

Introduction à Arduino et l'internet des objets

Trakk Namur - Novembre et Décembre 2018

Présentation du Workshop

- Séance 1 : Introduction et première approche
- Séance 2 : Bases de la programmation
- Séance 3 : Bases de la programmation
- Séance 4 : IoT : communication, collecte et traitement des données
- Séance 5 : IoT : communication, collecte et traitement des données

Qui suis-je ?

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

Smart Cities

Enjeux

- Data (et open data)
- Bien être
- 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

- Corrélation de plusieurs facteurs
 - Coût des capteurs
 - Coût des processeurs
 - Coût de la bande passante
 - 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

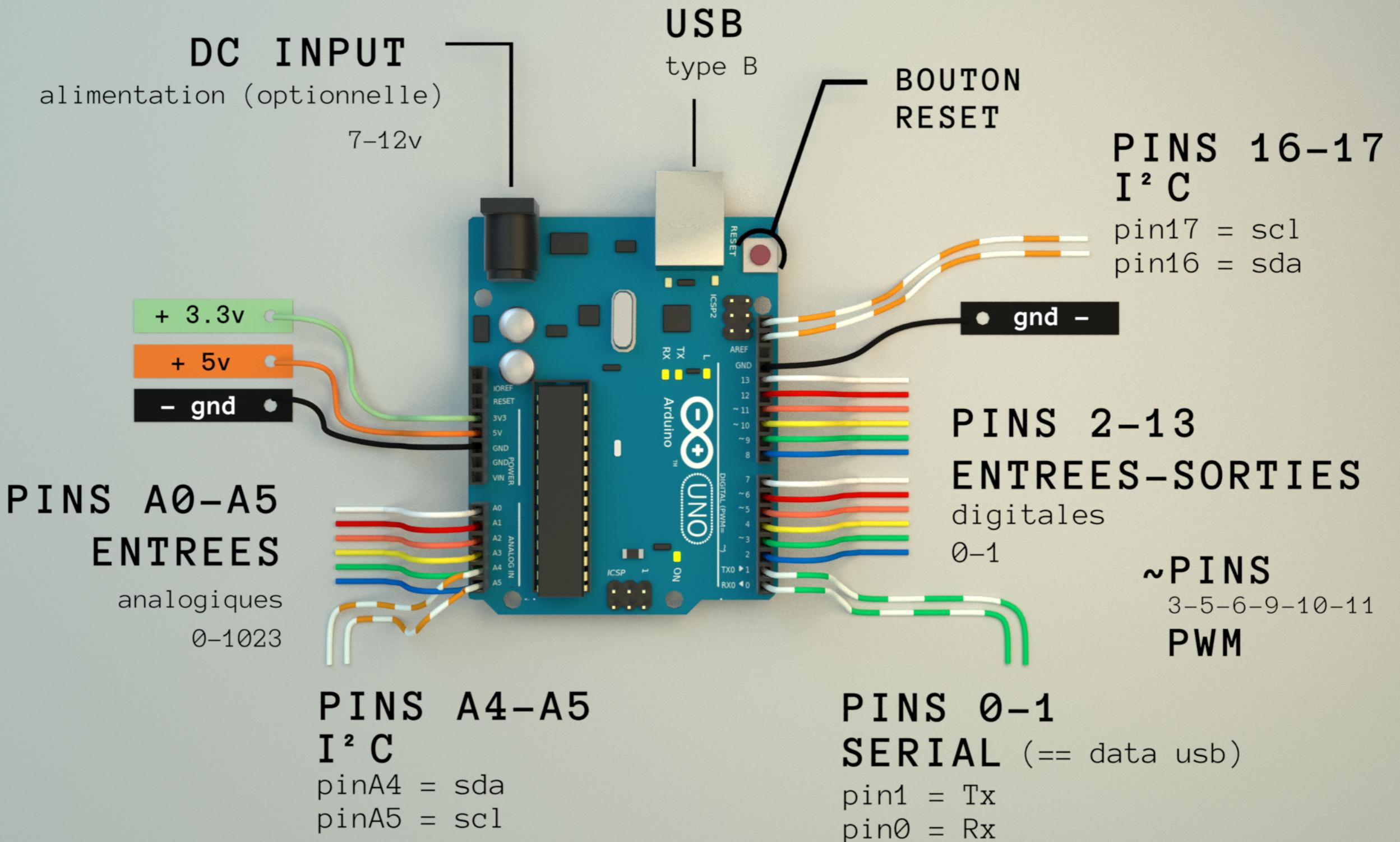
Présentation d'Arduino par l'un de ses créateurs : Massimo Banzi

Ted Talk : How Arduino is open sourcing imagination



Arduino UNO

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



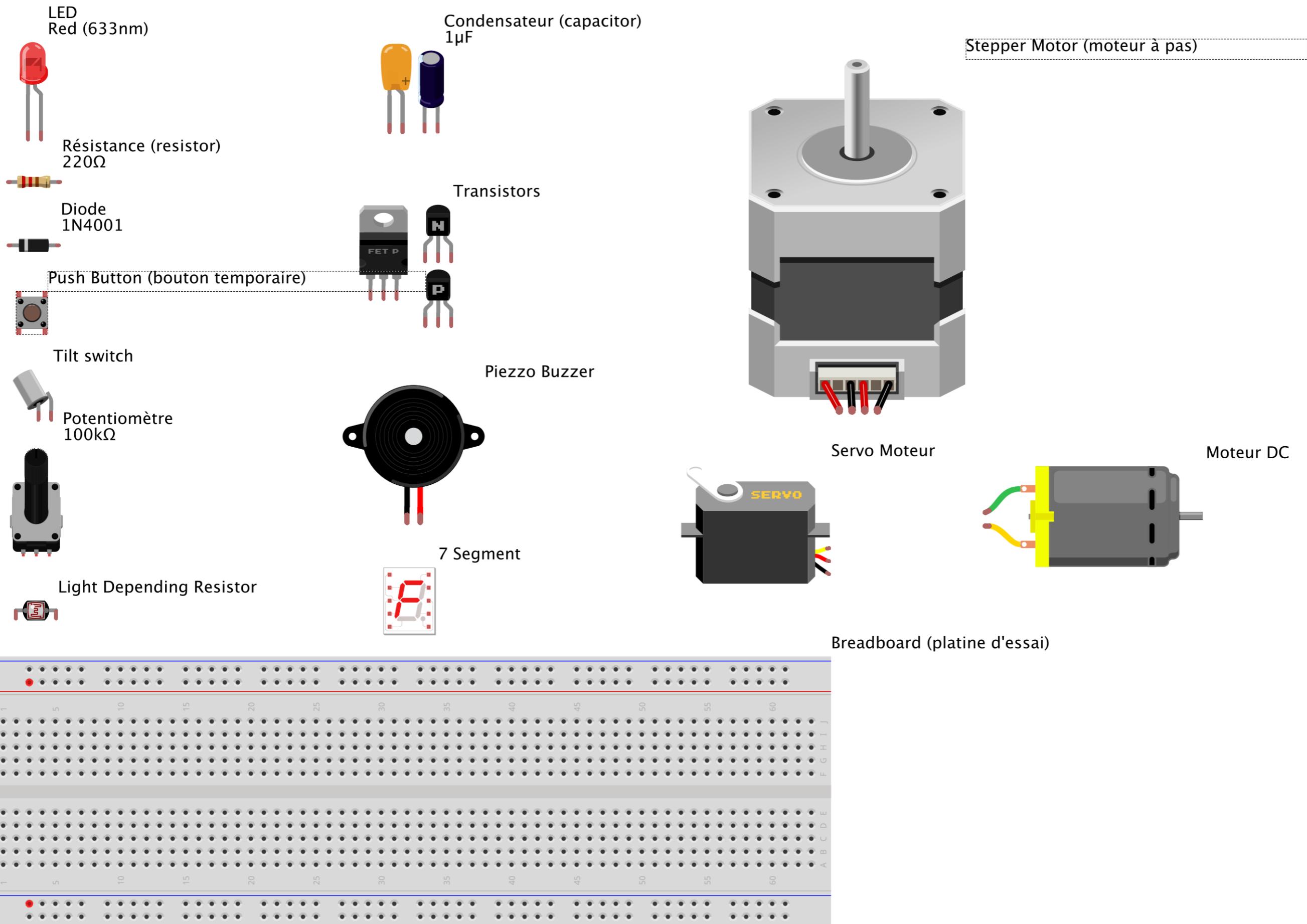
Adafruit Feather32

- TODO: insert features

TODO insert picture

Kit

- Kit de démarrage avec composants de base pour plusieurs montages
- TODO : détails Kit
-

