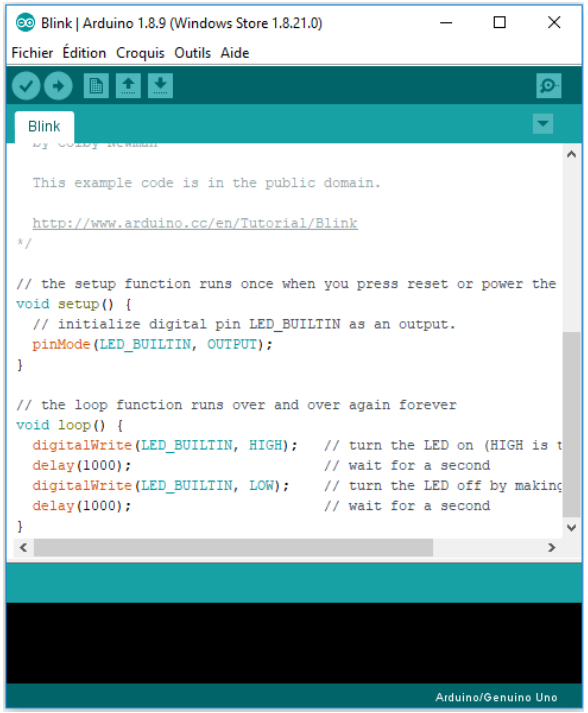
Formation

Initiation à l'utilisation d'un microcontrôleur pour la physique-chimie.

ARDUINO™



Programme de la journée

Matin

- Présentation générale
- Installation du logiciel Arduino
- La carte Arduino
- Branchement et test de la carte
- Programmation du microcontrôleur (les bases)

Après-midi

- Test de plusieurs capteurs (TP)
- Ecriture d'un TP 2nde
- Autre exemple de microcontrôleur : Trinket M0

Objectifs de la formation

- Découvrir le microcontrôleur Arduino
- Savoir utiliser un programme Arduino
- Savoir comment l'utiliser en physique-chimie
- Fournir des idées de TP
- Savoir qu'il existe d'autre carte qu'Arduino

Bibliographie

- https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino

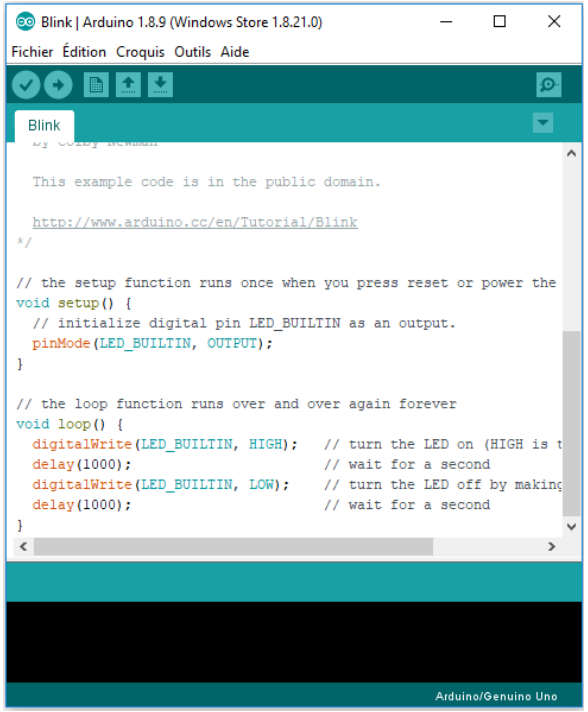
- « Démarrer avec Arduino, principe de base et premiers montages », Massimo Banzi, ETSF.

- « Arduino, Maîtrisez sa programmation et ses cartes d'interface (shield) », Christian Tavernier, DUNOD.

1

Qu'est-ce qu'Arduino?

ARDUINO™

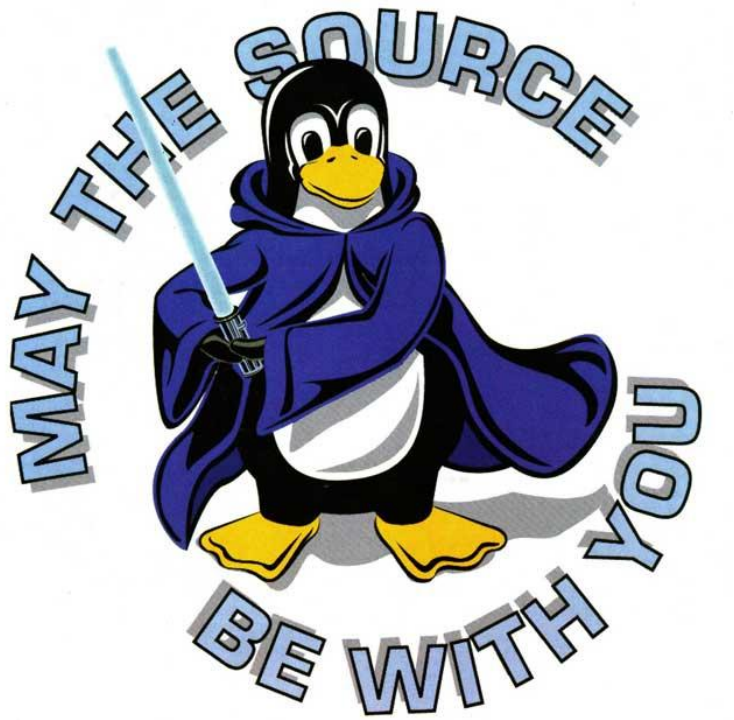


1 Historique

- Créé par **Massimo Banzzi**
  - Interaction Design Institute Ivrea (IDII) à Ivrea, en Italie
  - Fait par des designer pour des designer
- Date de naissance : 2003
- Dérivé du projet « Processing » créé au MIT (Casey Reas et Ben Fry en 2000)
- **Objectif** :
  - Carte Microcontrôleur **pas cher**
  - **Facile à programmer**

ARDUINO™

open source™



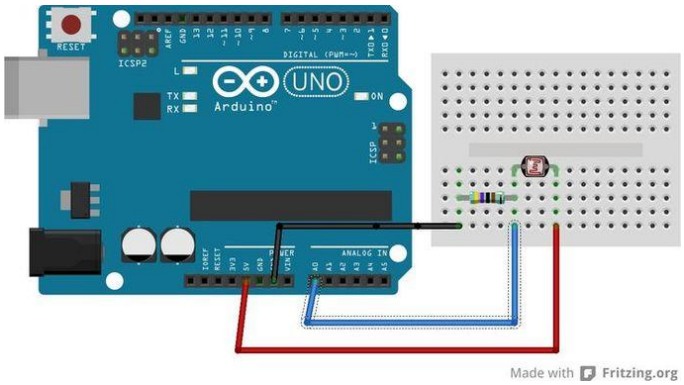
- **Plan de la carte libre d'accès**
  - Plusieurs modèles disponibles…

1 Pour quoi faire ?

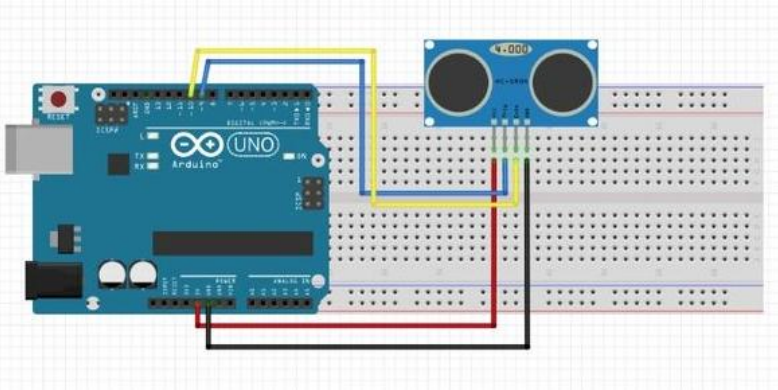
➡ Plate-forme de prototypage d'objets interactifs

BIP !!

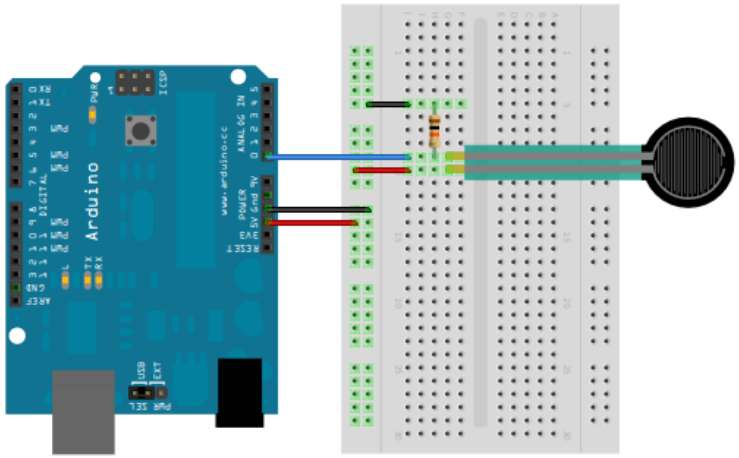
Plate-forme de prototypage d'outils de mesure



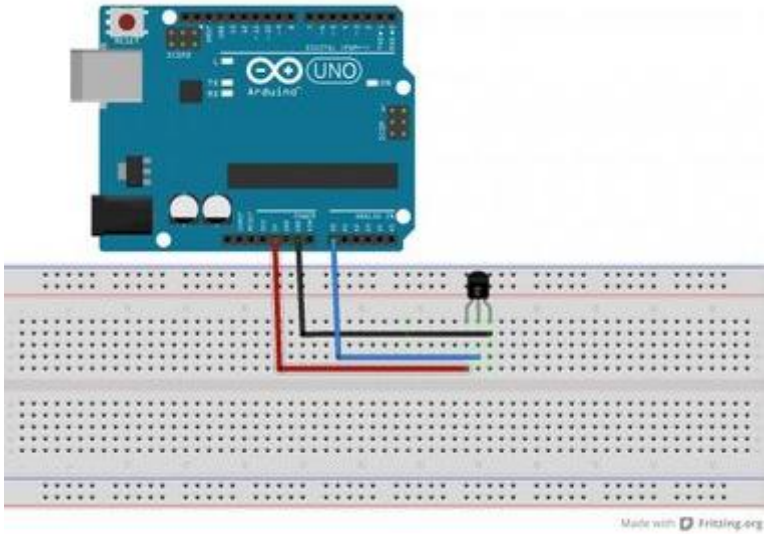
Intensité lumineuse



Distance (E/R ultrason)

