

# Introduction à Arduino et l'internet des objets

Trakk Namur - Octobre-Novembre 2019

# Disclaimer

Station Météo = support pédagogique

capteurs relativement bon marché = résultats potentiellement imprécis par rapport à des solutions industrielles

# Présentation du Workshop

- Séance 1 : Introduction et première approche
- Séance 2 : Bases de programmation; capteurs et actionneurs. Présentation et analyse du projet.
- Séance 3 : Assemblage et mise en route du projet
- Séance 4 : IoT : communication : récupérer des données depuis Internet : Le protocole HTTP
- Séance 5 : IoT : communication, publier des données : le protocole MQTT

# Me, myself, I

- Web developer
- Co-fondateur du studio digital 3kd.be
- Membre du hackerspace de Liège
- Enseignant

# Smart Cities

## Enjeux

- Data (et open data)
- Bien être citoyen
- Optimisation des coûts
- Motiver des décisions et actions

# Smart Cities

## Acteurs

- Pouvoirs publics
- Entreprises privées
- Initiatives citoyennes

# Smart Cities

## Critique

- Tensions entre acteurs publics et privés
- Manque de vision dans la gouvernance
- Trop souvent approche “top to bottom”

# Smart Cities

## Exemple

- Namur :
  - <https://data.namur.be>
  - <https://www.namur.be/fr/ma-ville/smart-city/namur-ville-intelligente/la-position-de-namur-en-tant-que-smart-city>

# Smart Cities

## Références

- Rapport Audacities : <https://bit.ly/2HWvRgx>
- HEC-ULiège Smart City Institute :
  - <http://labos.ulg.ac.be/smart-city/>
  - Baromètre Smart Cities Wallonie 2018: <https://bit.ly/2zYxNI1>
  - Guide Pratique : <http://guidesmartcity.be/>

# L'internet des objets

Extension d'internet à des choses et des lieux du monde physique

Prise de décisions basée sur les données captées

# L'internet des objets

- Emergence grâce à une corrélation de plusieurs facteurs
  - Faible coût des capteurs et processeurs
  - Faible coût de la bande passante
  - Augmentation de la capacité de calcul (big data)
  - Infrastructures télécom => connectivité omniprésente
  - Augmentation du nombre de smartphones

# Arduino et les microcontrôleurs

- Microcontrôleur :
  - Circuit intégré reprenant les caractéristiques d'un ordinateur (mémoire et processeur)
  - Souvent programmé pour n'exécuter qu'une seule tâche
  - Performances limitées par rapport à un ordinateur mais son coût l'est également

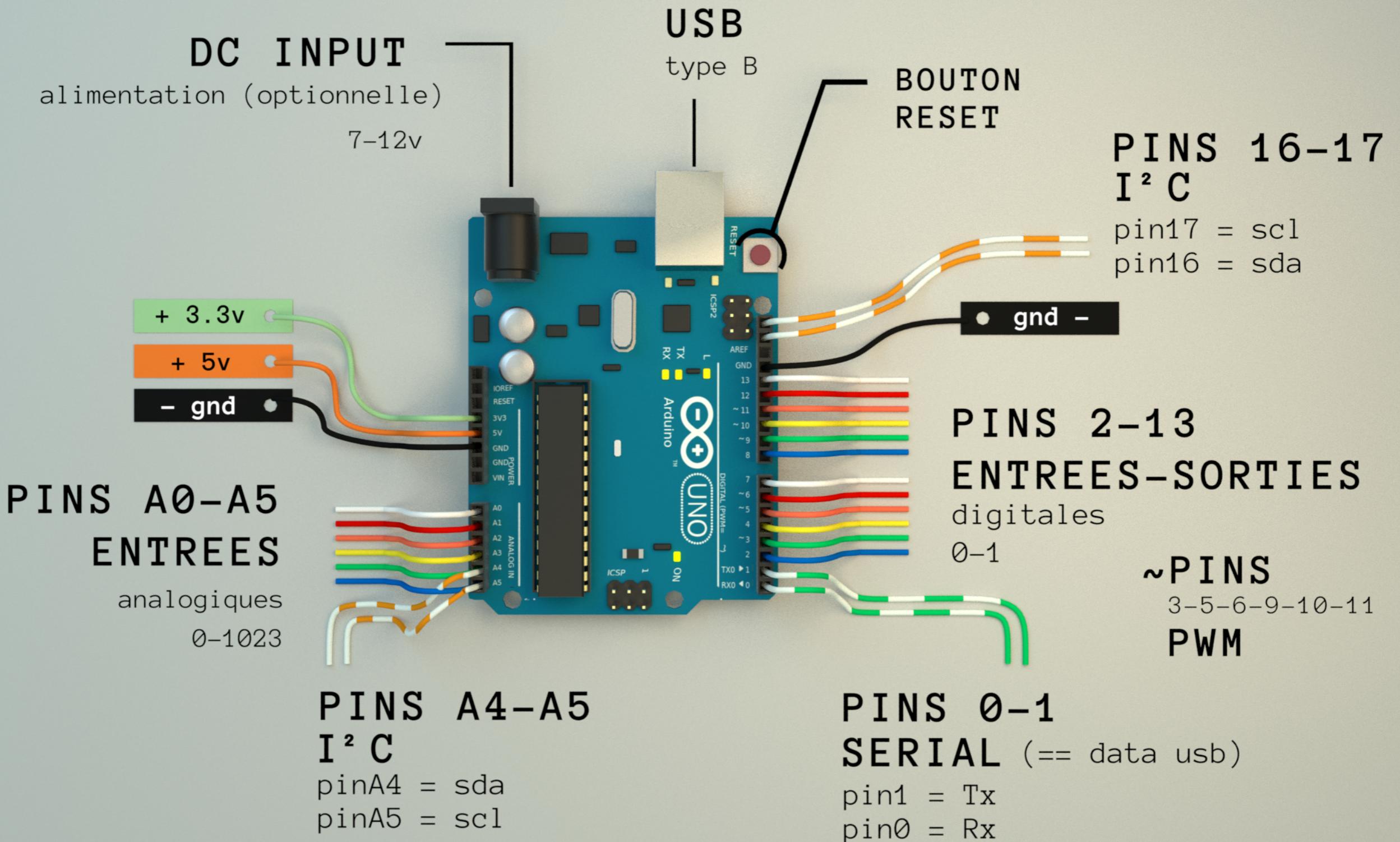
# Présentation d'Arduino par l'un de ses créateurs : Massimo Banzi

Ted Talk : How Arduino is open sourcing imagination



# Arduino UNO

- Carte la plus répandue : précurseur
- ATMega 328p (16Mhz - 8 bits)
- 14 Entrées et sorties digitales
- 6 Entrées analogiques (ADC 10 bits)
- Tension de fonctionnement **5v**