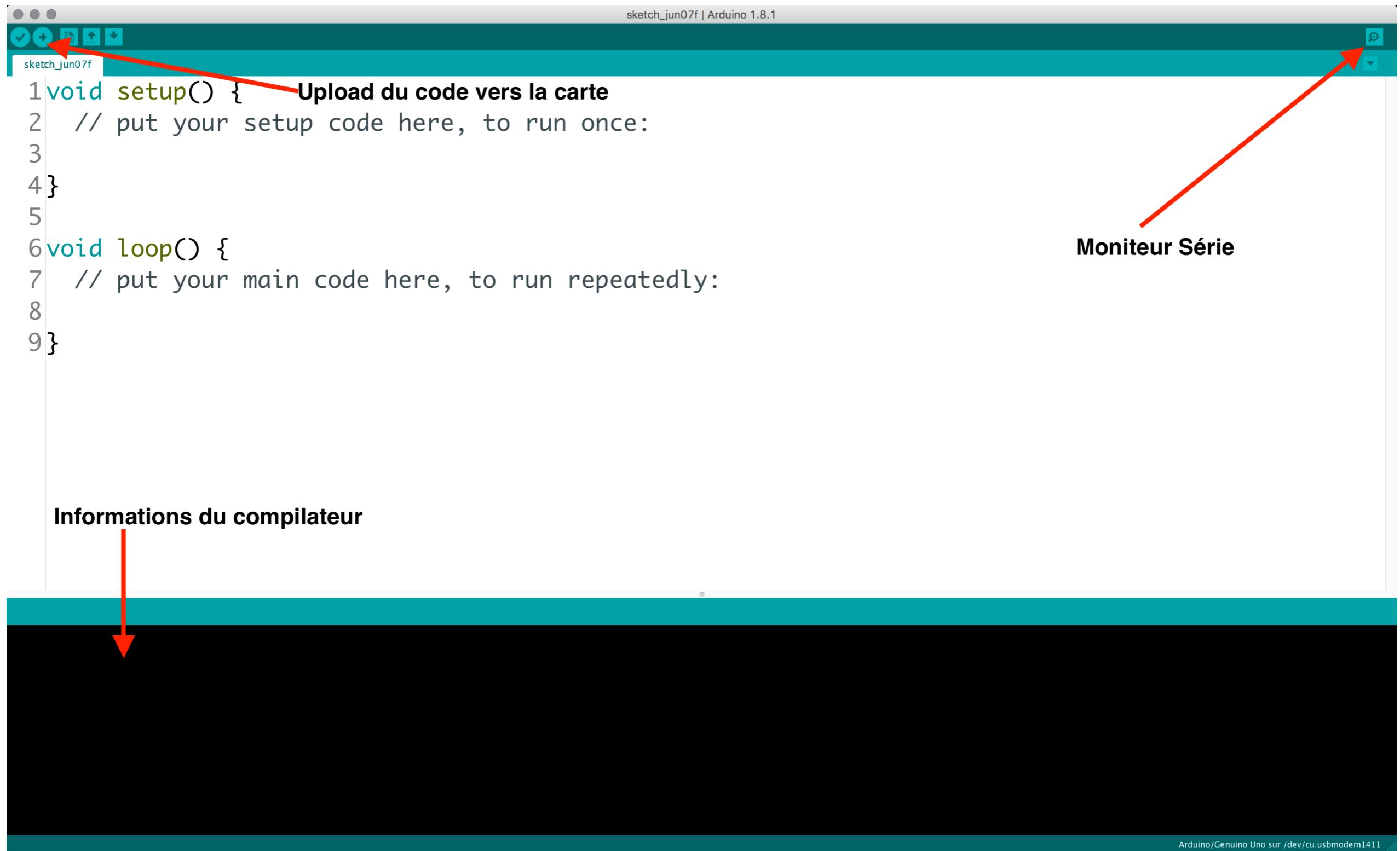


Installation et configuration des logiciels

<https://arduino.cc/downloads>

- **Web Editor:** exige l'installation d'un plugin permet de programmer la carte depuis une application web
- **IDE :** Integrated Development Environment, logiciel installé sur la machine

Présentation de l'IDE Arduino



Installation du support pour Adafruit Feather 32

The screenshot shows the Arduino IDE interface. In the foreground, a sketch titled "test-IR_V4_equalizer" is open in the code editor. The code includes #include "IRrecv.h", Adafruit_NeoPixel definitions, and IRrecv-related functions. In the background, the "Gestionnaire de carte" (Board Manager) is displayed, listing various Arduino boards. The "Adafruit Circuit Playground" board is selected, indicated by a checkmark.

```
#include "IRrecv.h"
#include <Adafruit_NeoPixel.h>
#ifndef __AVR_ATmega328P__
#define PIN 6
int strobe = 4; // strobe pins on digital pins
int res = 5; // reset pins on digital pins
int left[7]; // store band values in array
int right[7];
int band;
int receiver = 2;
int NbrsLed = 120;
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NbrsLed, PIN, receiver, _GRB + NEO_KHZ800);

```

Arduino Version: unknown

Gestionnaire de carte

- Teensyduino
- Teensy 3.6
- Teensy 3.5
- Teensy 3.2 / 3.1
- Teensy 3.0
- Teensy LC
- Teensy++ 2.0
- Teensy 2.0
- Arduino SAMD (32-bits ARM Cortex-M0+) Boards
- Arduino/Genuino Zero (Programming Port)
- Arduino/Genuino Zero (Native USB Port)
- Arduino/Genuino MKR1000
- Arduino MKRZero
- Arduino MKRFox1200
- Adafruit Circuit Playground Express
- Arduino MO Pro (Programming Port)
- Arduino MO Pro (Native USB Port)
- Arduino MO
- Arduino Tian
- Cartes Arduino AVR
- Arduino Yún
- Arduino/Genuino Uno
- Arduino Duemilanove or Diecimila
- Arduino Nano
- Arduino/Genuino Mega or Mega 2560
- Arduino Mega ADK
- Arduino Leonardo
- Arduino Leonardo ETH
- Arduino/Genuino Micro
- Arduino Esplora
- Arduino Mini
- Arduino Ethernet
- Arduino Fio
- Arduino BT
- LilyPad Arduino USB
- LilyPad Arduino
- Arduino Pro or Pro Mini
- Arduino NG or older
- Arduino Robot Control
- Arduino Robot Motor
- Arduino Gemma
- ✓ Adafruit Circuit Playground
- Arduino Yún Mini
- Arduino Industrial 101
- Linino One
- Arduino Uno WiFi
- Cartes Arduino ARM (32-bits)
- Arduino Due (Programming Port)
- Arduino Due (Native USB Port)