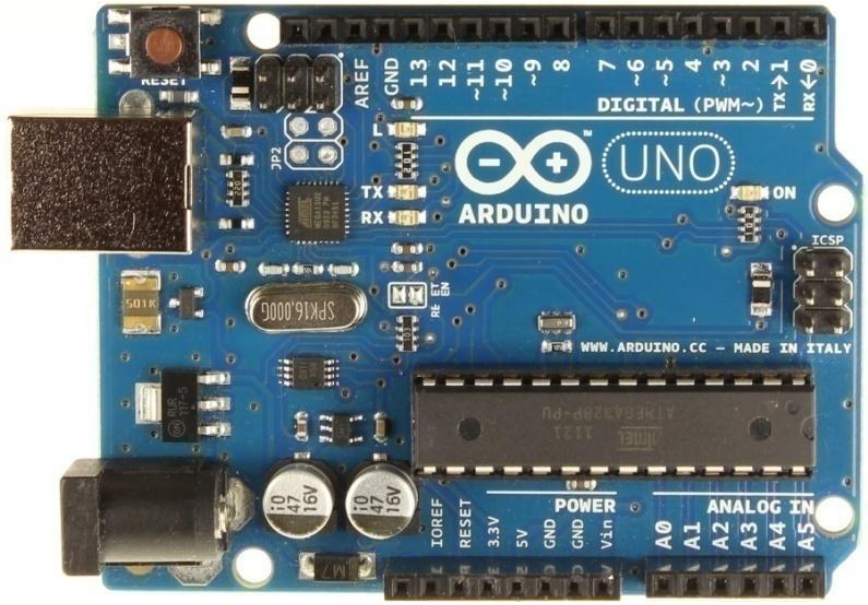


Intensité d'une Force

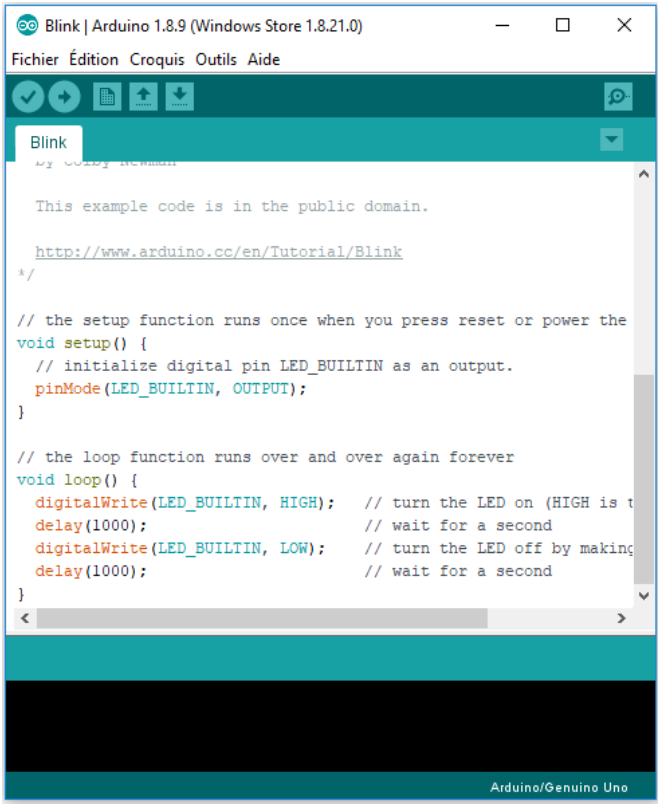


Température

1 Eléments constitutifs



Carte électronique



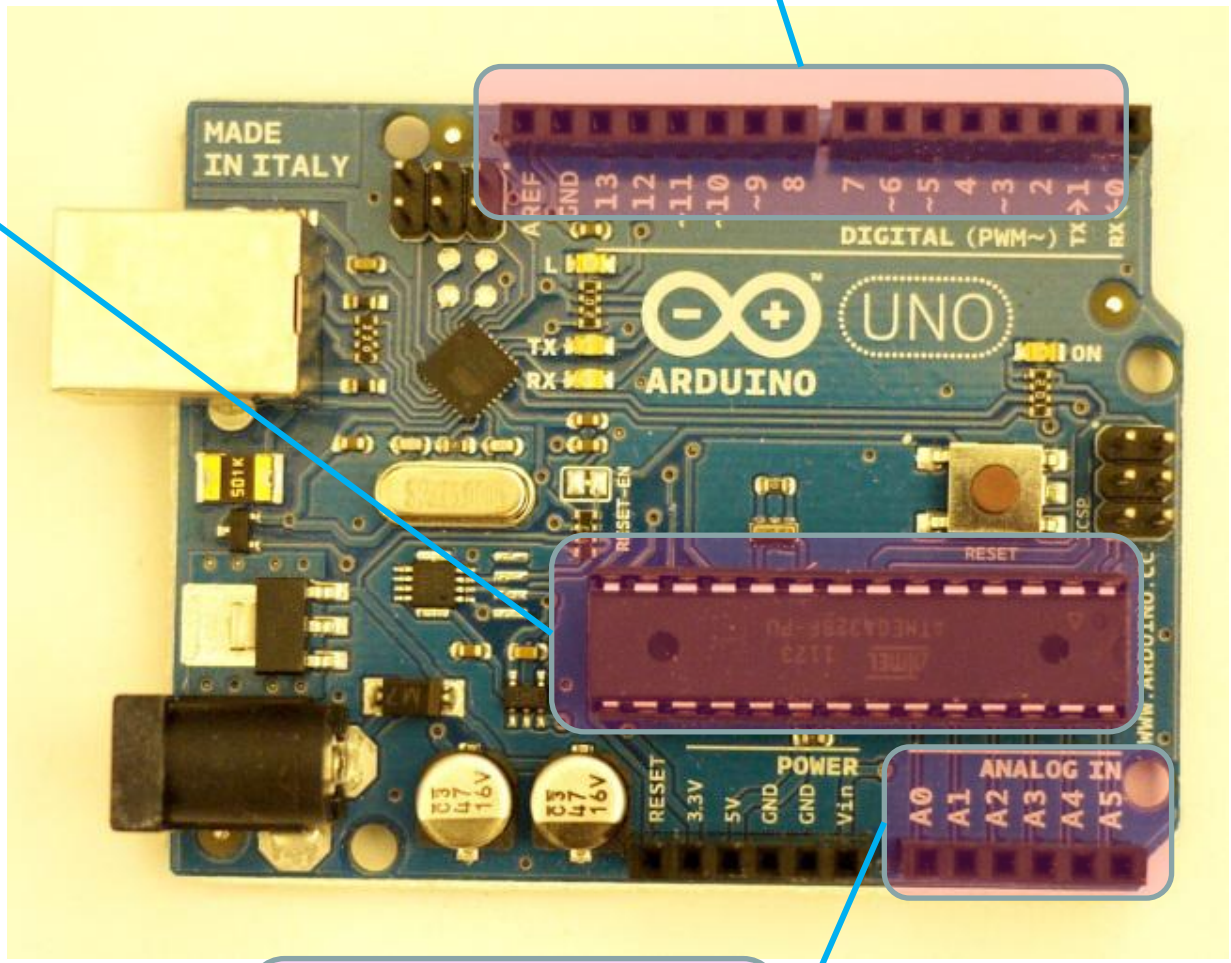
Environnement de programmation

IDE : *Integrated Development Environment*

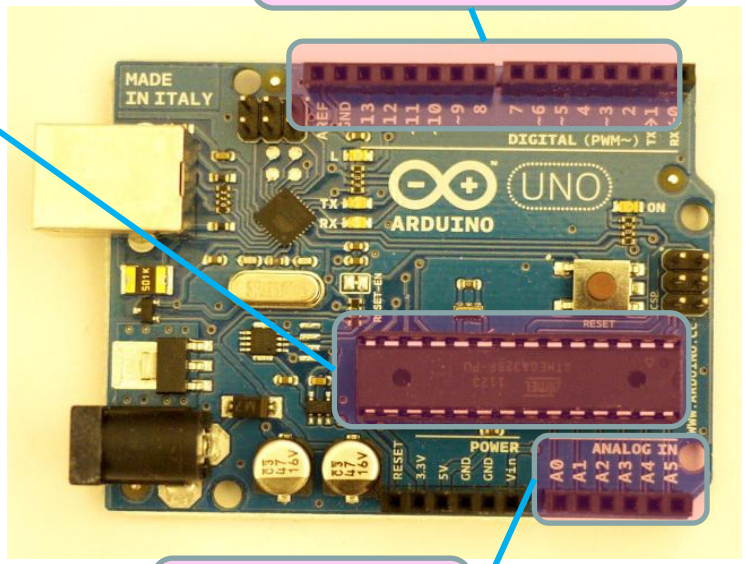
1 Un circuit intégré

Entrées/sorties numériques

Micro contrôleur



Entrées analogiques



Entrées / Sorties:

microcontrôleur

dialogue

extérieur

Entrées → capteurs :

Collecte les informations sur leur environnement (T°, bouton poussoir, capteur de mouvement, etc…)

Sorties → actionneurs:

Action sur le monde extérieur (moteur, lampe, etc…)

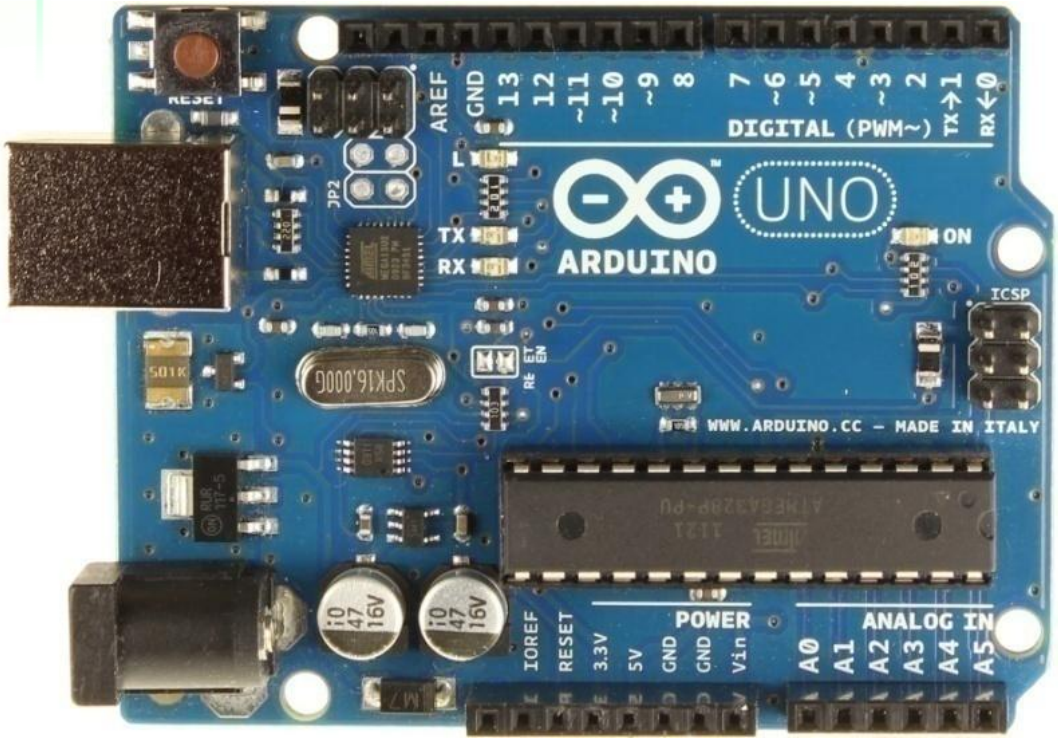
1 Un environnement de programmation

IDE : *Integrated Development Environment*

- Langage C
- **Ecrire/modifier un programme**
- **Convertir le programme en « langage machine » compréhensible par la carte**

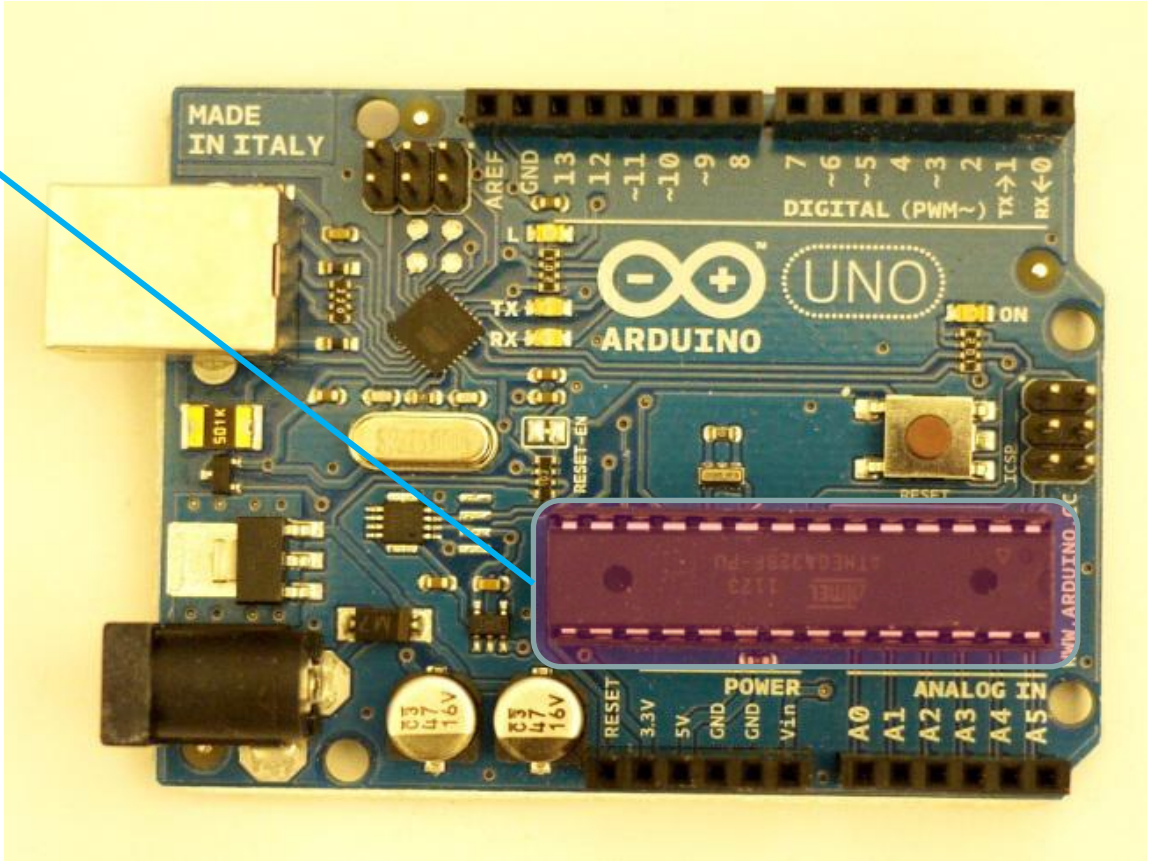
2

Description détaillée de la carte Arduino



2 Le microcontrôleur

Micro contrôleur



- Cerveau de la carte
- Stoke le programme en mémoire (Eprom)
- Exécute le programme

- **ATmega328 cadencé à 16MHz**
  - ⚠ Ordinateur standard : 1GHz minimum
- **Espace de stockage du programme: ≈32ko**
  - ⚠ Clé USB: 32Go !!