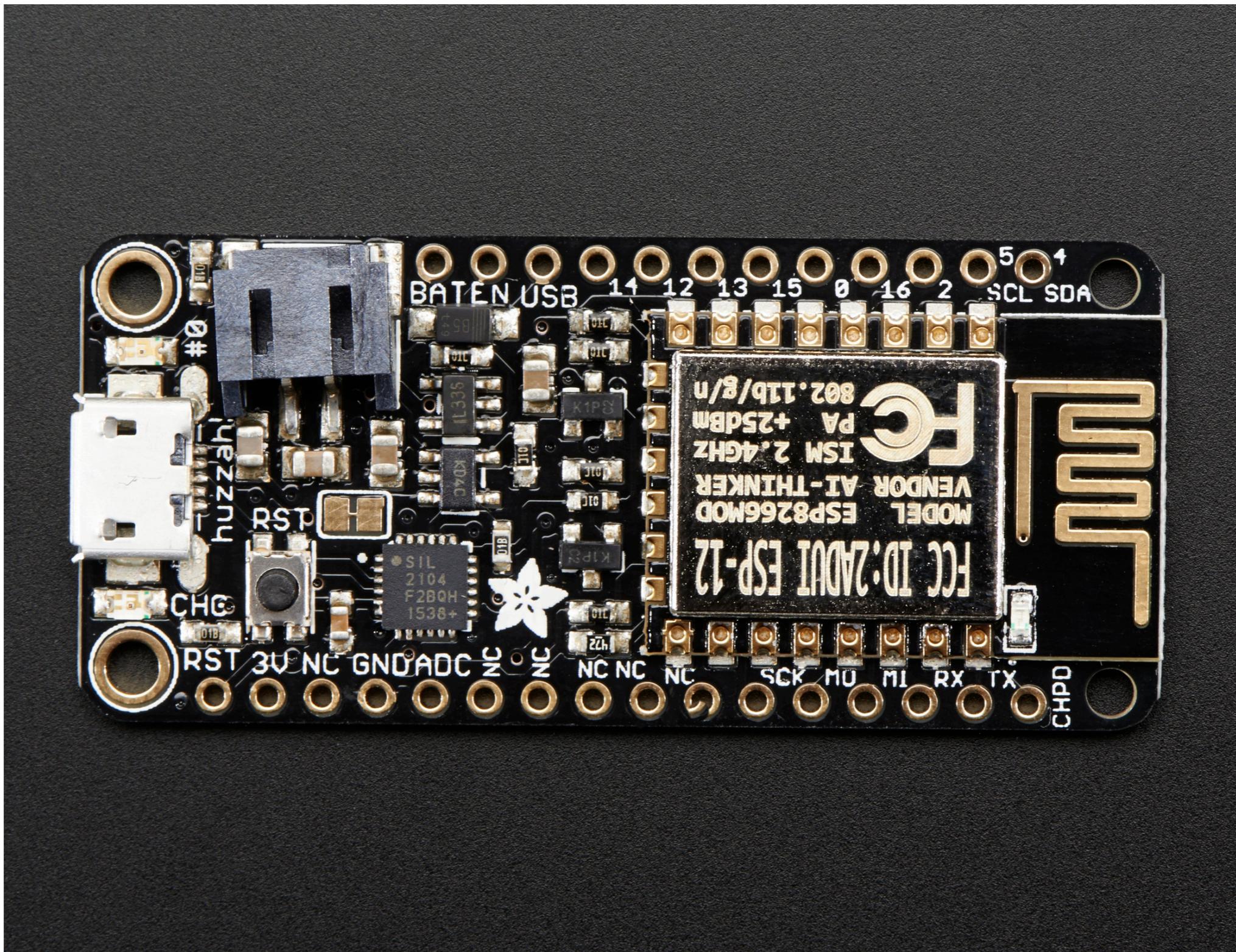


# Adafruit Feather Huzzah ESP8266

- Espressif ESP8266
  - Tensilica 80Mhz
  - WiFi 2.4Ghz
  - 4MB Flash
- 9 entrées/sorties (GPIO)
- Tension de fonctionnement : **3.3v**

# Matériel

- Feather Huzzah EP8266 :
  - [\*\*https://learn.adafruit.com/adafruit-feather-huzzah-esp8266\*\*](https://learn.adafruit.com/adafruit-feather-huzzah-esp8266)
- 2.4" TFT FeatherWing
  - [\*\*https://learn.adafruit.com/adafruit-2-4-tft-touch-screen-featherwing/\*\*](https://learn.adafruit.com/adafruit-2-4-tft-touch-screen-featherwing/)
- DHT22 : Temperature and Humidity sensor
  - [\*\*https://learn.adafruit.com/dht/overview\*\*](https://learn.adafruit.com/dht/overview)
- push button
- Leds
- potentiomètre
- phototransistor
- résistances, jumper wire
- breadboard



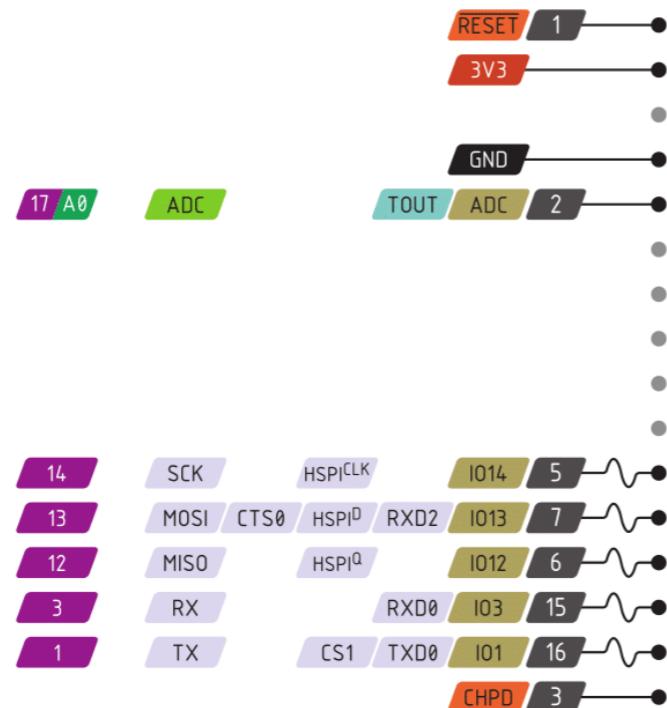
<https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/pinouts>

# feather HUZZAH ESP8266

PINOUT

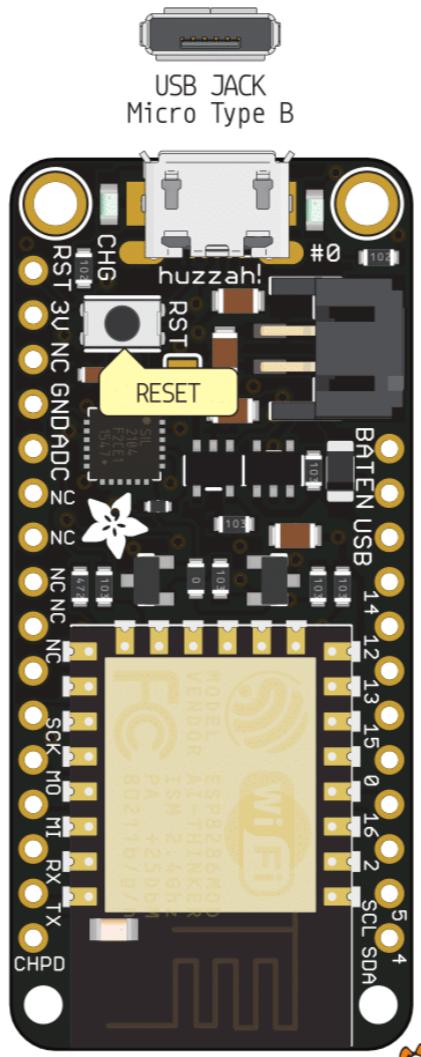


PWM Pin

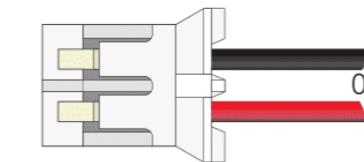


**Absolute MAX per pin**  
12mA, 6mA recommended

**Absolute MAX** 85mA for  
the entire package



USB JACK  
Micro Type B



Optional Lipoly Battery

VBAT	En	VBUS	i	Connect to ground to disable the 3.3V regulator	
5	I014	HSPI <sup>CLK</sup>	SCK	14	
6	I012	HSPI <sup>D</sup>	MISO	12	
7	I013	RXD2	HSPI <sup>Q</sup>	MOSI	13
10	I015	TXD2	HSPI <sup>CS</sup>	RTS0	15
12	I00	CS2		0	
4	I016	WAKE		16	
11	I02	TXD1		2	
14	I05		SCL	5	
13	I04		SDA	4	