Adafruit Circuit Playground sur /dev/cu.usbmodem146401

Installation du support pour Feather32

Todo : insert screenshot

Introduction à la programmation

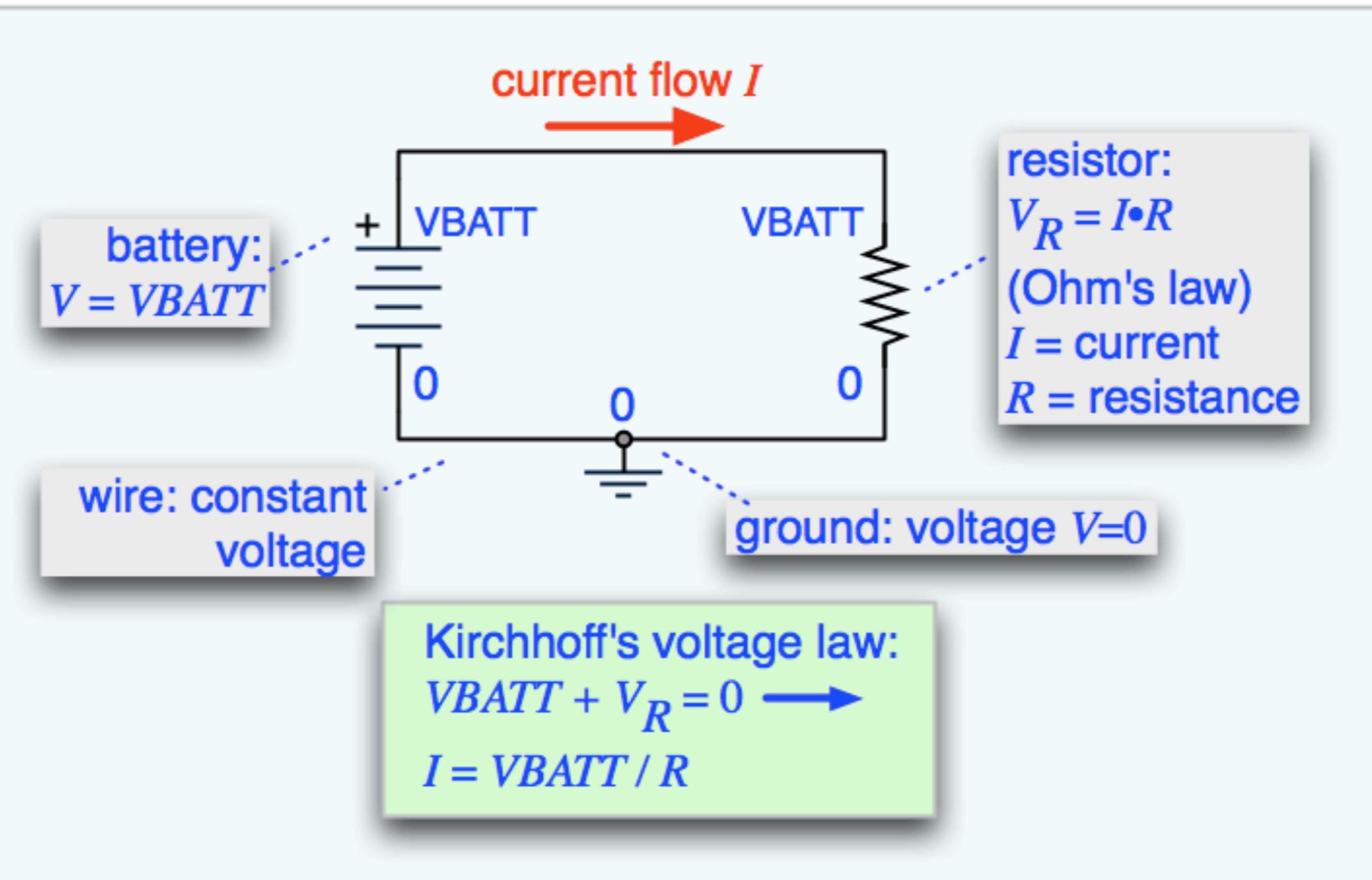
Concepts de base

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

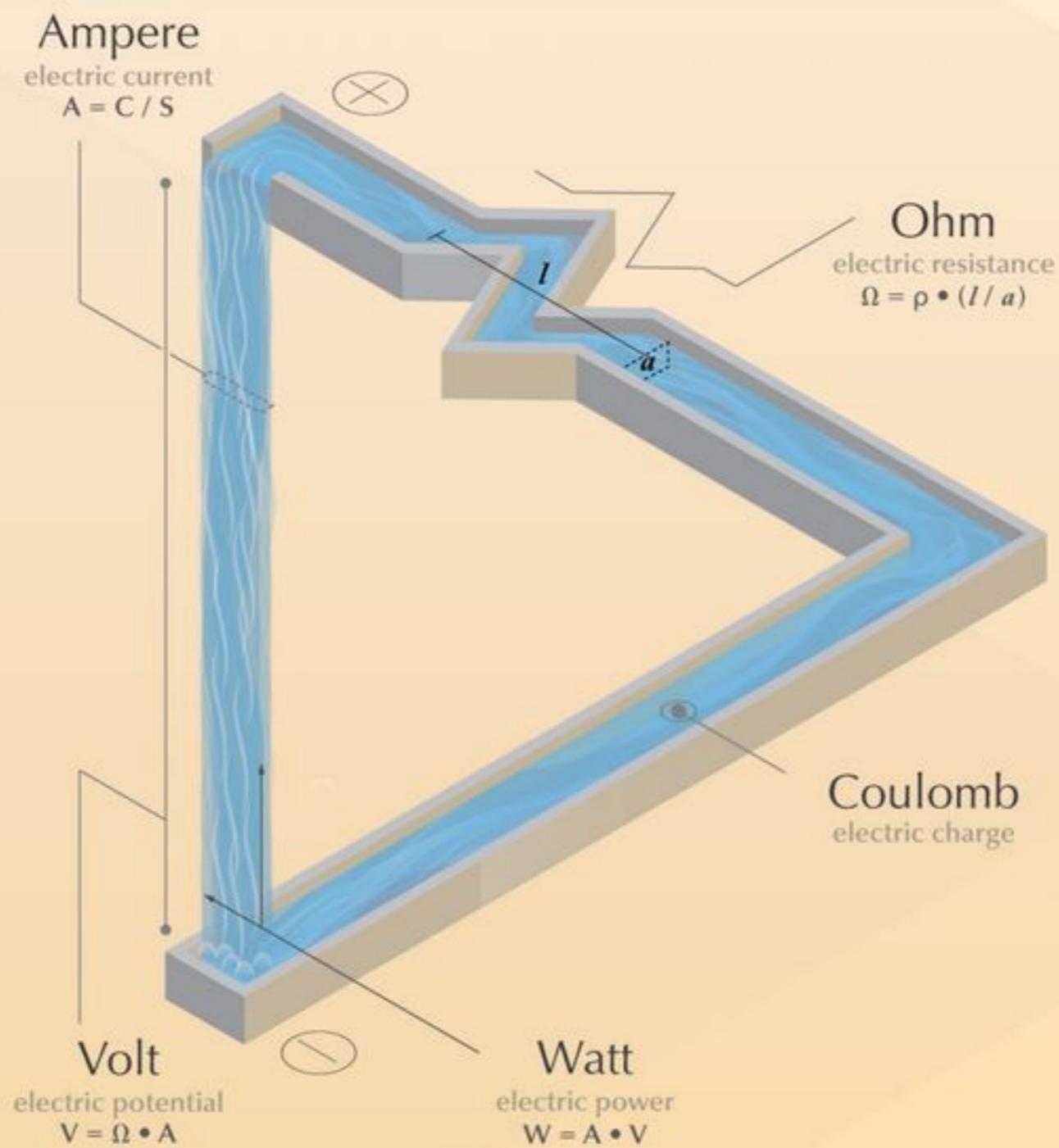
Introduction à l'électronique

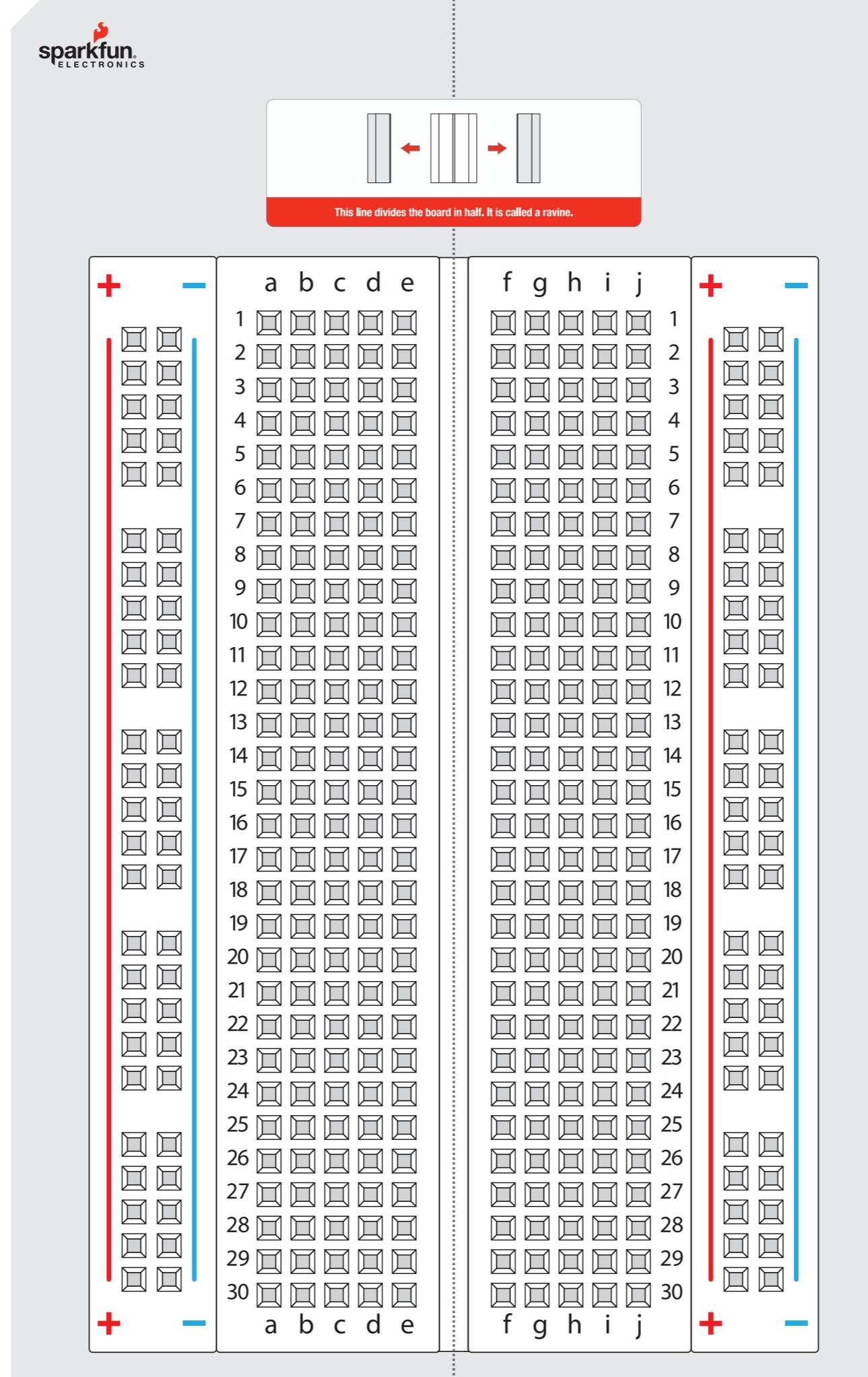
Concepts de base

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