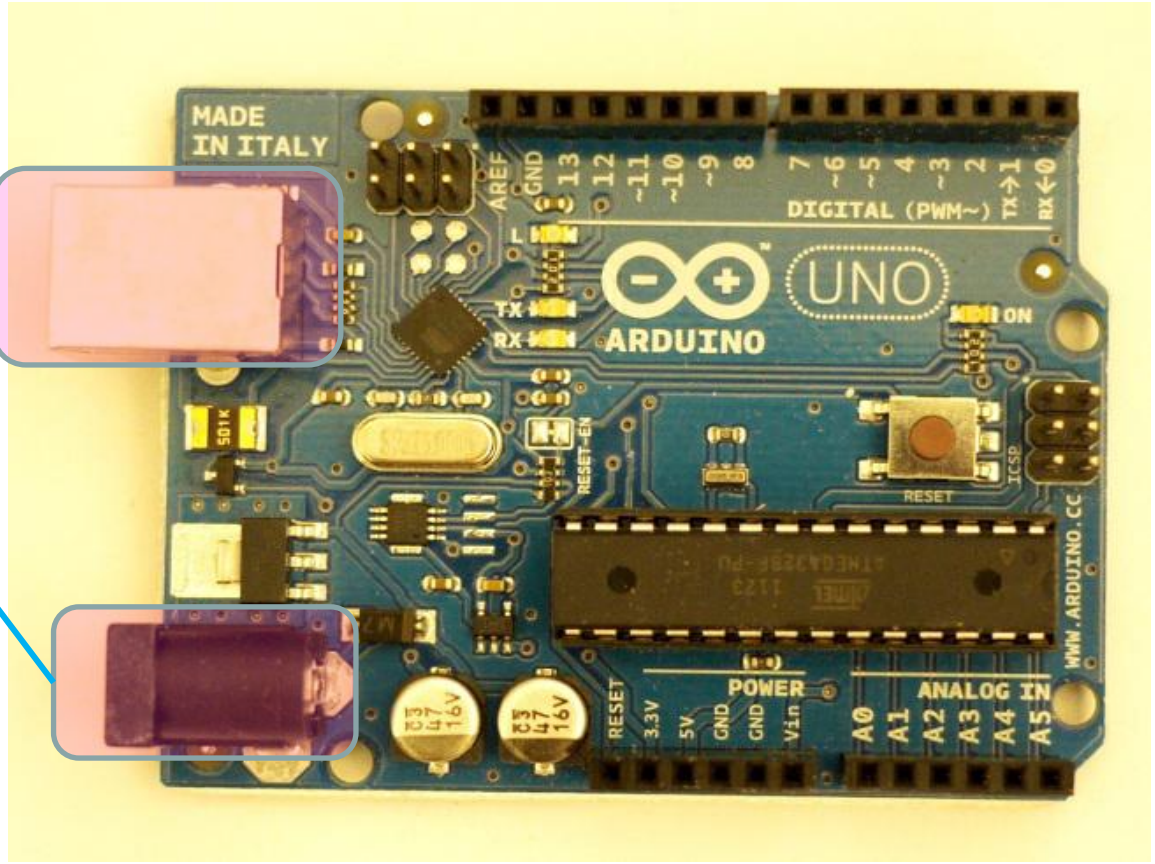
2 L’alimentation

Alimentation via USB en 5V

Alimentation externe [7V, 12V]

- Pile 9V
- **Régulateur de tention réduit la tension à 5V**

Bon fonctionnement de la carte



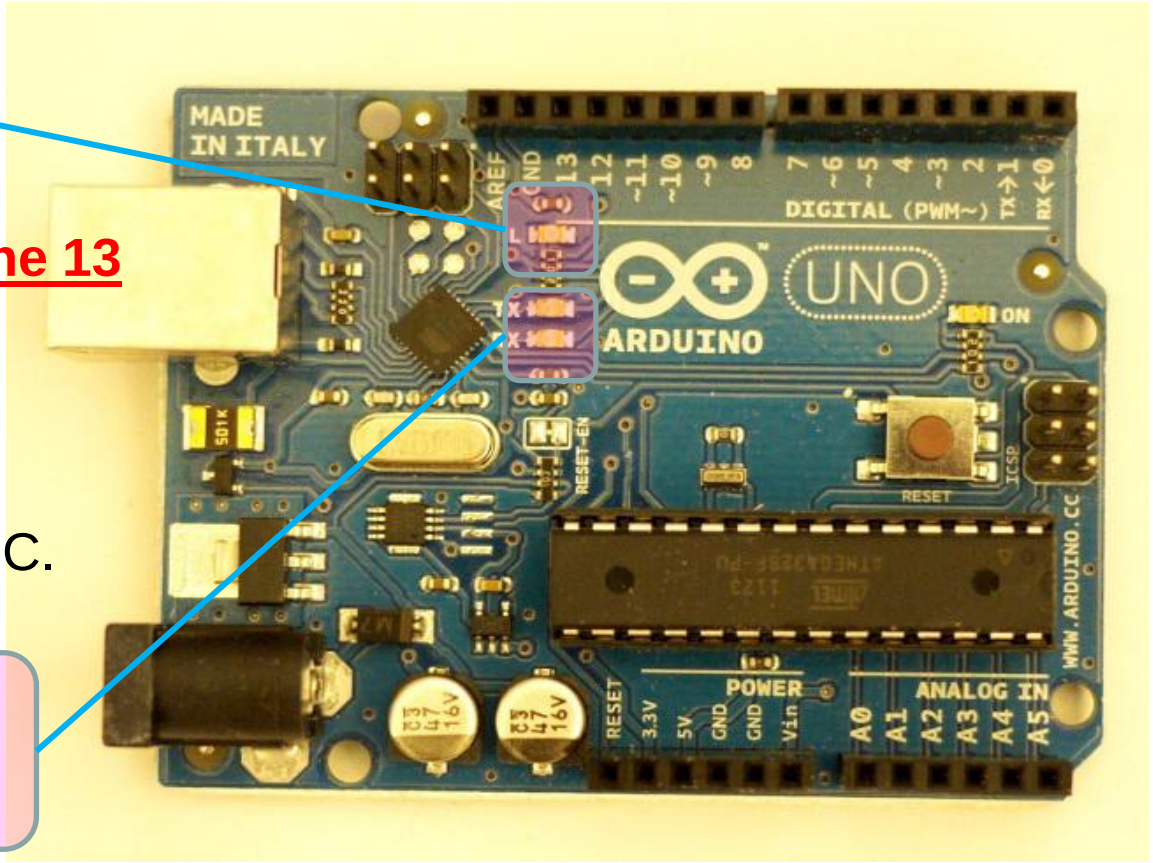
Respecter l'intervalle de 7V à 15V (même si le régulateur peut supporter plus)

2 Visualisation par LED

1 LED de test du matériel

- Connectée à la **broche 13** du microcontrôleur.
- Clignote quelques secondes quand on branche la carte au PC.

2 LED de de visualisation de l'activité du port série



- 1 LED pour l'émission (TX)
- 1 LED pour la réception (RX)

→ Clignotement des LED lors du téléchargement du programme…

2 La connectique

5V →1
0V → 0

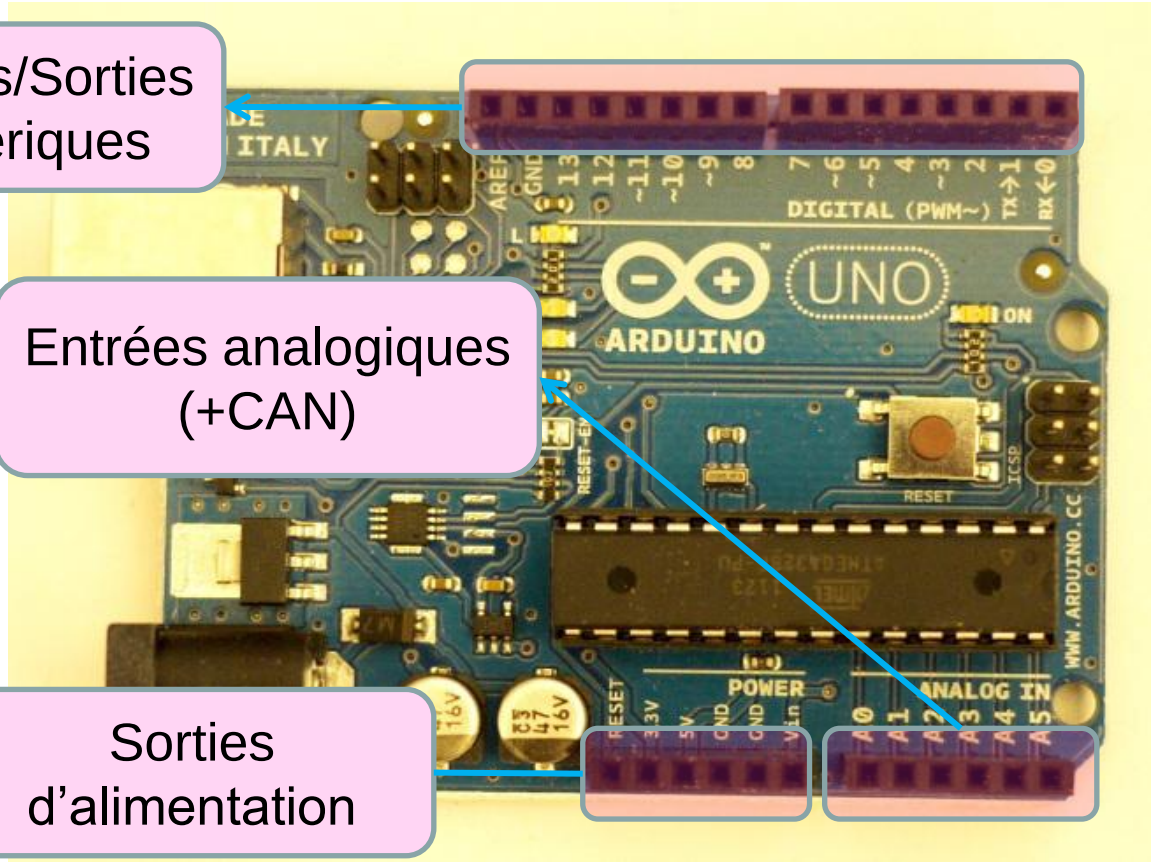
Entrées/Sorties numériques

10 bits (0 à 1023)
0V → 0
5V → 1023

Entrées analogiques (+CAN)

5V, 3,3V, masse (gnd)

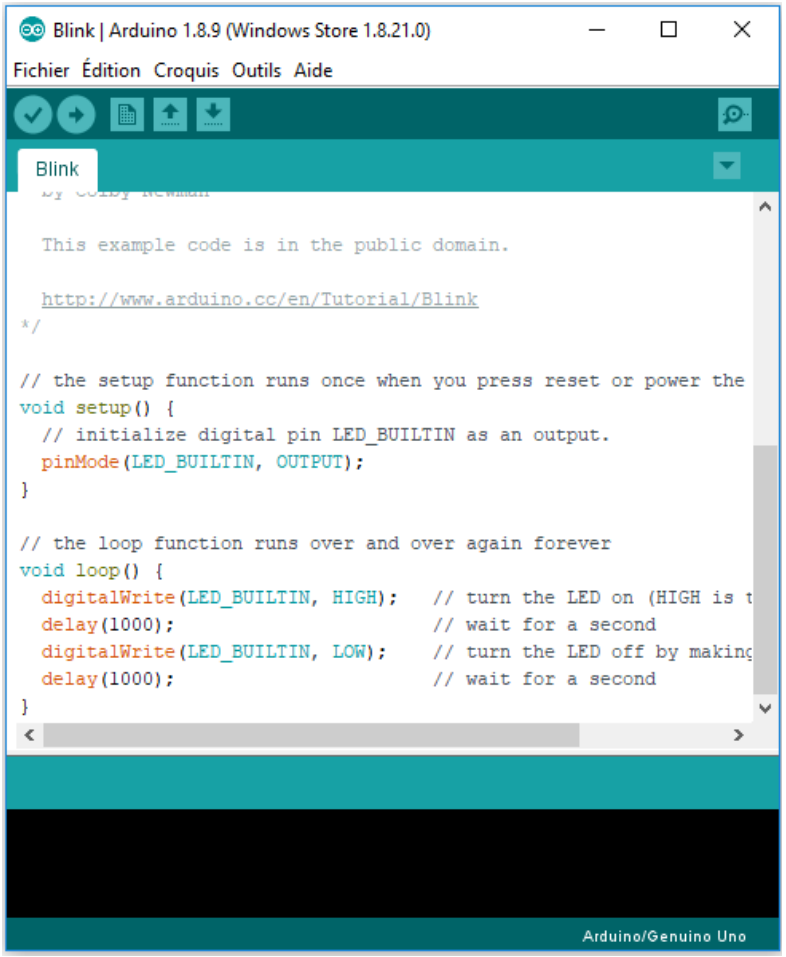
Sorties d'alimentation



- **Ordre du "brochage" très important**
- **Possibilité de brancher tous types de montages et modules**