**VBUS** Connected to 5V USB Port  
**Absolute MAX** 500mA

**VBAT** It's the positive voltage  
from to JST Batt jack

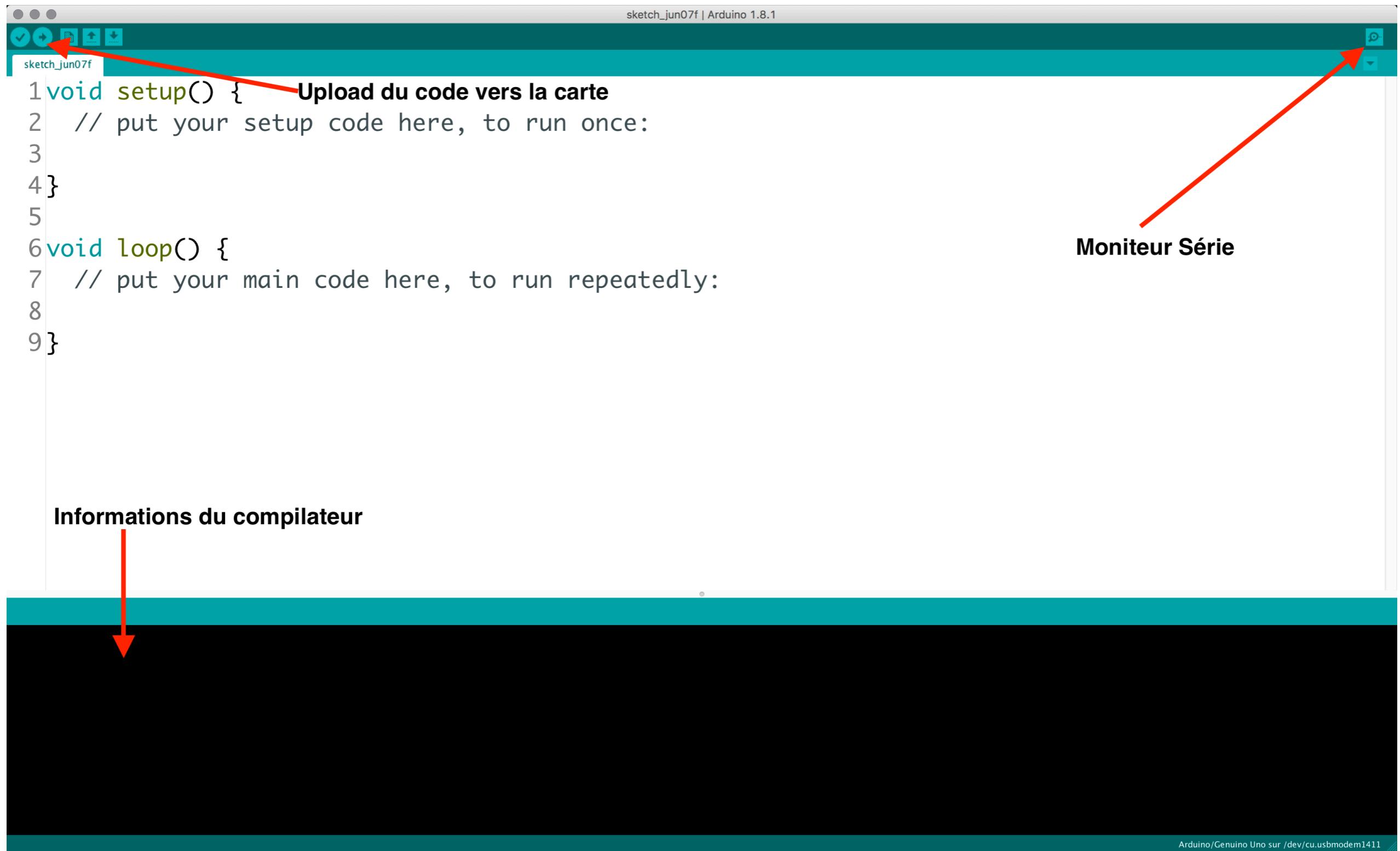
**3V3** 3V3 output from regulator  
**Absolute MAX** 400mA

# Installation et configuration des logiciels

<https://arduino.cc/downloads>

- **IDE** : Integrated Development Environment, logiciel installé sur la machine contenant tous les outils pour compiler le code et flasher le microcontrôleur

# Présentation de l'IDE Arduino



# Configuration de l'IDE pour ESP 8266

Télécharger et installer les  
drivers de la carte

**<https://frama.link/drivers-usb-esp8266>**

# Configuration de l'IDE pour ESP 8266

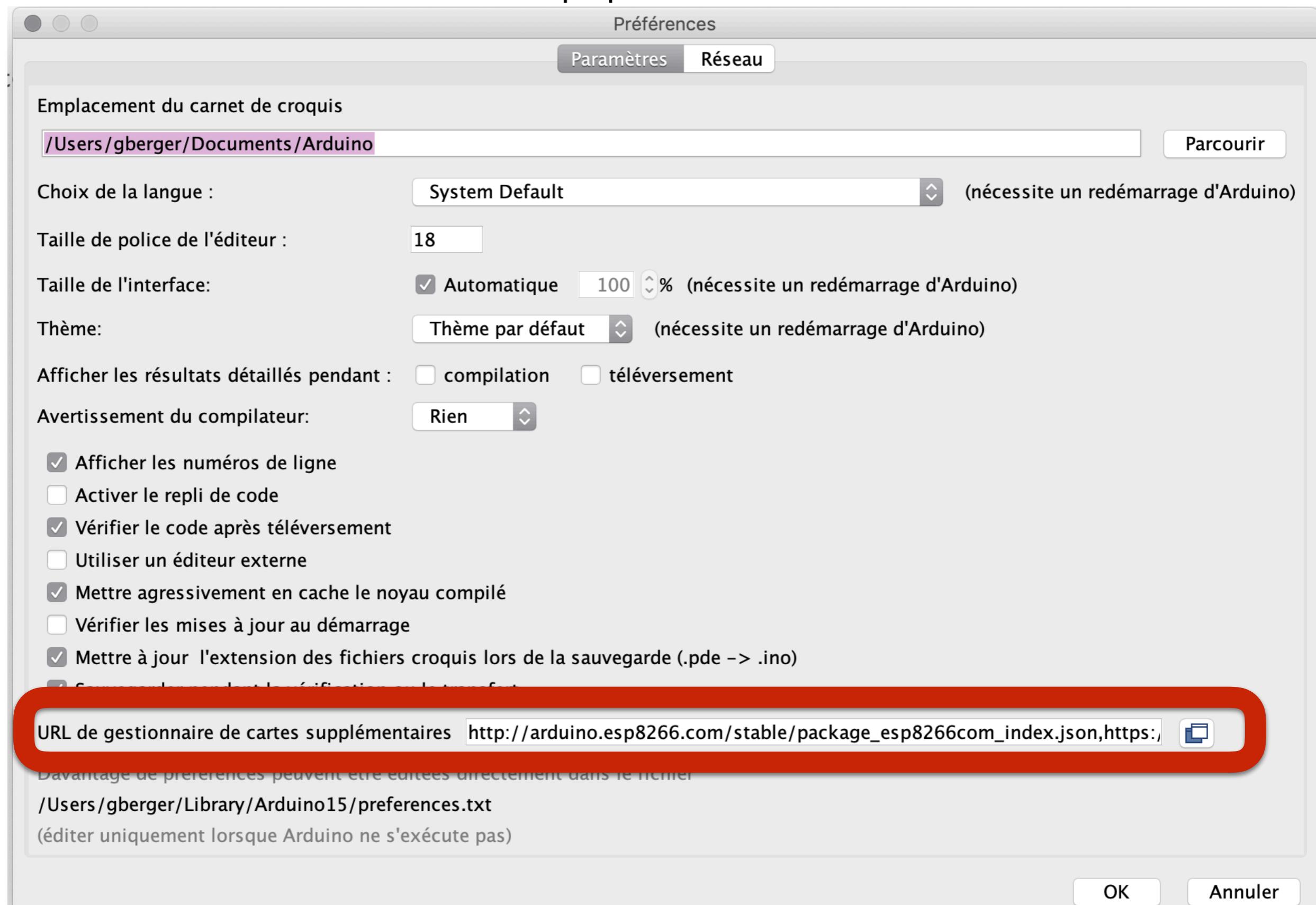
Installer les outils de développement pour ESP8266