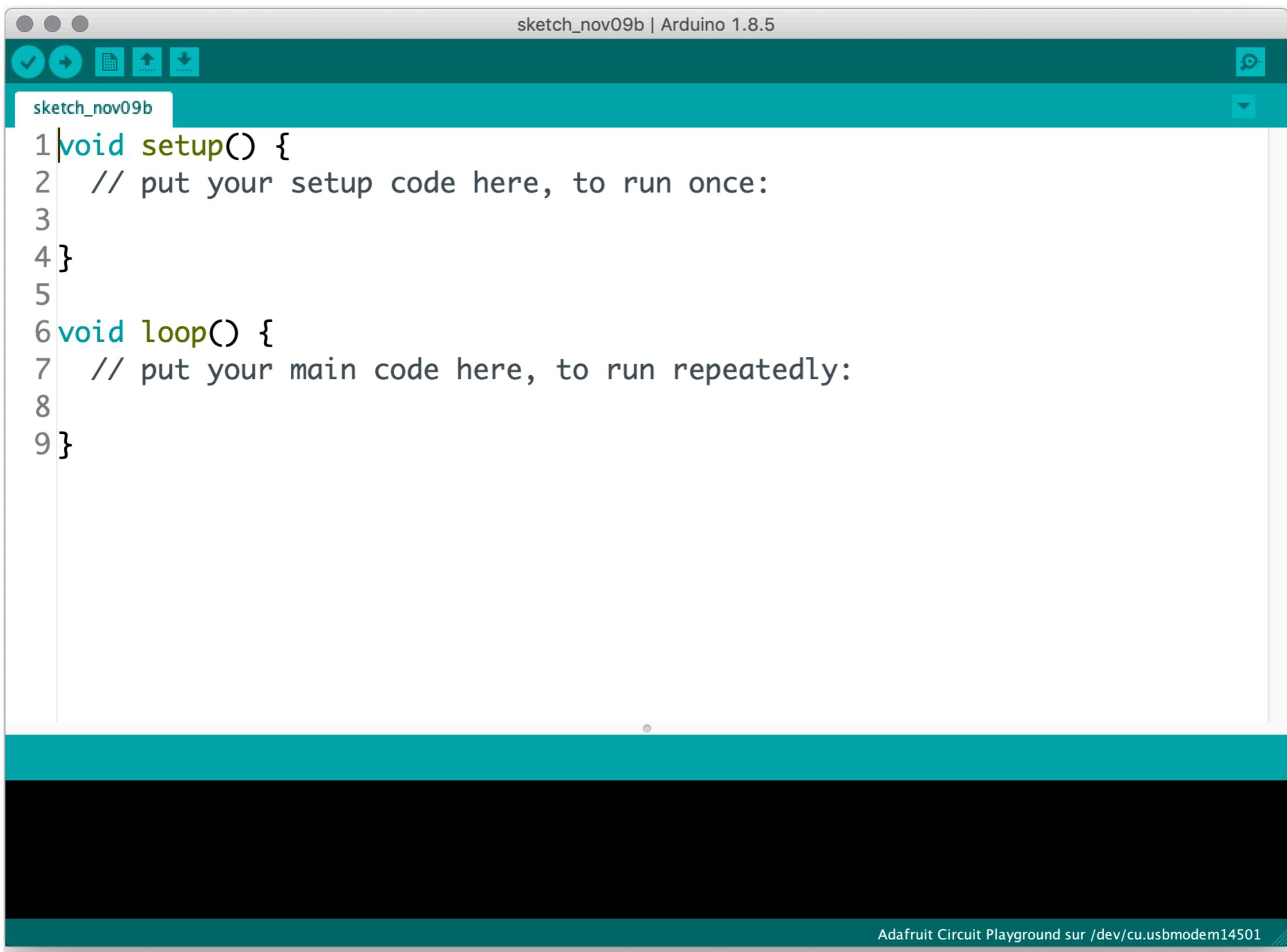


- ELECTRICITY -





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 }
```

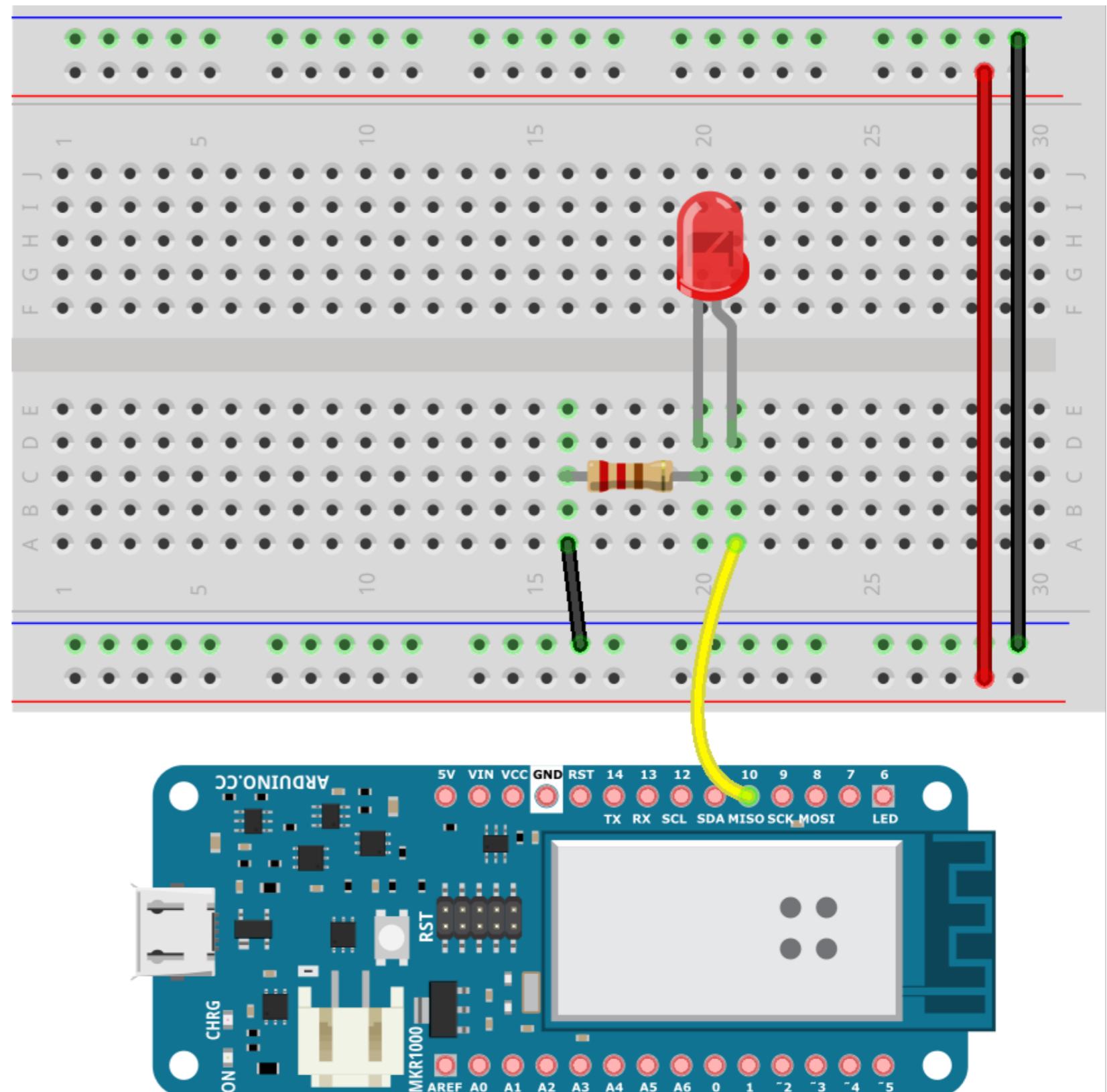
The code consists of two main sections: `setup()` and `loop()`. The `setup()` section is intended for one-time initialization code, and the `loop()` section is for repeated execution.

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