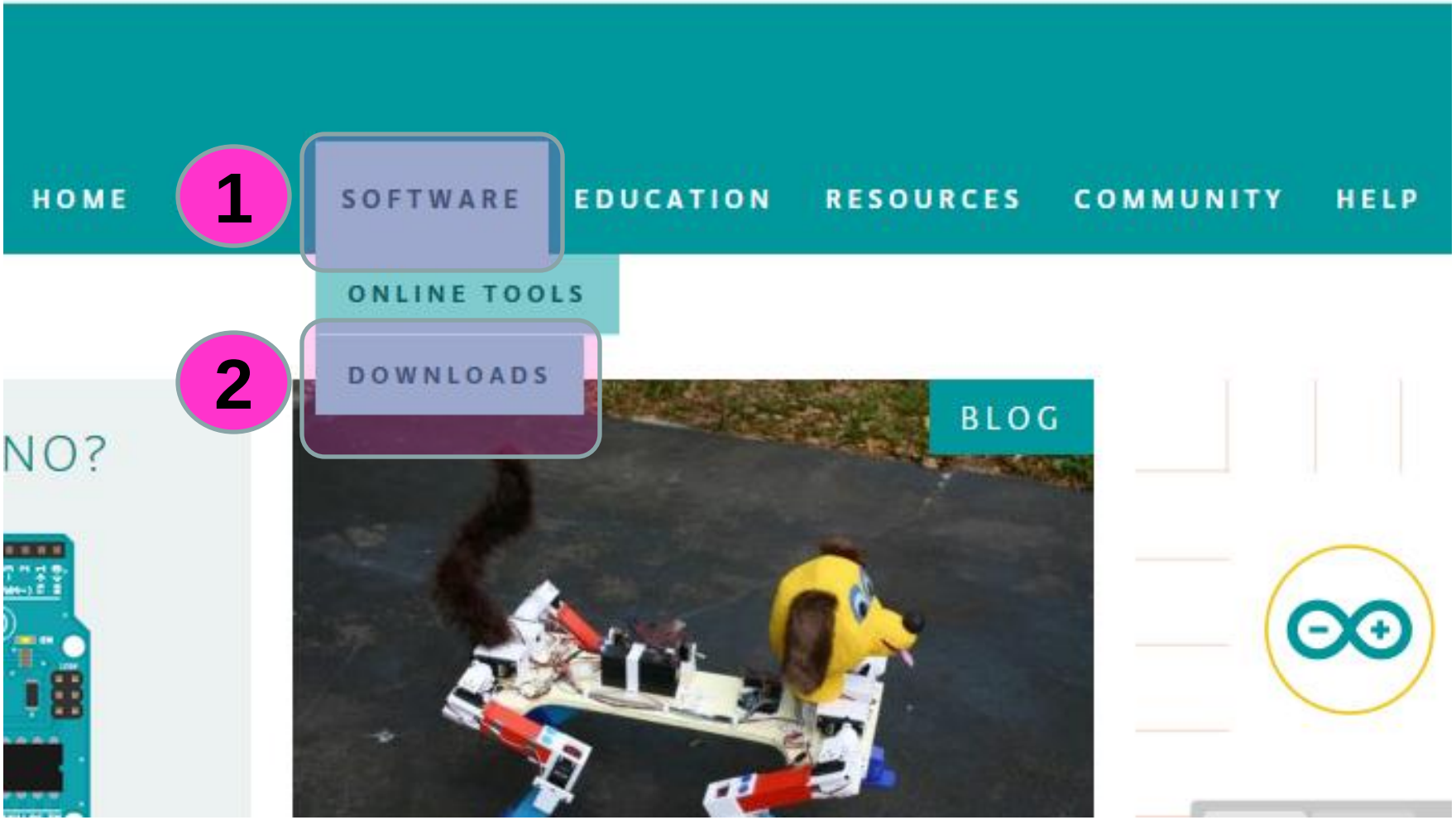
3

Installation de l'IDE
Test de la carte

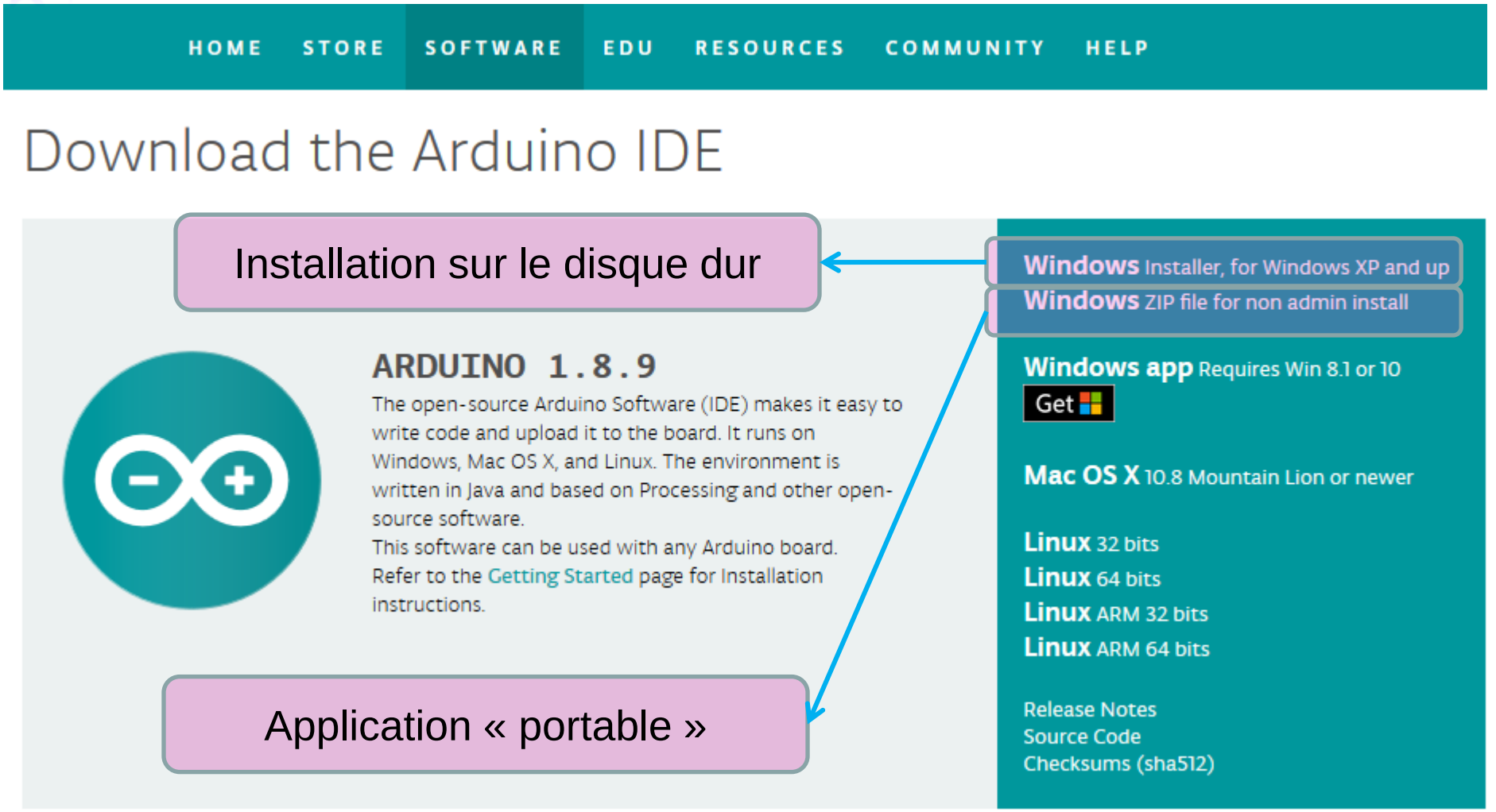


3 Téléchargement de l'IDE

https://www.arduino.cc/



3 Téléchargement de l'IDE



Choisir l'application adapté à son système d'exploitation

3 Téléchargement de l'IDE

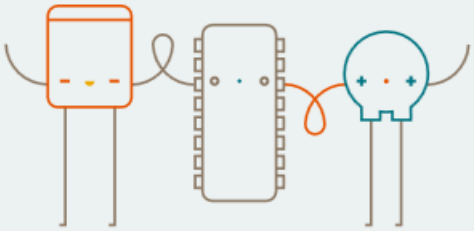
ARDUINO

SIGN IN

HOME STORE SOFTWARE EDUCATION RESOURCES COMMUNITY HELP

Contribute to the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.



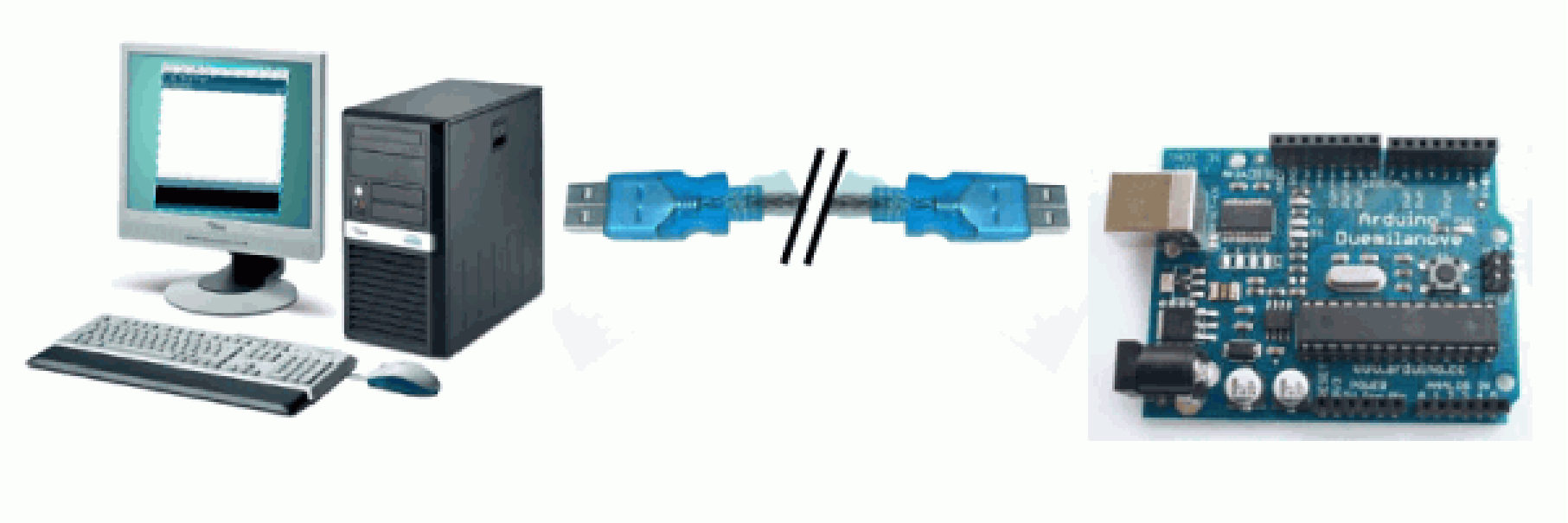
SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 31,253,036 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