Copier le lien vers le fichier de configuration :

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

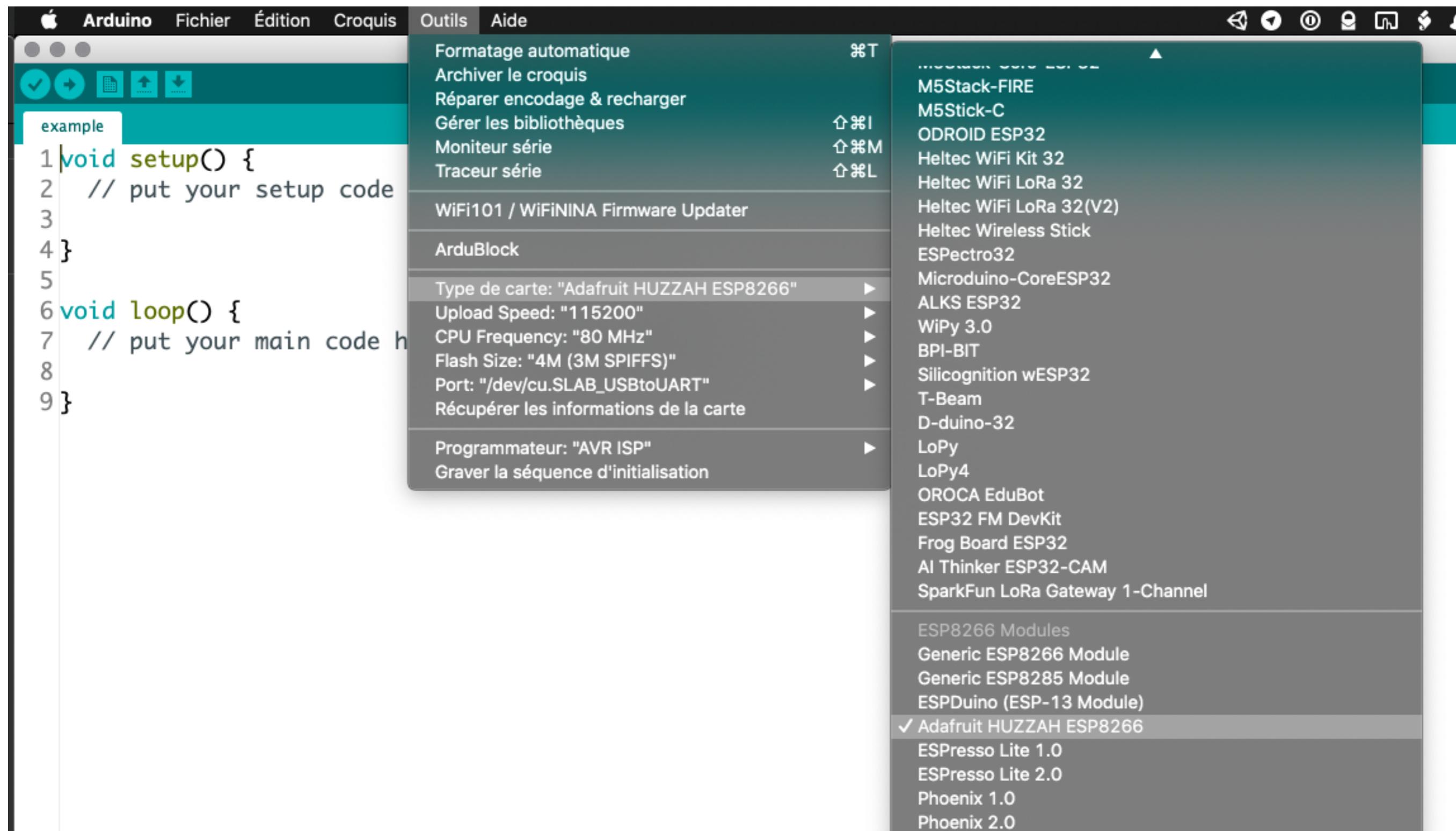
# Configuration de l'IDE pour ESP 8266

Coller le lien dans le champ prévu



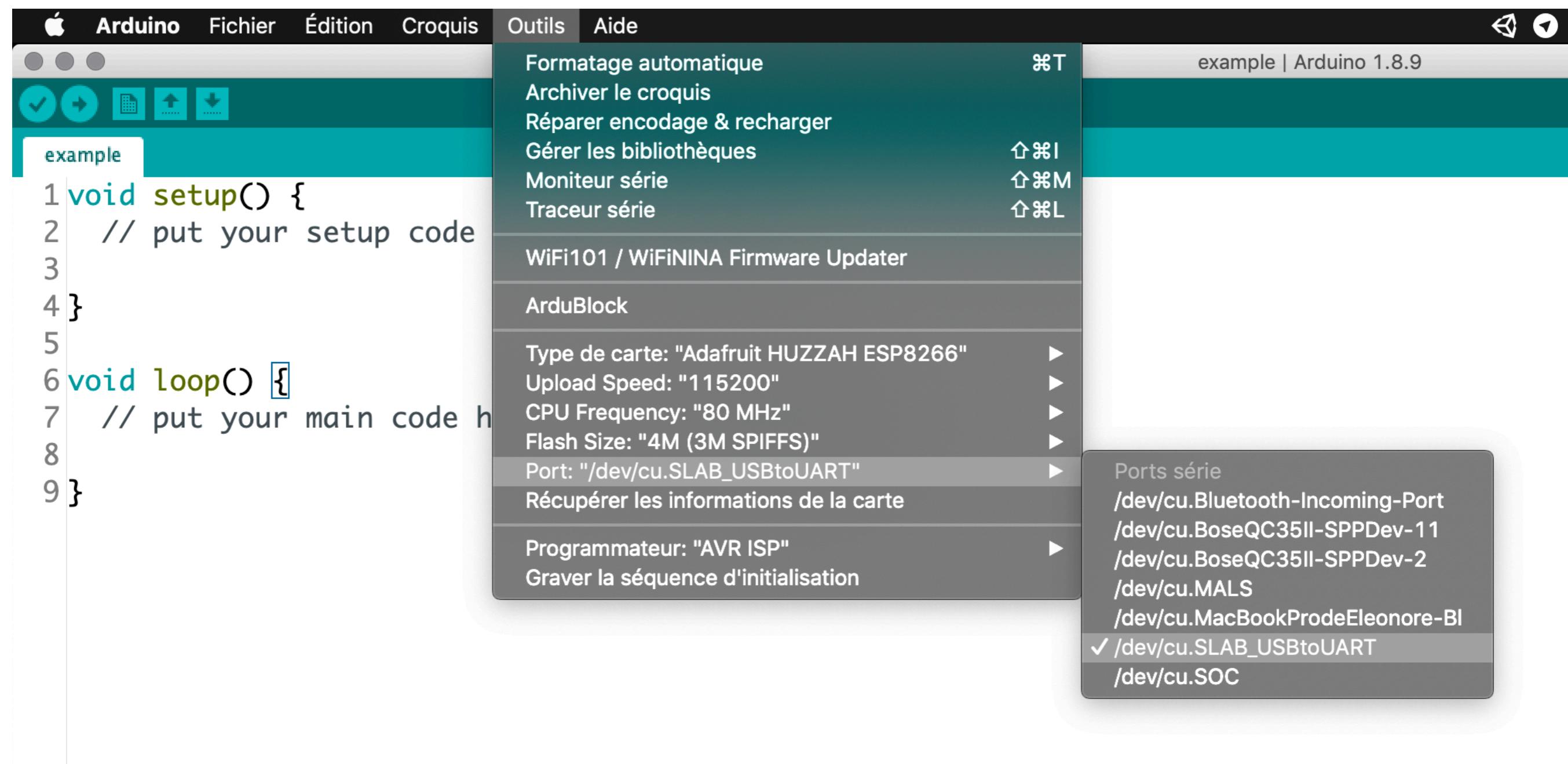
# Configuration de l'IDE pour ESP 8266

Sélectionner la carte dans la liste