fritzing

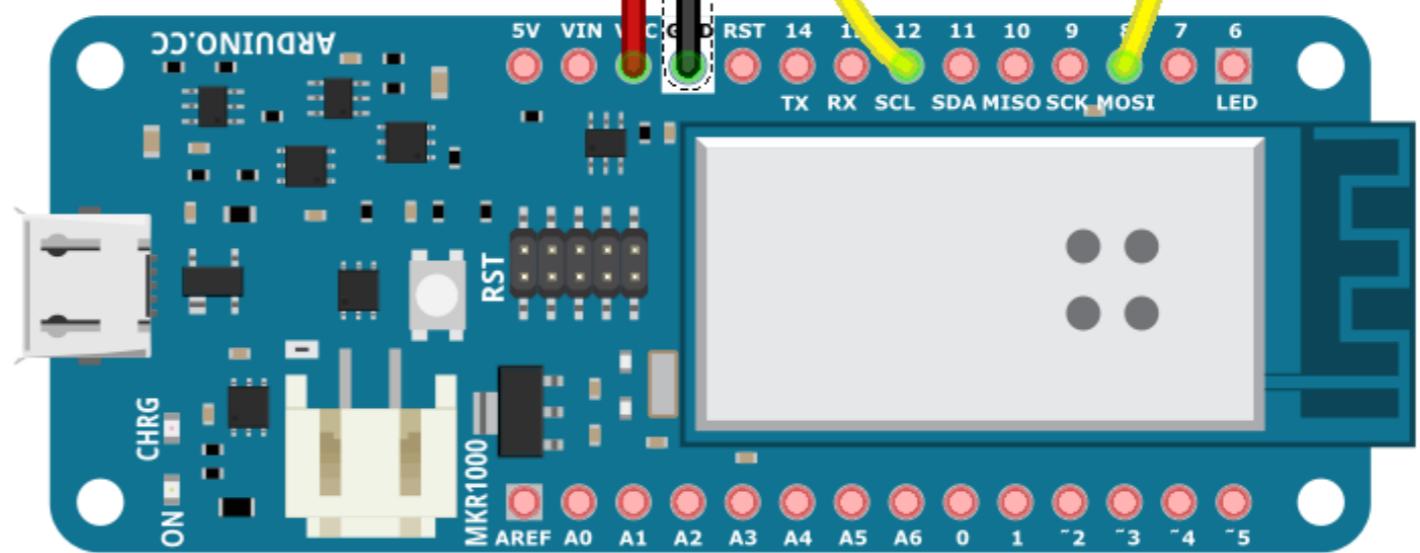
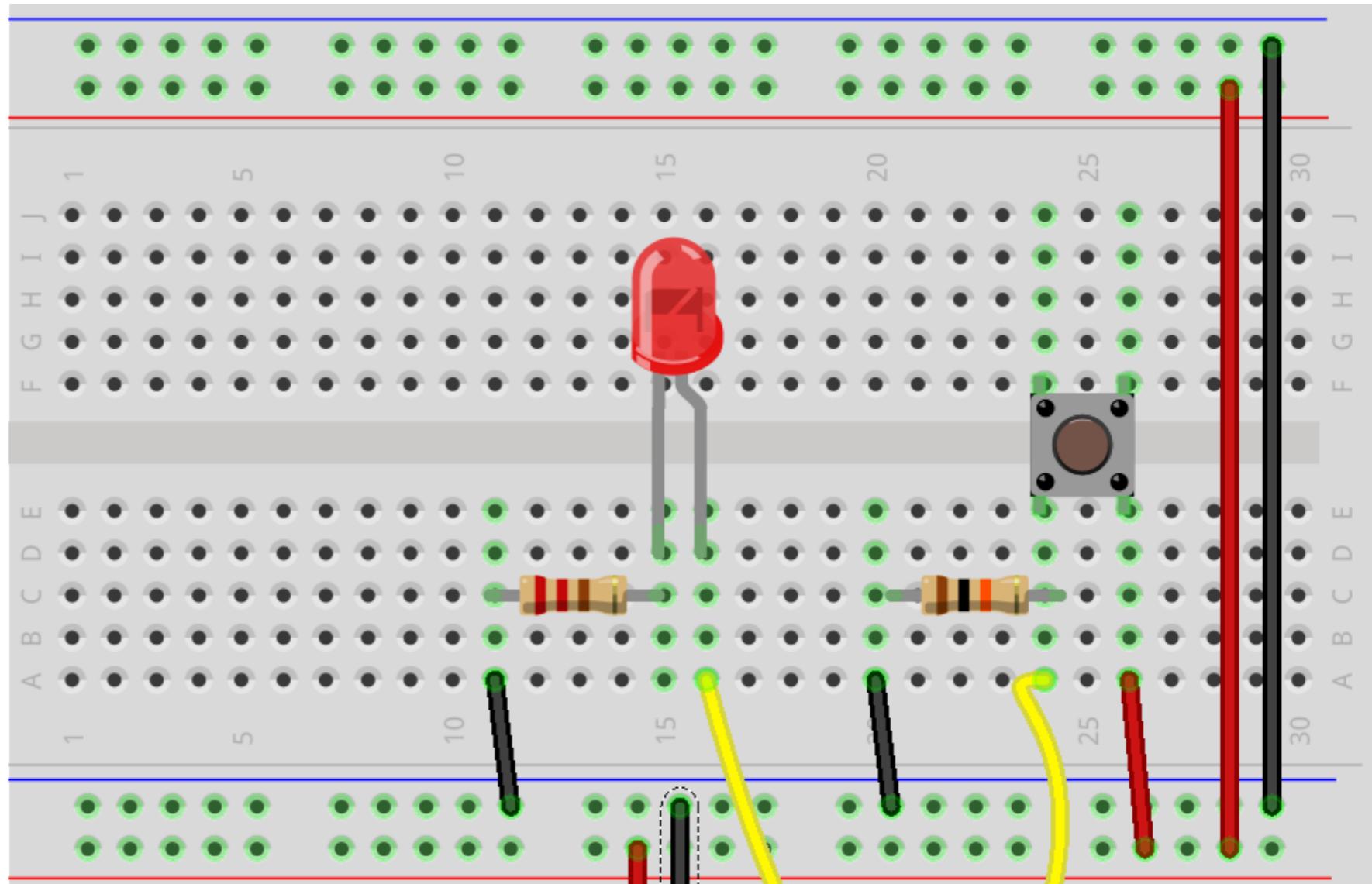
Exercices

- 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

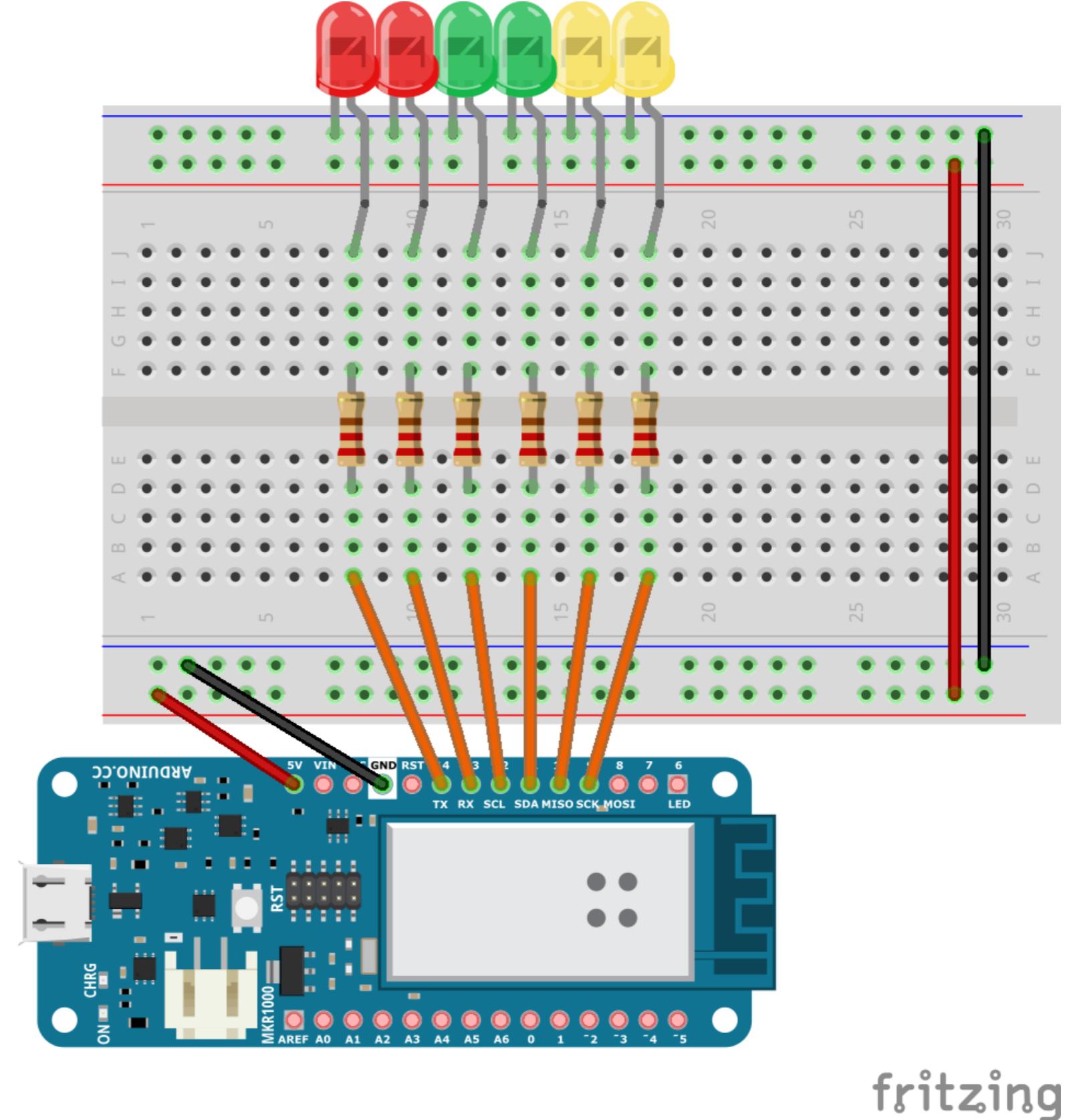
Exemples

- 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



fritzing

Exemples

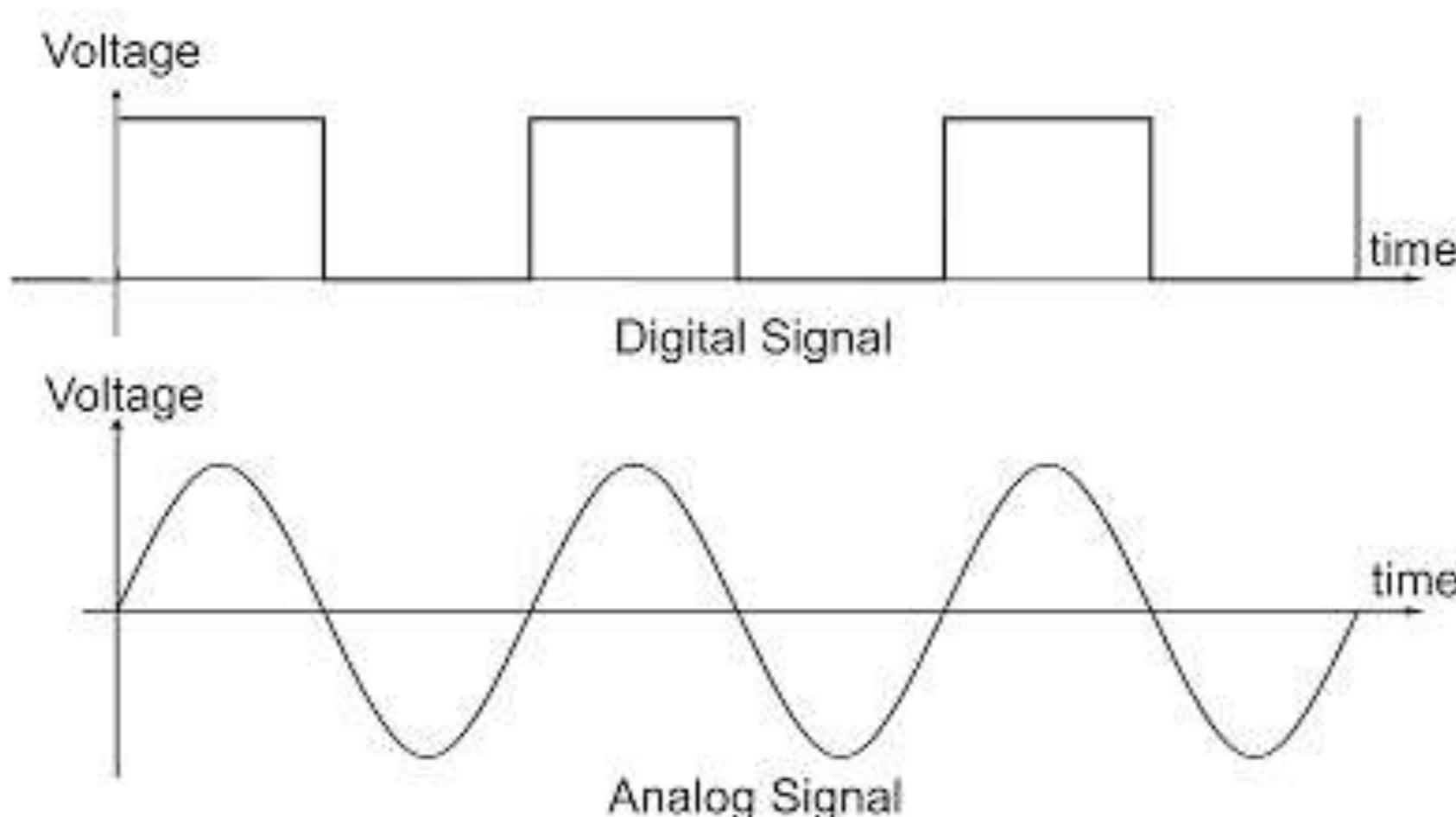
- Utiliser les boucles pour commander l'état de plusieurs LEDS

Exercices