$3 $5 $10 $25 $50 OTHER

JUST DOWNLOAD CONTRIBUTE & DOWNLOAD

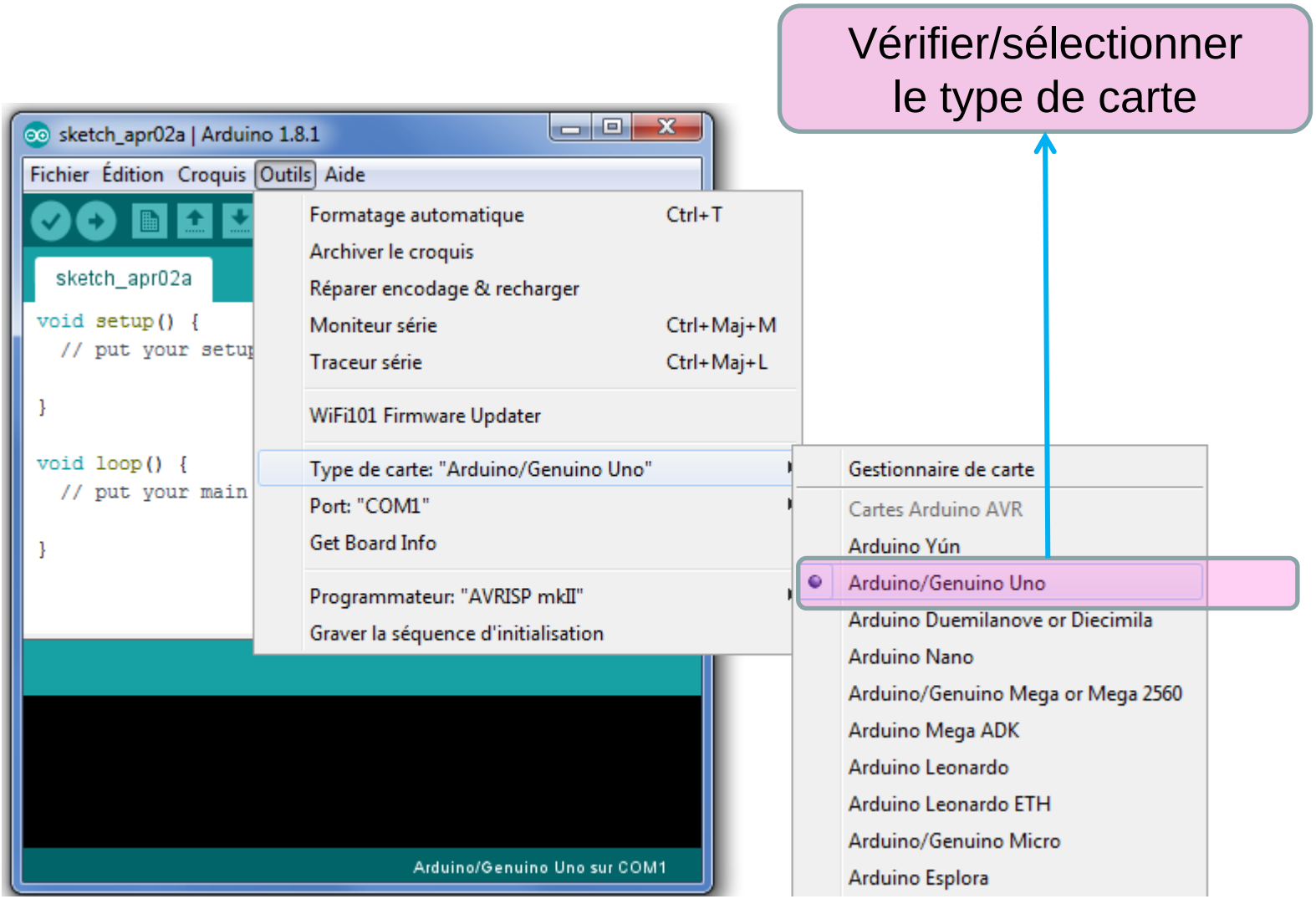
Pour une utilisation sans contribution

3 Brancher la carte

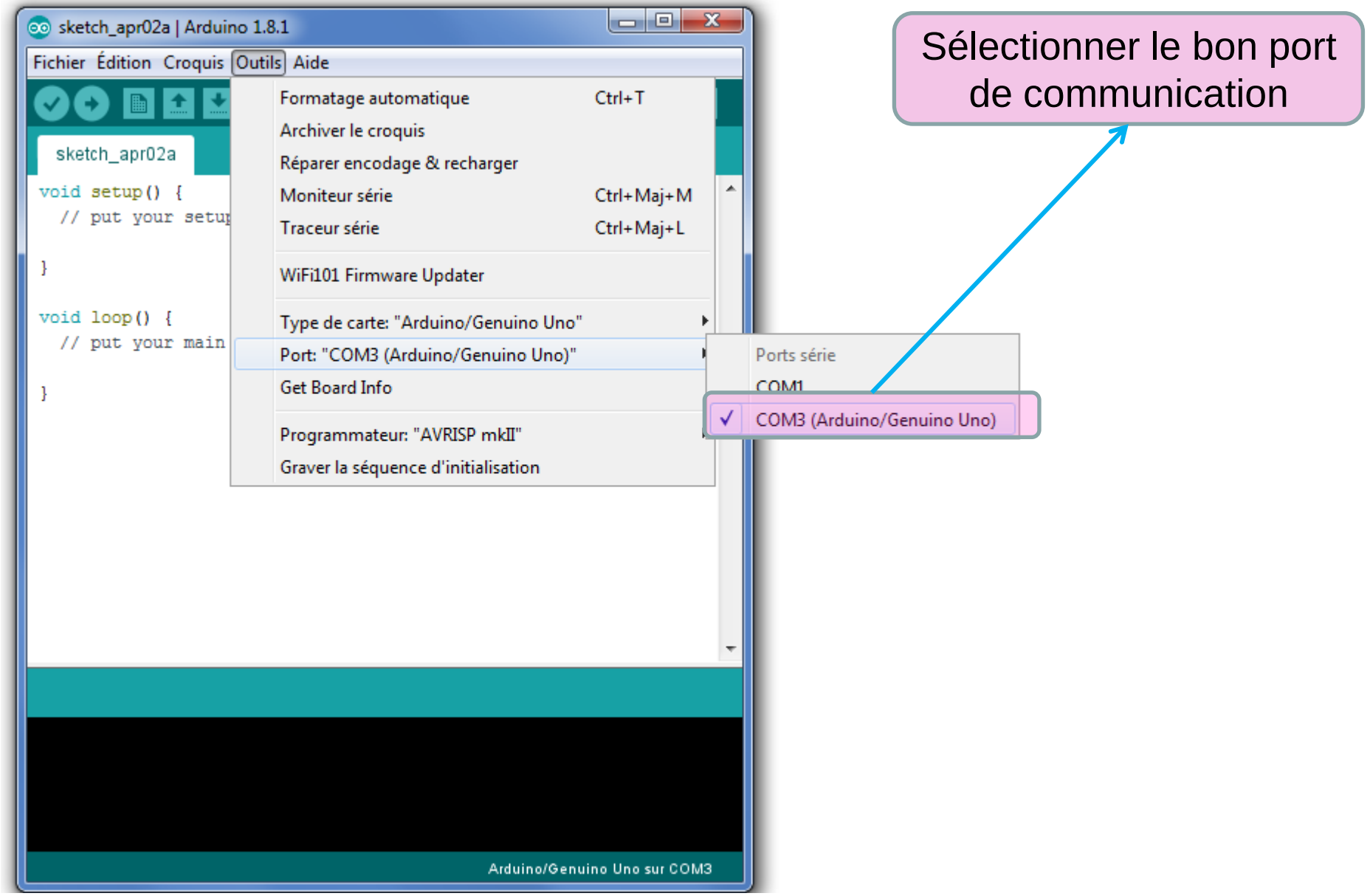


- Vérifier le type de carte utilisé
- Vérifier le port de communication

3 Brancher la carte

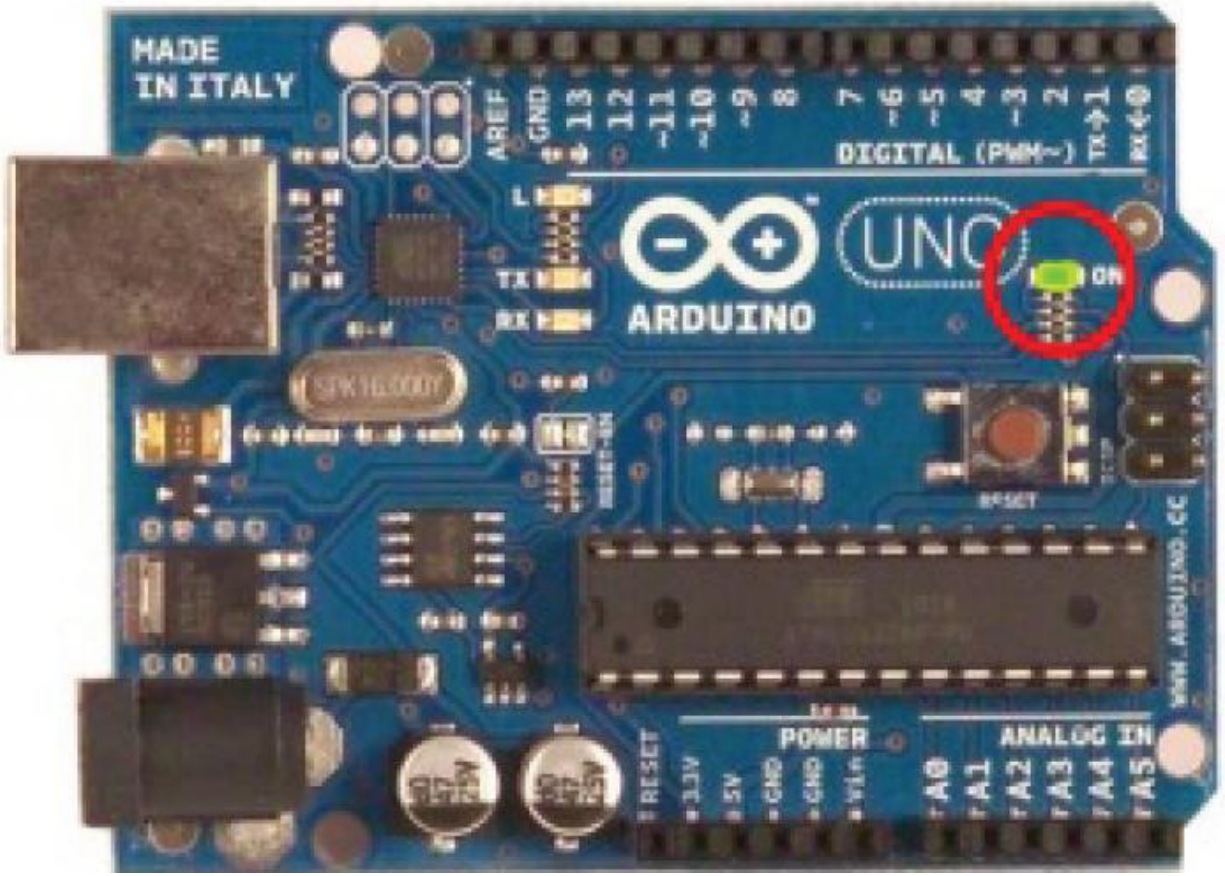


3 Brancher la carte

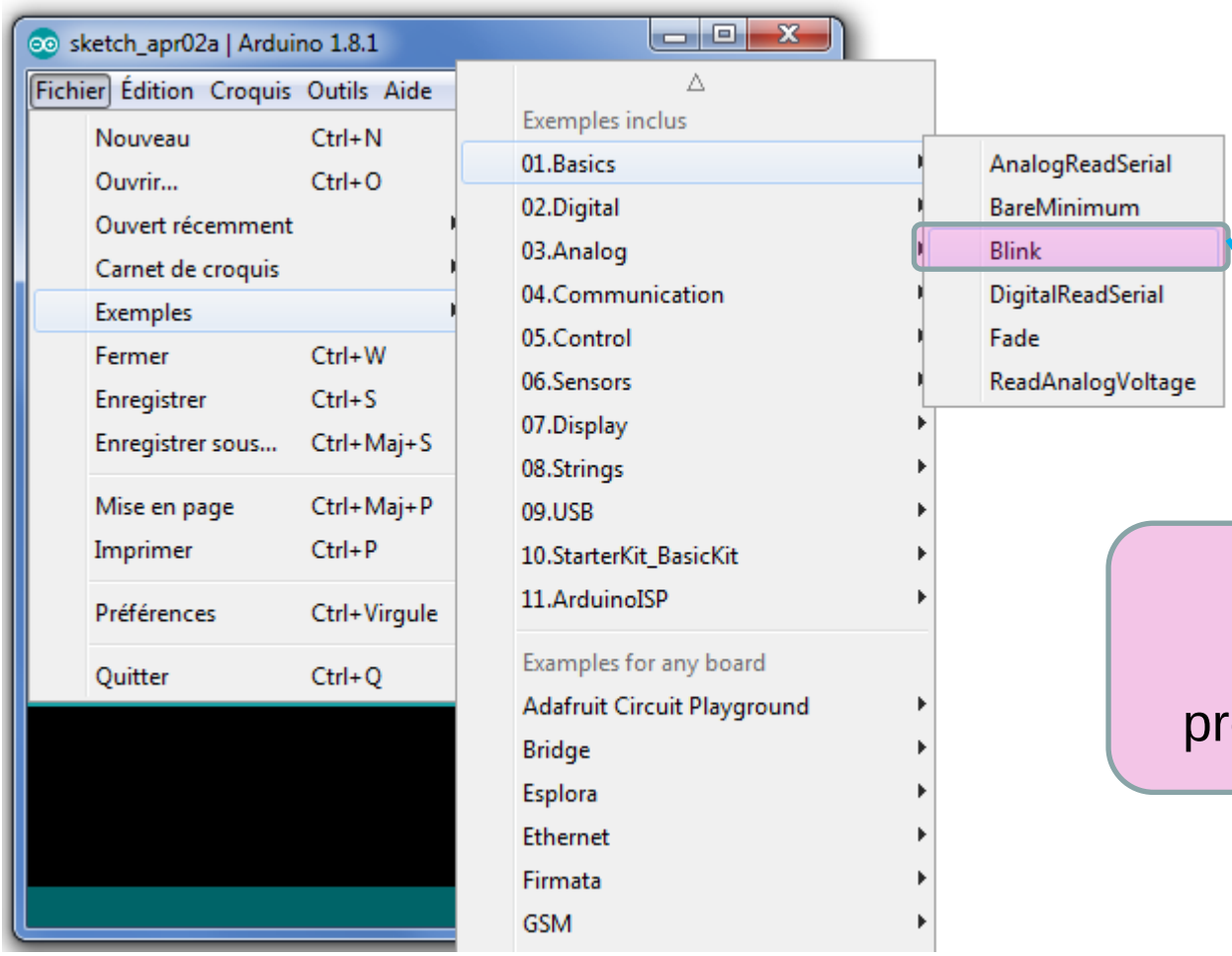


3 Carte correctement reconnue