# Configuration de l'IDE pour ESP 8266

## Sélectionner le port de communication



# Introduction à la programmation

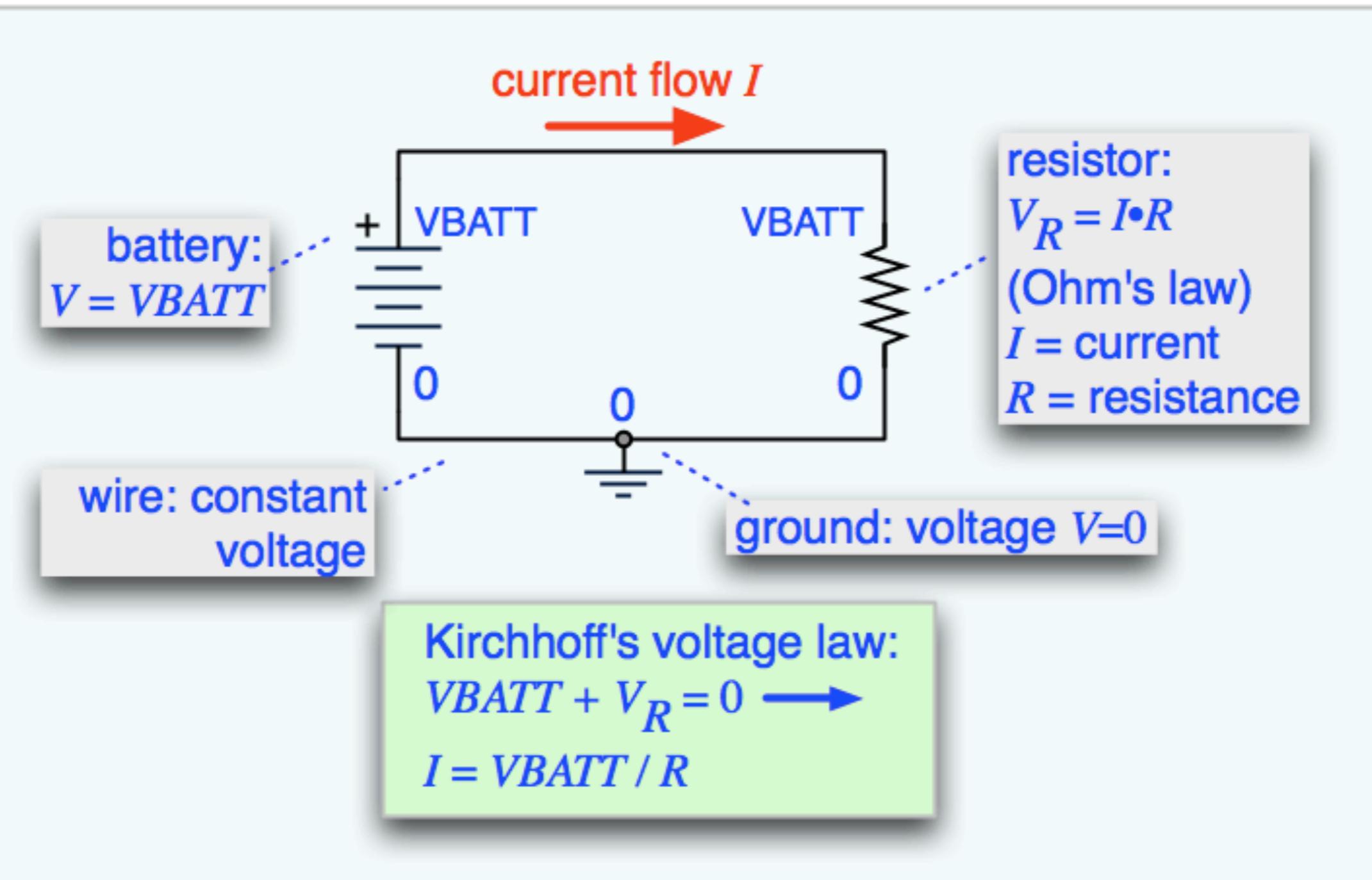
# Concepts de base

- Instructions
- Variables
- Conditions
- Boucles
- Fonctions
- Bibliothèques

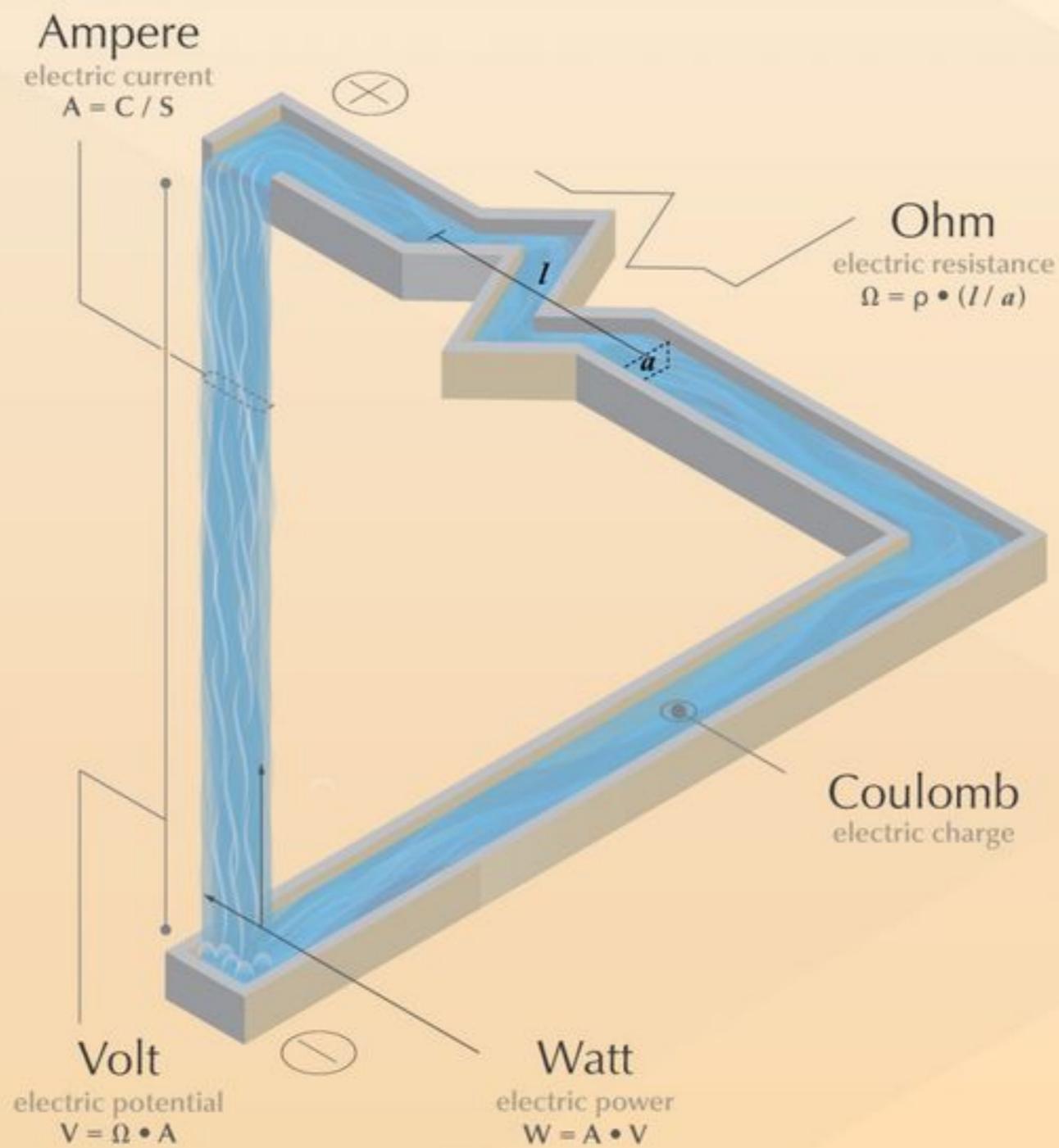
# Introduction à la programmation Arduino