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

3. Entrées Analogiques

- Signal analogique vs Digital



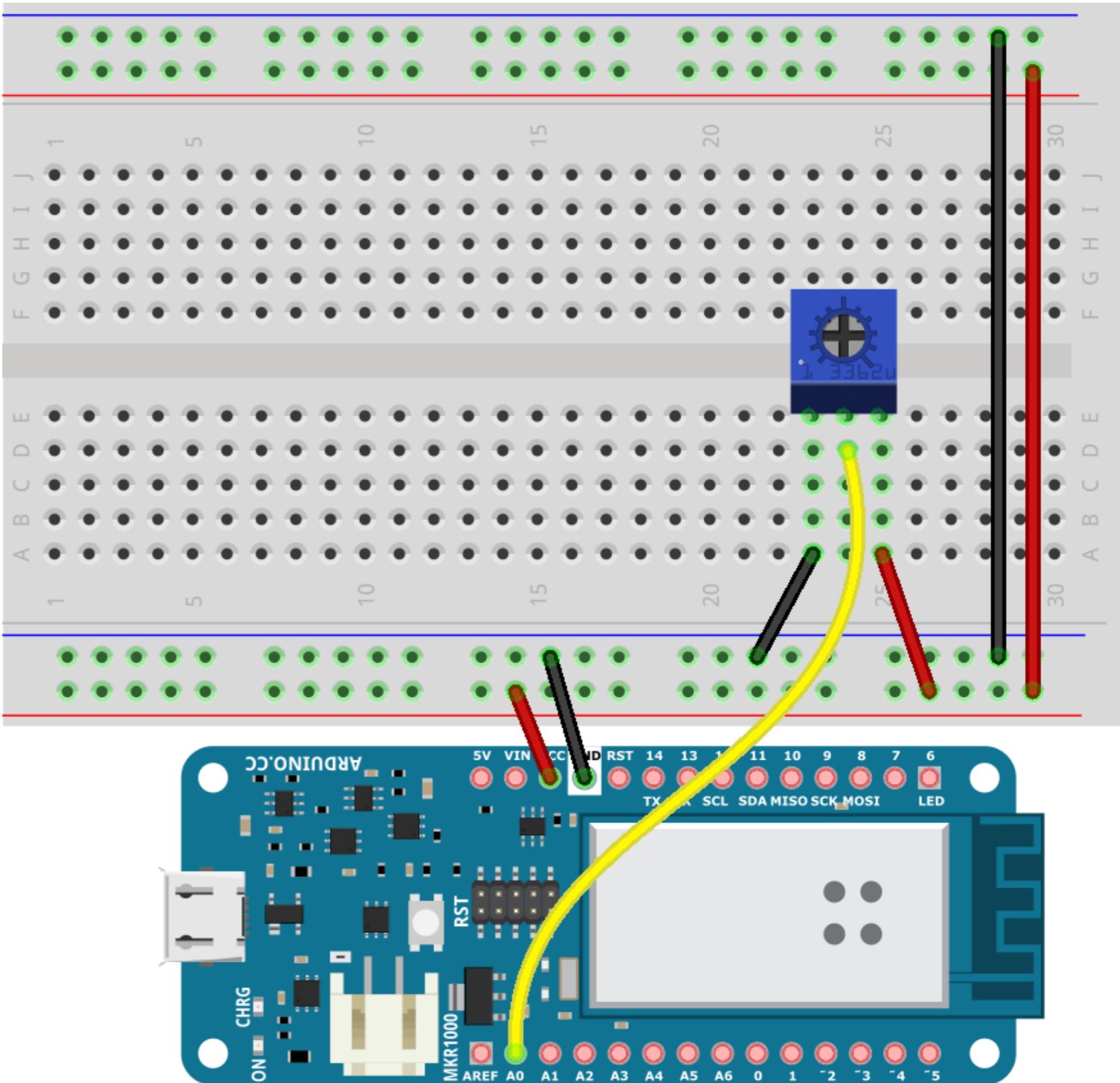
3. Entrées Analogiques

- 7 entrées analogiques sur le MKR1000
 - Numérotées de A0 à A6
 - Mesure une variation de la tension appliquée sur la broche et la convertit en une valeur numérique

3. Entrées Analogiques

- ADC 8/10/12 bits configurable
(analogReadResolution(bits))
- **analogRead(pin#);**