- **Clignotement des micro-LED de la carte au branchement**
- **Micro-LED verte allumée : bonne alimentation de la carte**

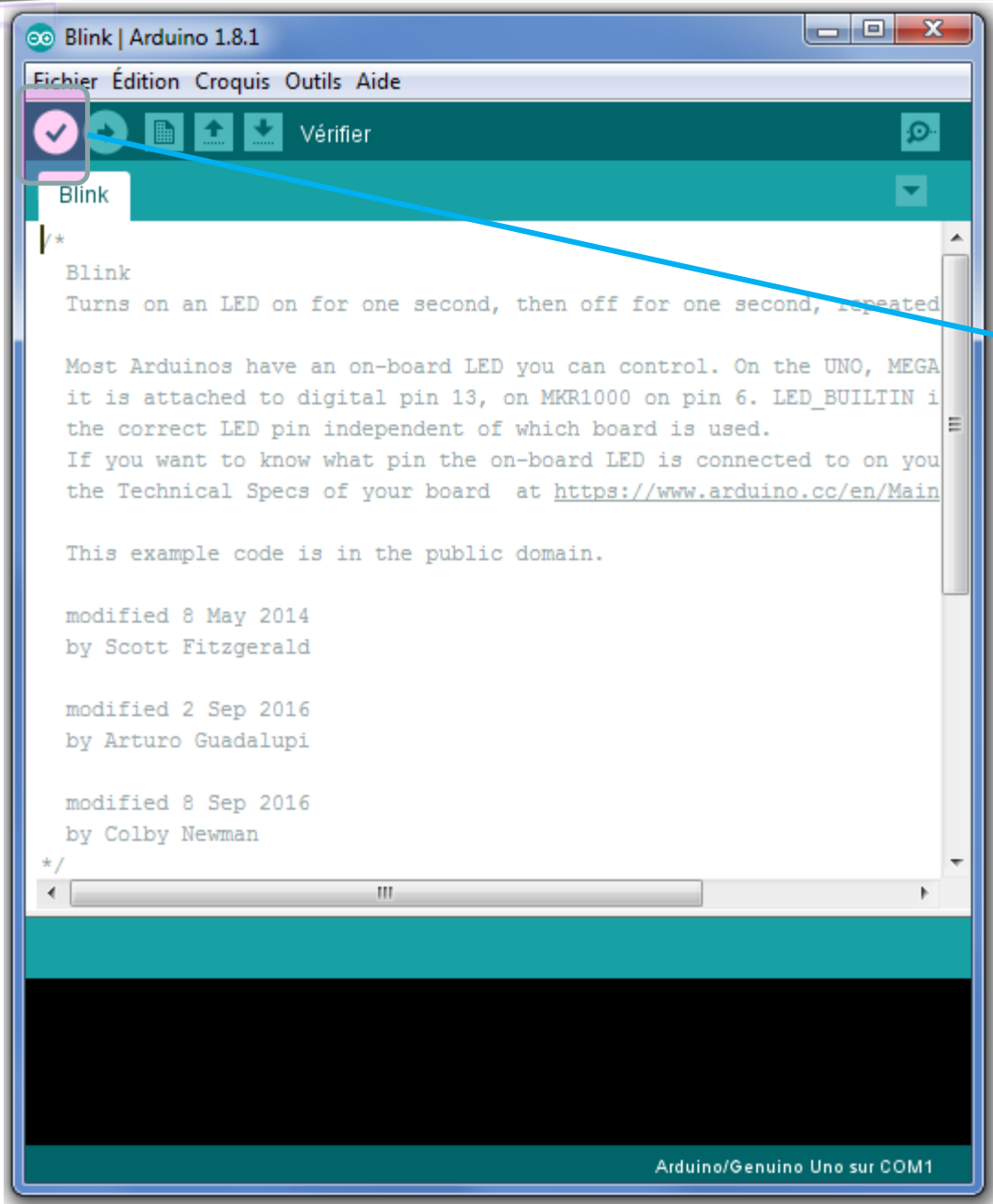


3 Tester la carte : « Blink »



Choisir dans les exemples le programme « blink »

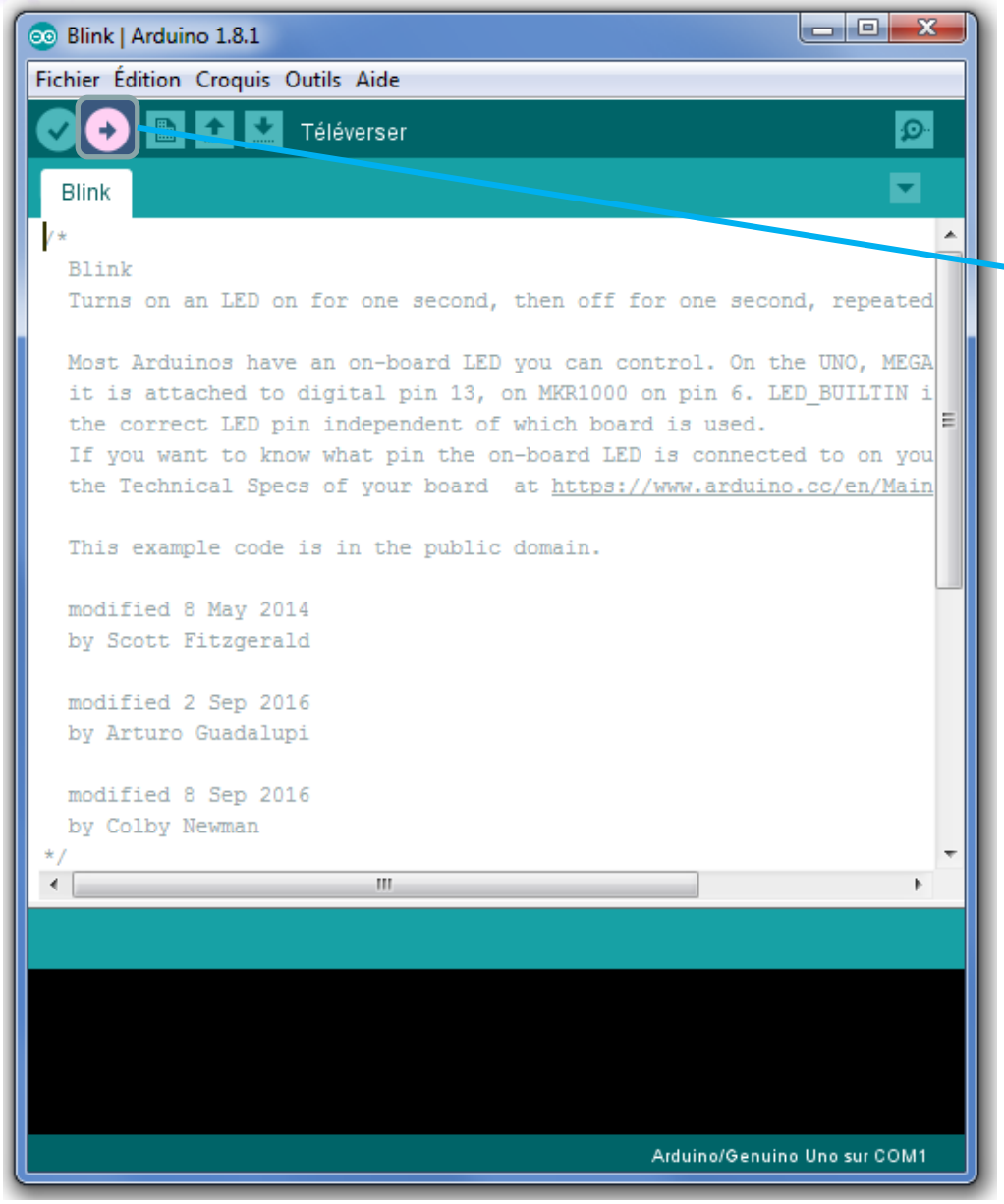
3 Tester la carte : « Blink »



Compiler le programme

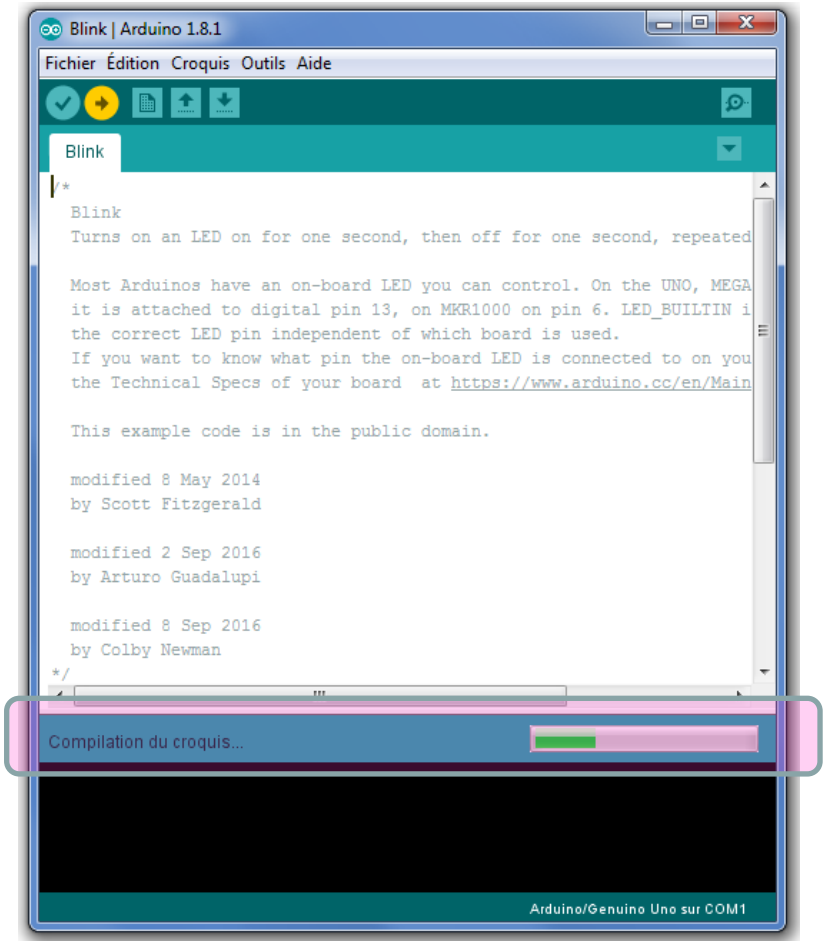
- Vérifie et indique les erreurs
- Traduit le code en « langage machine »