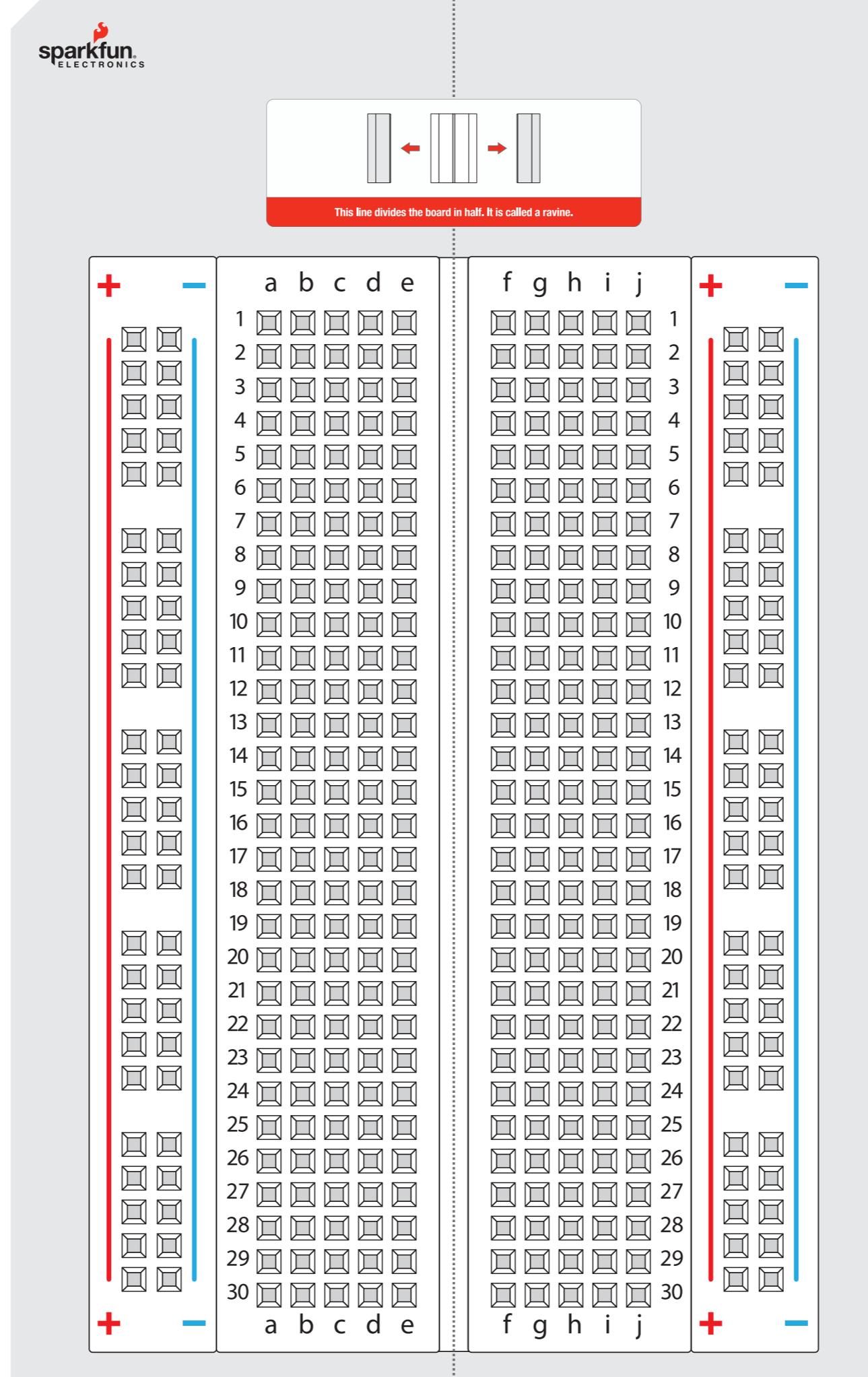
# Concepts de base

- Circuit
- La loi d'Ohm
- La breadboard (ou platine d'essai)