 - lit l'état de la broche **pin#**
 - retourne une valeur entre 0 et 1024 (10 bits, par défaut)

Potentiomètre



fritzing

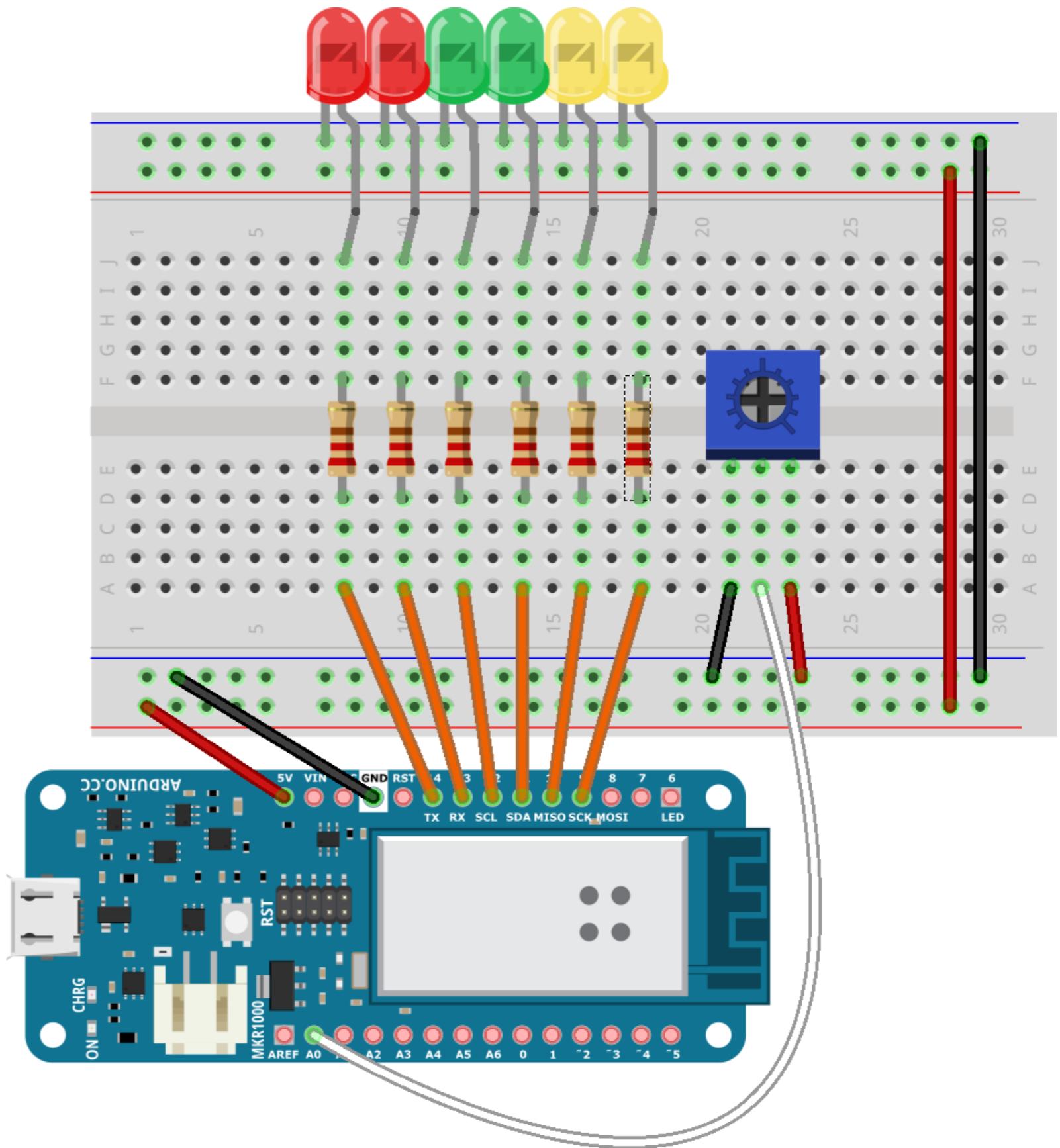
La librairie Serial

- Permet de faire communiquer la carte Arduino et l'ordinateur via l'USB
- dans **setup()** :
 - **Serial.begin(*rate*);**
 - initialise la communication à une vitesse de ***rate*** bits/seconde
- dans **loop()**
 - `Serial.println("texte ou variable");`
