une carte **esp8266** ou **esp32**

<https://www.ebay.fr/itm/OTA-WeMos-D1-CH340-WiFi-Development-Board-ESP8266-ESP-12E-For-Arduino-IDE-UNO-R3/163429353623>

<https://www.ebay.fr/itm/ESP32-UNO-R3-D1-R32-WIFI-Bluetooth-USB-B-CH340-Development-Board-For-Arduino/264083453537>

Sans oublier le cordon micro-usb