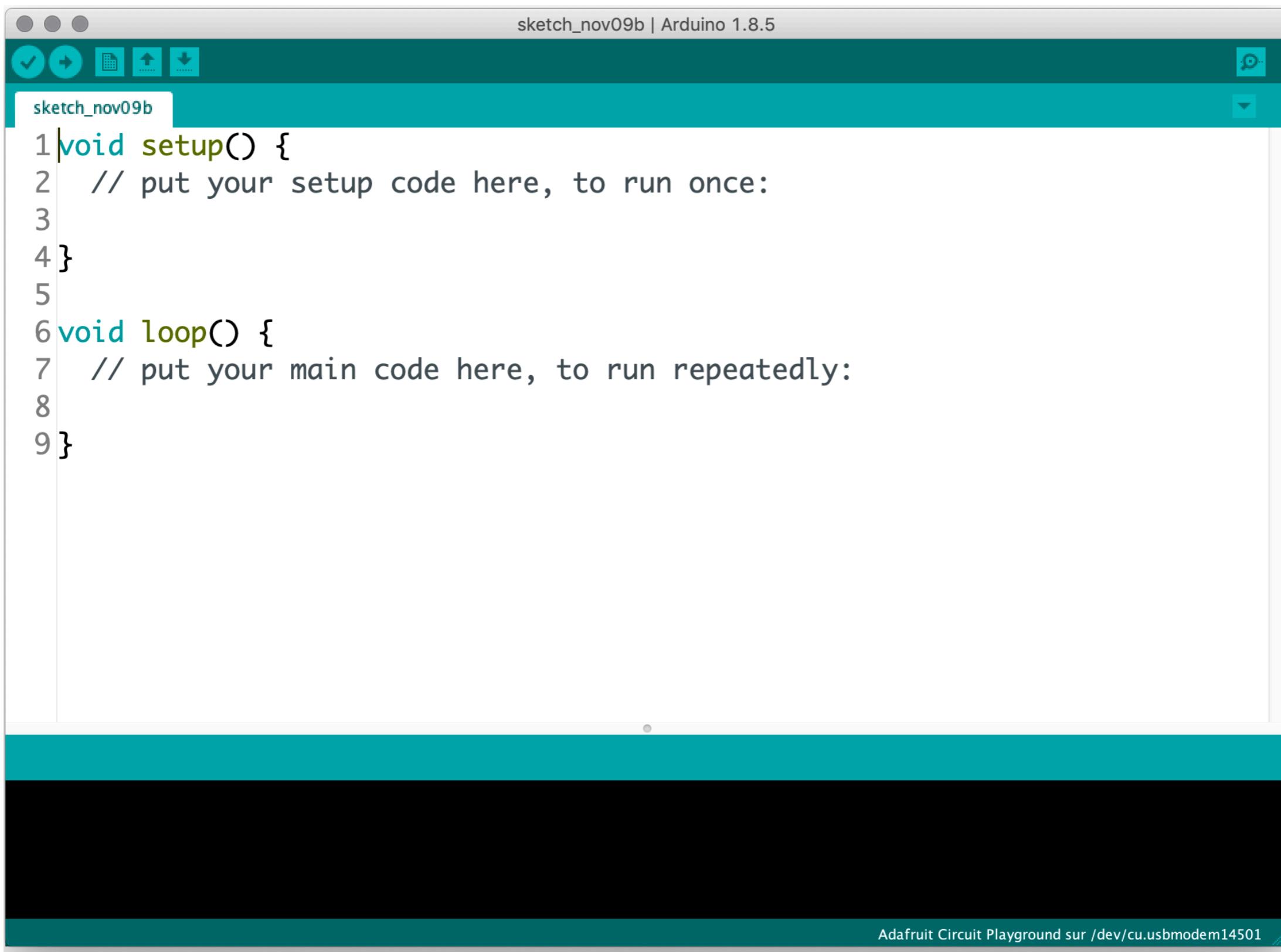


# - ELECTRICITY -





# Structure de base d'un programme Arduino



The screenshot shows the Arduino IDE interface with the title bar "sketch\_nov09b | Arduino 1.8.5". The main window displays the following code:

```
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8
9 }
```

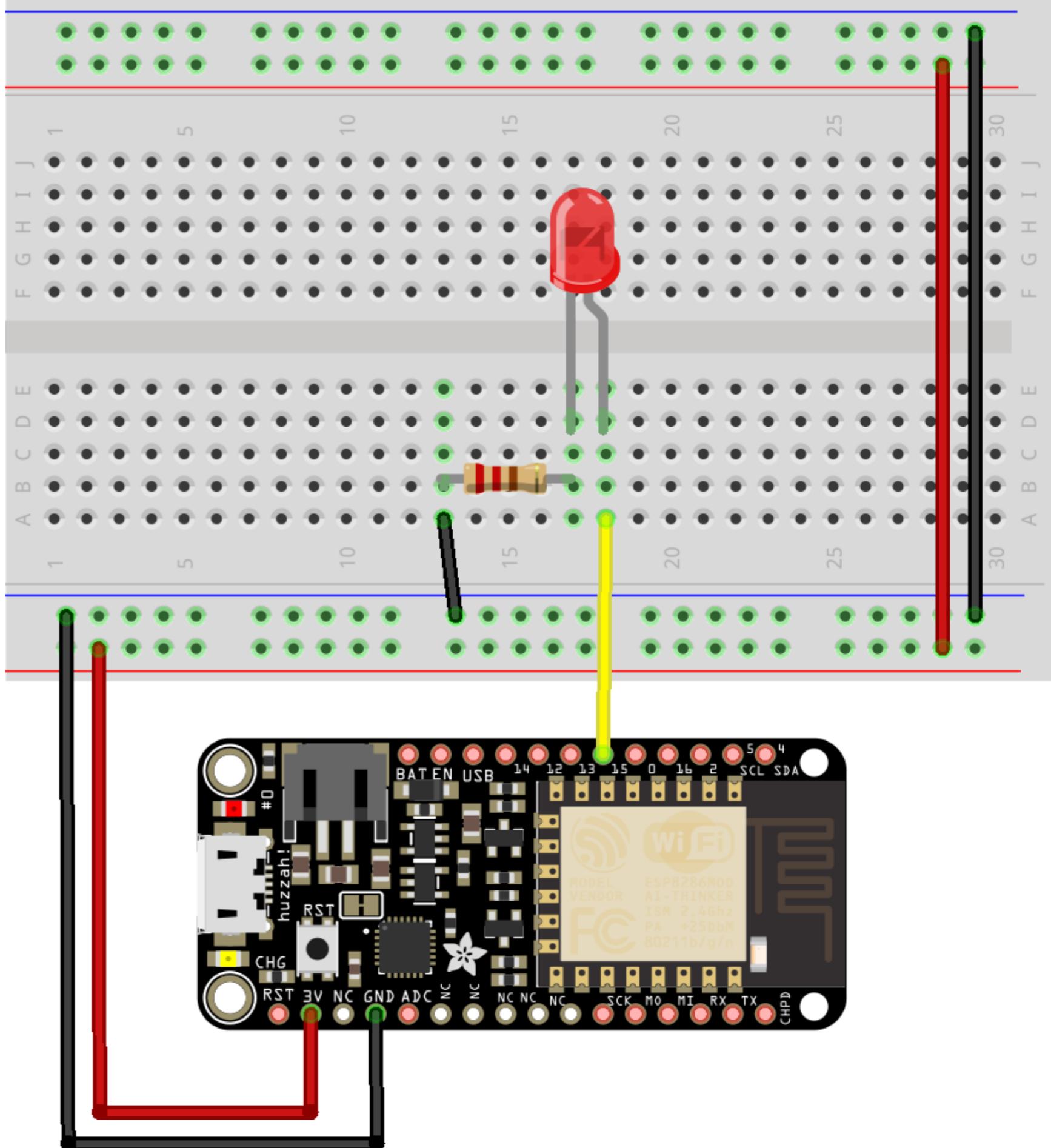
The code consists of two main functions: `setup()` and `loop()`. The `setup()` function is intended for one-time initialization code, while the `loop()` function contains the main logic that runs repeatedly.

Premiers montages

# 1. Sorties Digitales

- `pinMode(pin#, OUTPUT);`
  - configure la direction de la broche digitale **pin#**
  - second paramètre **OUTPUT** (utilisé pour la sortie)
- `digitalWrite(pin#, HIGH/LOW);`
  - modifie l'état de la broche **pin#**
  - le second paramètre peut être soit **HIGH** soit **LOW**