 - Affiche un texte ou la valeur d'une variable dans le moniteur série

Exercices

- Utiliser le potentiomètre pour faire varier la vitesse de clignotement d'une LED

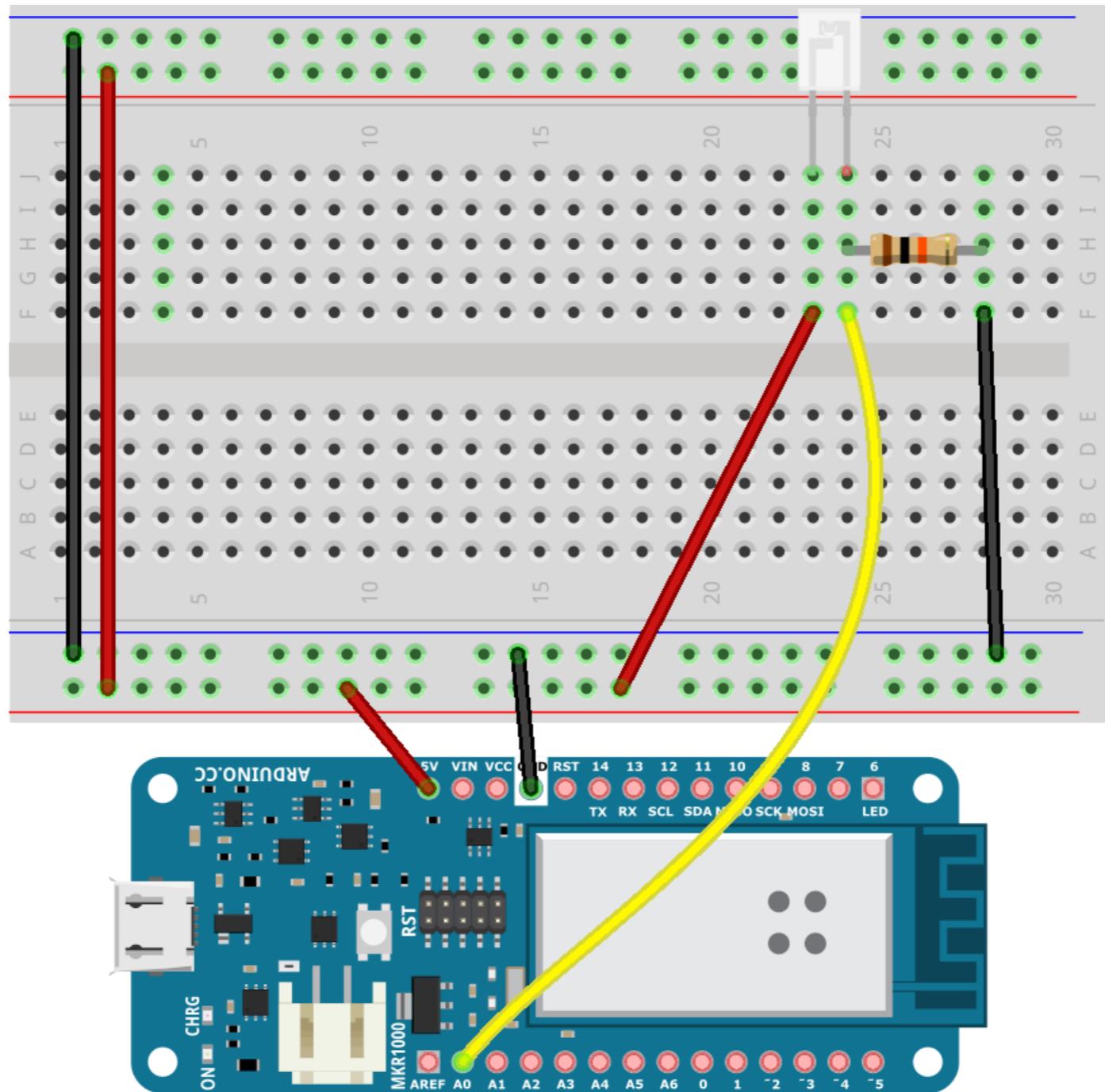
Potentiomètre + LEDs



Exercices

- Utiliser le potentiomètre pour allumer plus ou moins de LEDs

Phototransistor (pont diviseur de tension)



fritzing

Exemples

- La fonction **map**
- La calibration