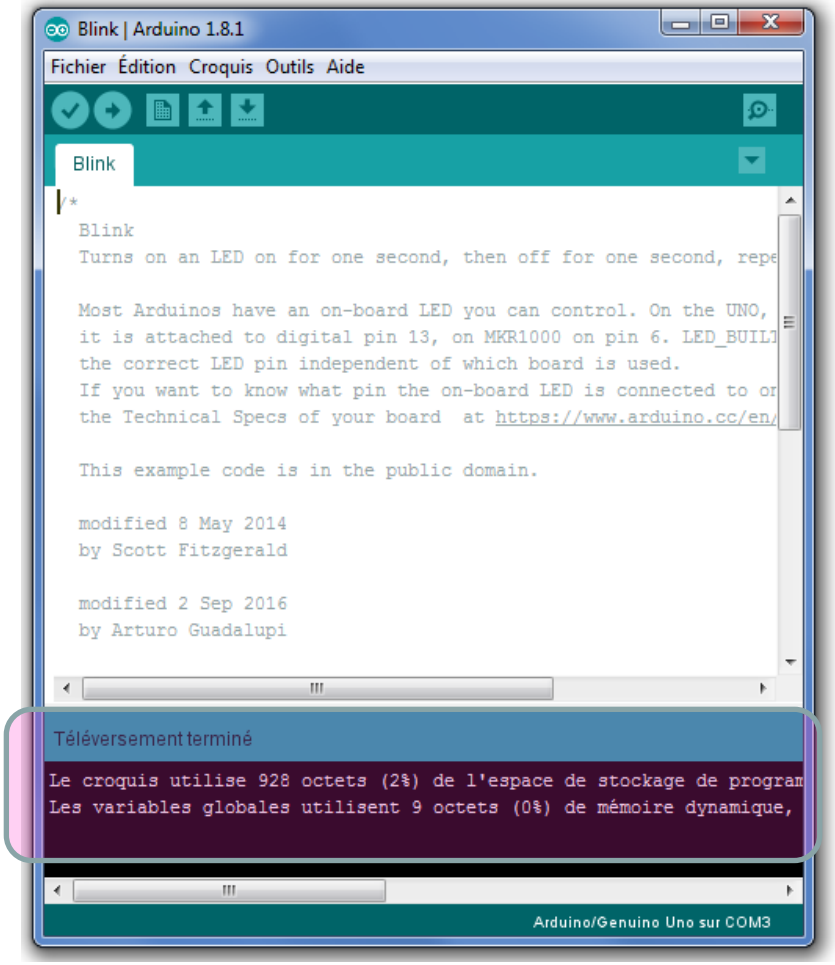
3 Tester la carte : « Blink »



Téléverser le programme sur la carte

3 Tester la carte : « Blink »



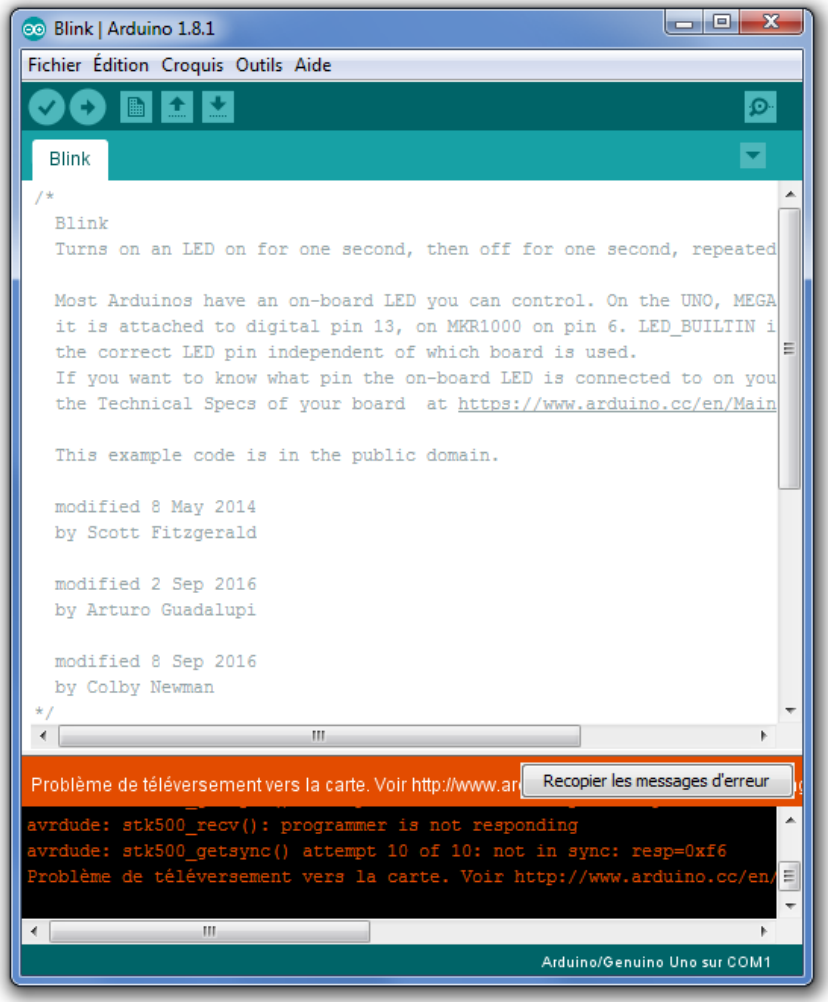


1

2

3 Tester la carte : « Blink »

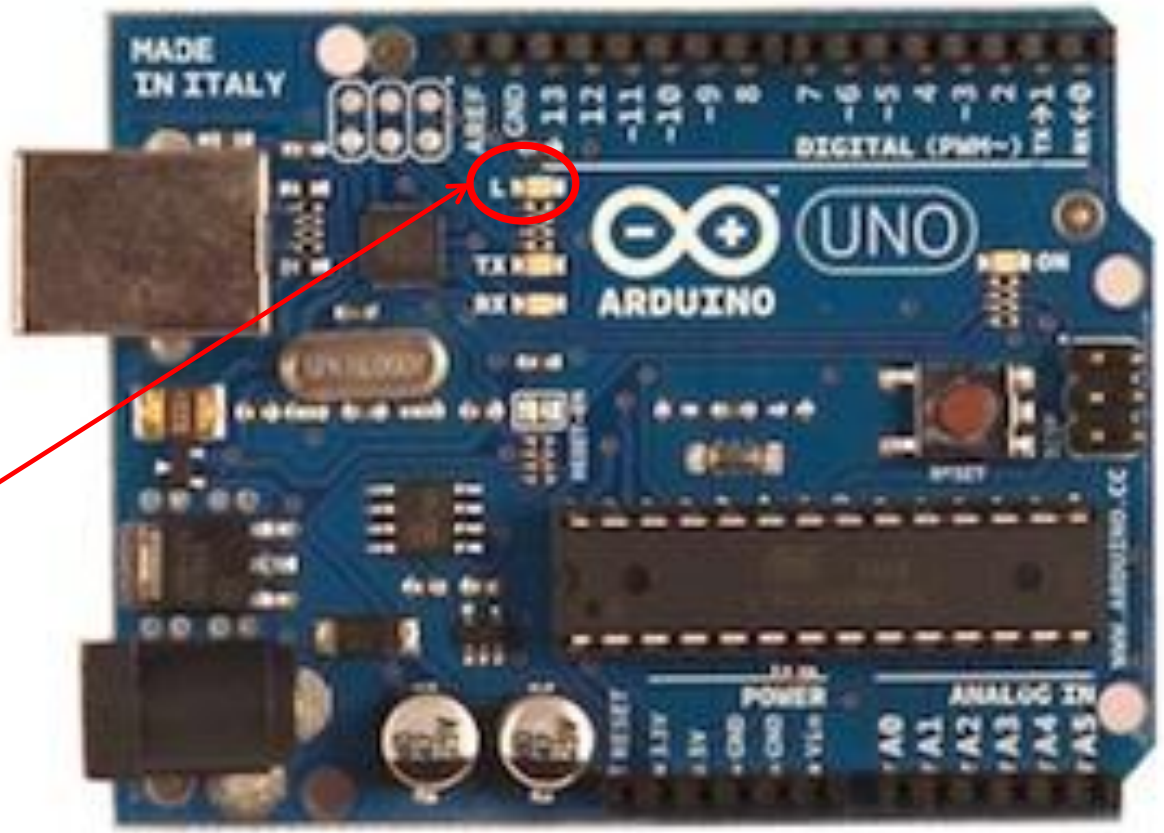
Si message d'erreur



- **Vérifier le type de carte utilisé**
- **Vérifier le port de communication**

3 Tester la carte : « Blink »

Bon fonctionnement !!



Clignotement de la LED à f = 1Hz

3 Commentaire du programme « Blink »

Zones de commentaires

Non exécutées à la compilation du programme

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode( LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite( LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite( LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Structure du programme ⇨ Deux grandes fonctions

void setup() {}

Instruction d'initialisation des ressources

- **Définition des entrées/sorties**
- **Vitesse du port série, …**

⚠ **Une seule exécution**

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode( LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite( LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                        // wait for a second
  digitalWrite( LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                        // wait for a second
}
```

Structure du programme ⇨ Deux grandes fonctions

void loop() {
Instruction répétées une infinité de fois tant que la carte est sous tension

Boucle infinie

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode( LED_BUILTIN, OUTPUT);
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite( LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                        // wait for a second
  digitalWrite( LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                        // wait for a second
}
```

Les instructions

pinMode():
Définit la broche N°13 de la carte comme une sortie numérique
(1 seule exécution)

digitalWrite():
Applique un état haut de tension (5V) à la broche N°13
=> La LED s'allume !

delay():
Demande à la carte de patienter 1000ms = 1s en conservant l'état haut de la pate N°13
=> La LED allumée 1s

digitalWrite():
Applique un état bas de tension (0V) à la broche N°13
=> La LED s'éteint !