# Une LED



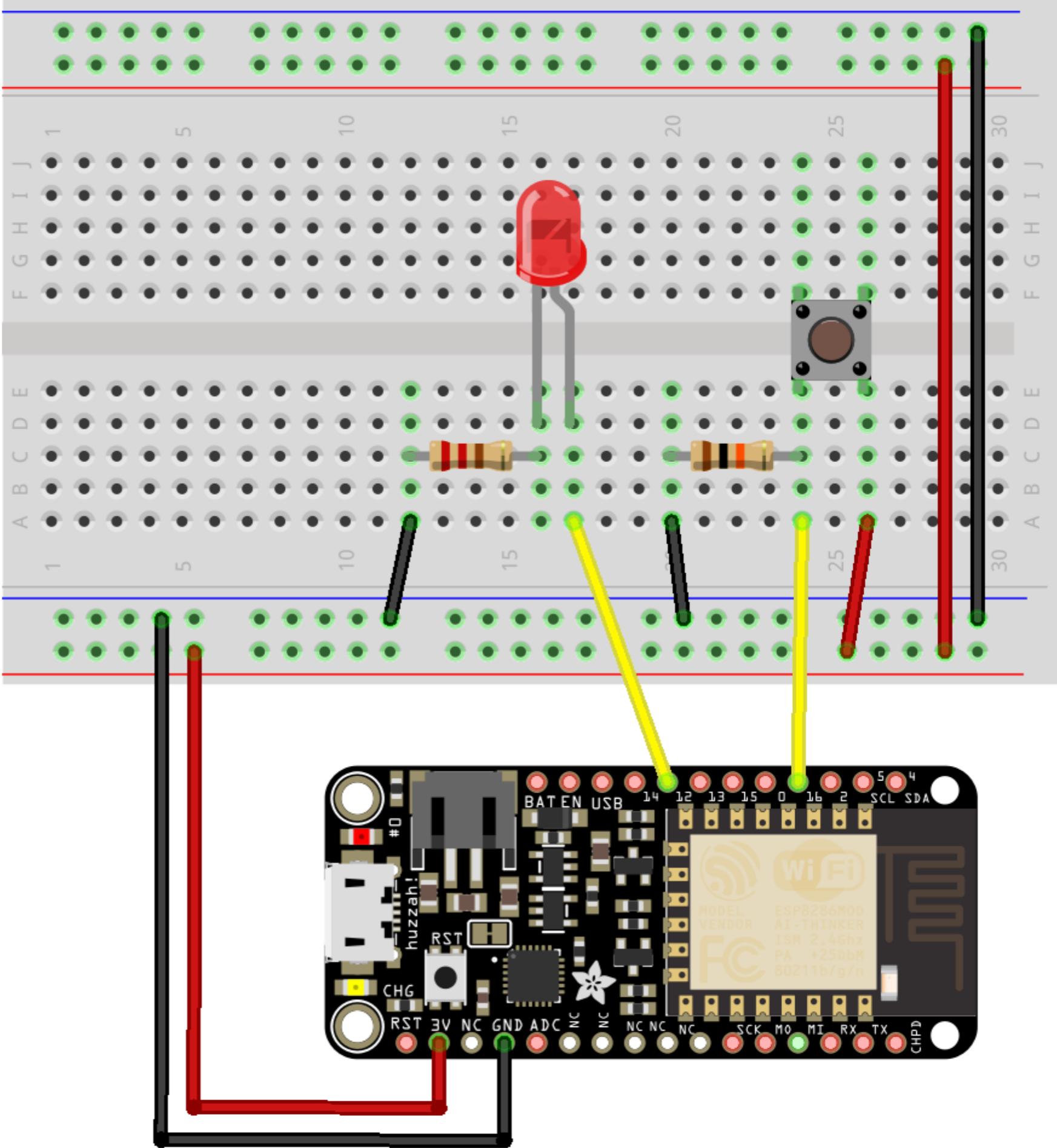
# Exercices

- Faire clignoter la LED à des vitesses différentes

# 2. Entrées Digitales

- `pinMode(pin#, INPUT);`
  - configure la direction de la broche digitale **pin#**
  - le second paramètre peut être **INPUT**,  
**INPUT\_PULLUP**
- `digitalRead(pin#);`
  - lit l'état de la broche **pin#**

LED +  
bouton



fritzing

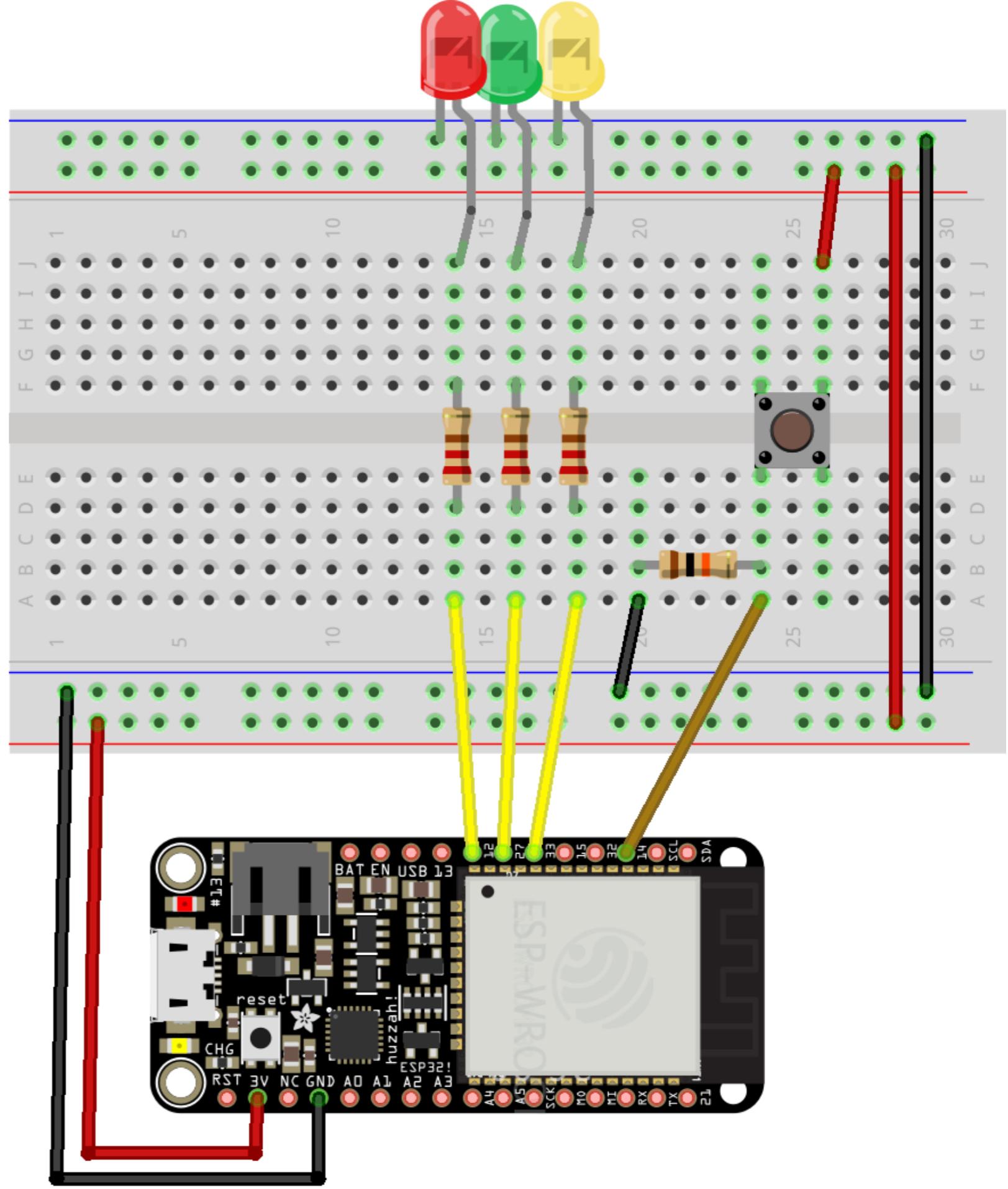
# Exemples

- Allumer la LED en fonction de l'état du bouton
- Debounce (anti-rebond)
- Détection de changement d'état

# Exercices

- Allumer la LED après 3 appuis sur le bouton
- Changer l'état de la LED au moment de l'appui sur le bouton. Quand le bouton est lâché, la led conserve son état.

# Plusieurs LEDs



# Exemples

- Utiliser les boucles pour commander l'état de plusieurs LEDS

# Exercices

- Utiliser les LEDS pour visualiser le nombre d'appuis sur un bouton