LED_BUILTIN = 13

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode( LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite( LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                        // wait for a second
  digitalWrite( LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                        // wait for a second
}
```

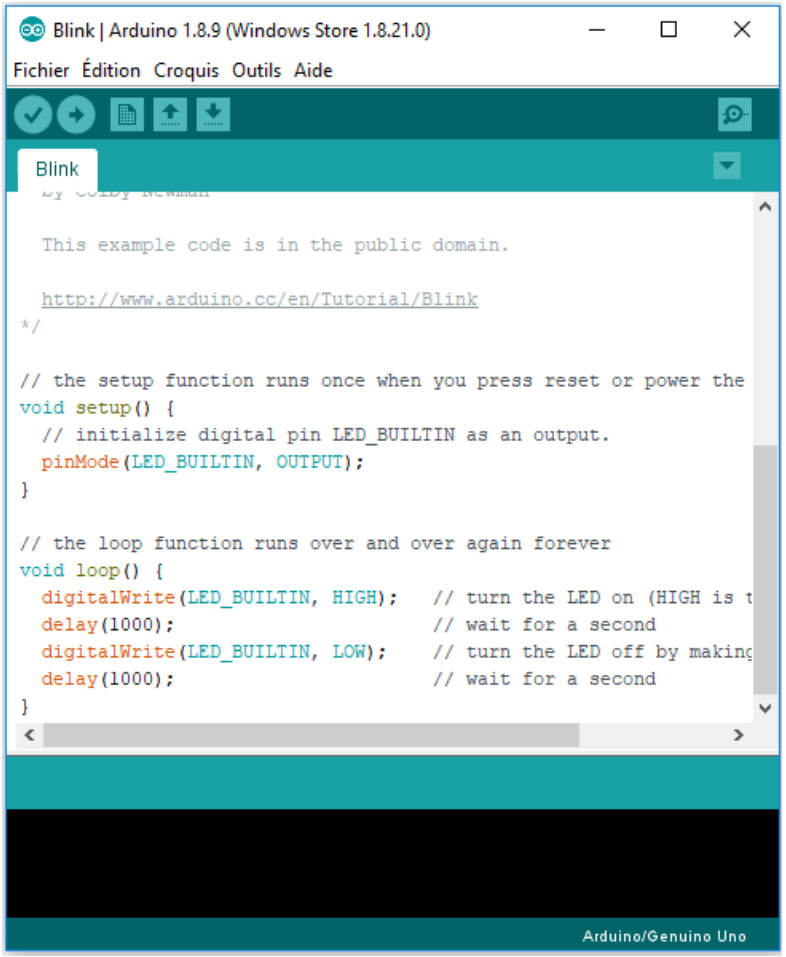
Début de la boucle

Fin de la boucle
=> retour au début de la boucle

delay():
Demande à la carte de patienter 1s en conservant l'état bas de la pate N°13
=> La LED éteinte 1s

4

Un premier montage

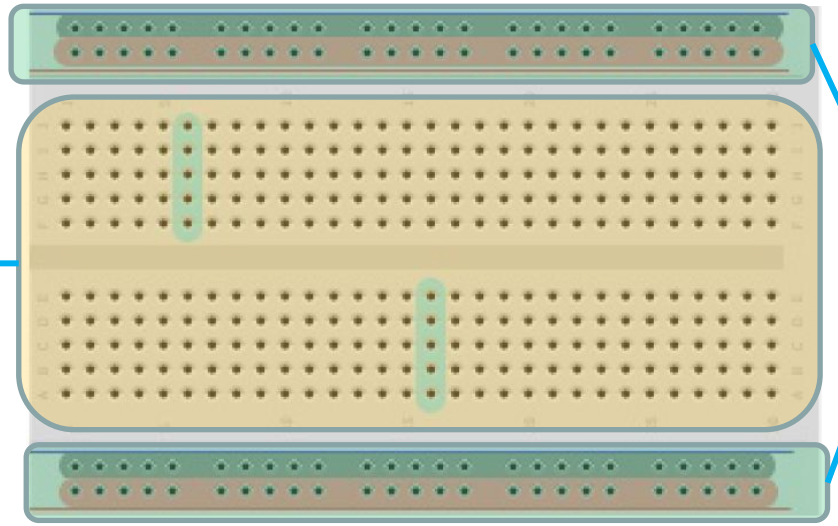


4 Objectif

Faire clignoter une LED connectée à la pate digitale 2 de la carte.

Utiliser une « breadbord » permettant des connections faciles:

Zone médiane:
Colonnes verticales interconnectées



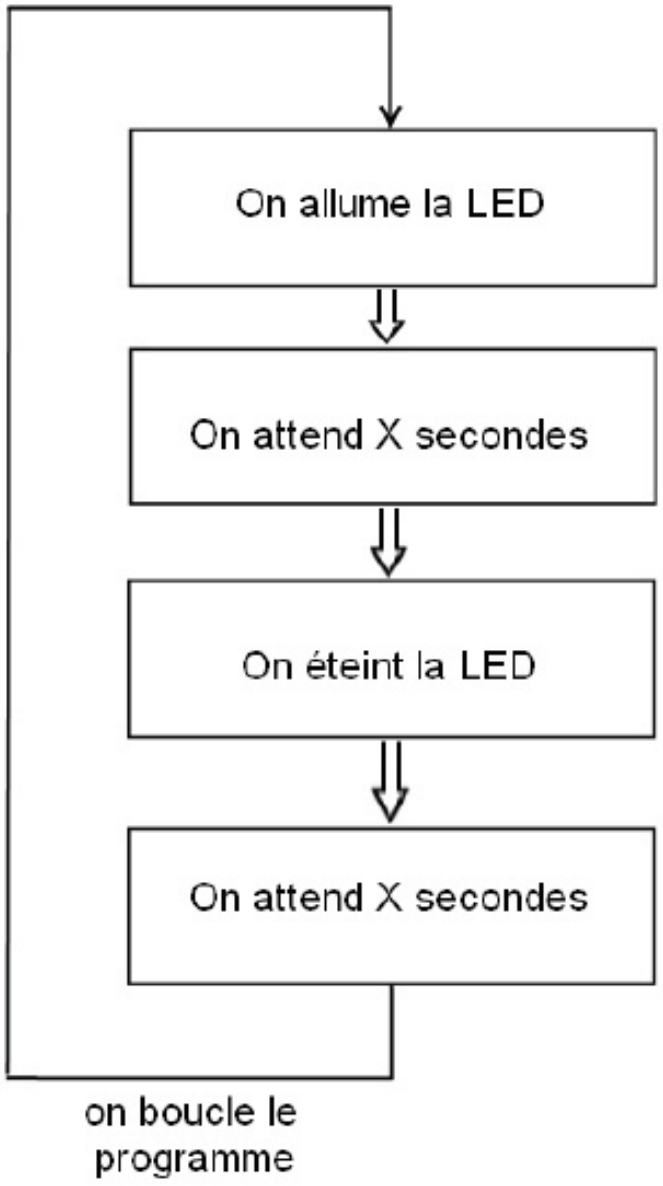
Alimentation:
Les trous d'une même ligne sont interconnectés

Brancher une résistance d'environ 300/400Ω pour protéger la LED

Faire attention au sens de parcours de la LED

Algorithme :

La LED est connectée à la pate digitale 2 de la carte.



Montage câblé:

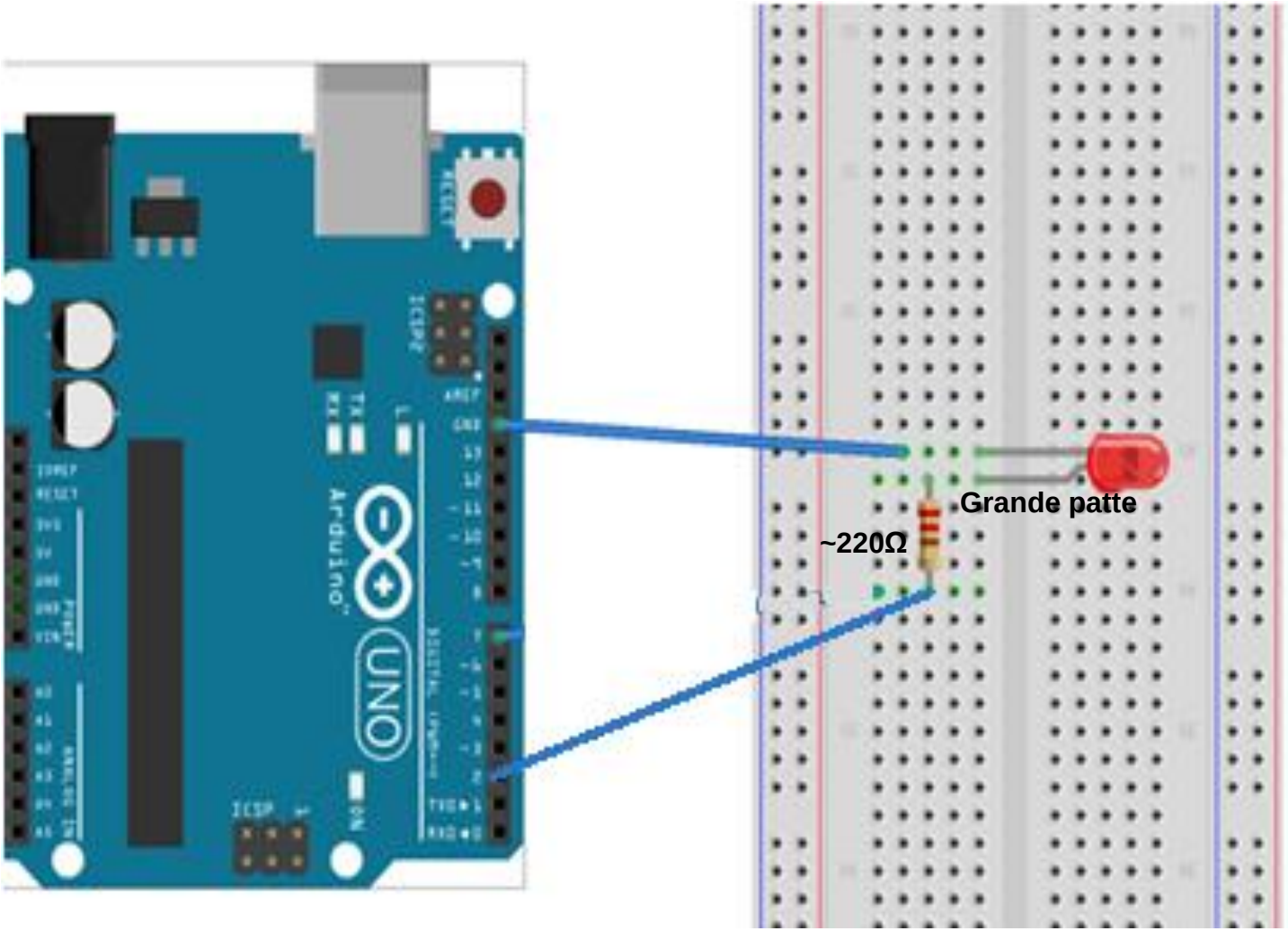
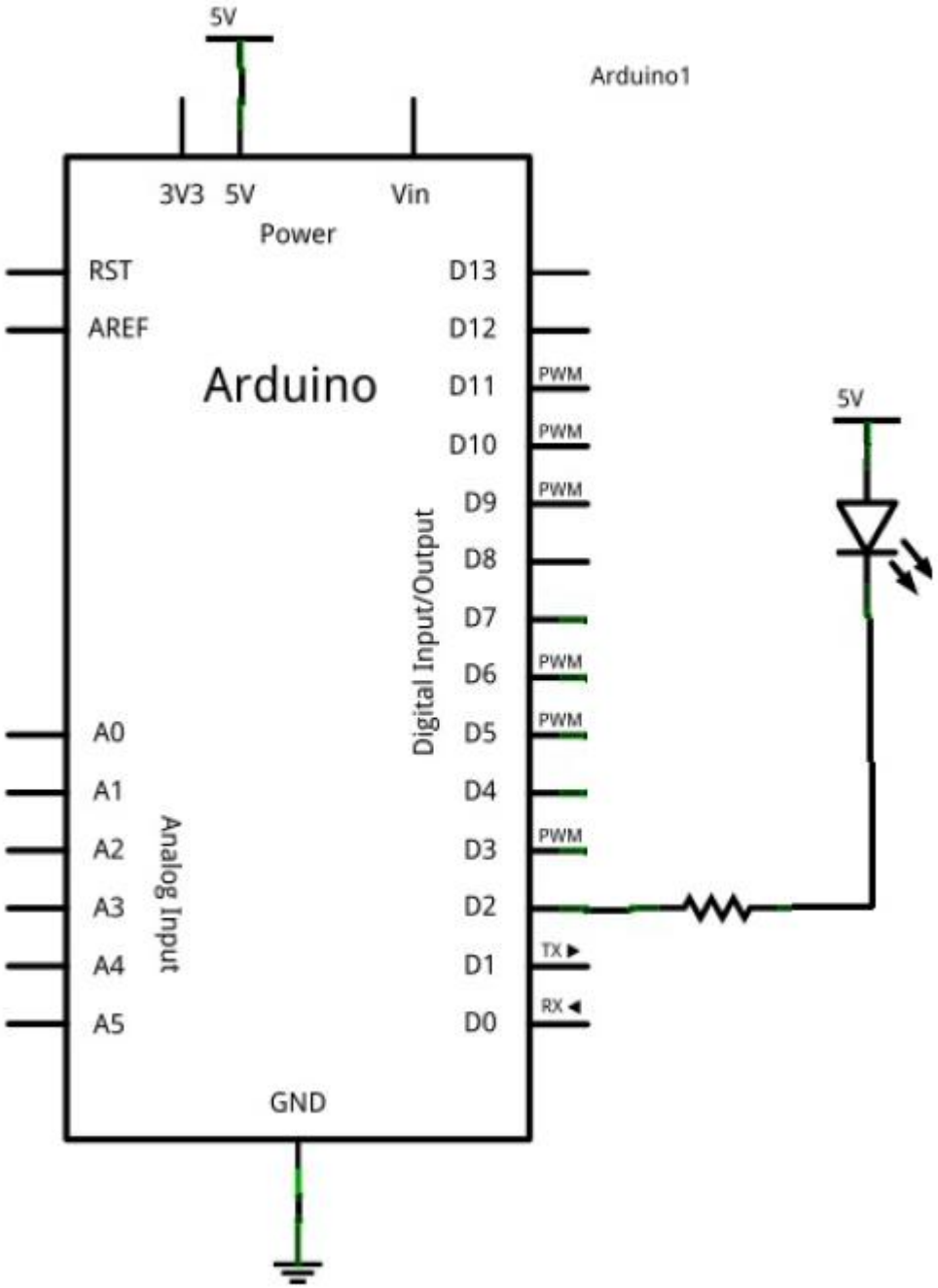
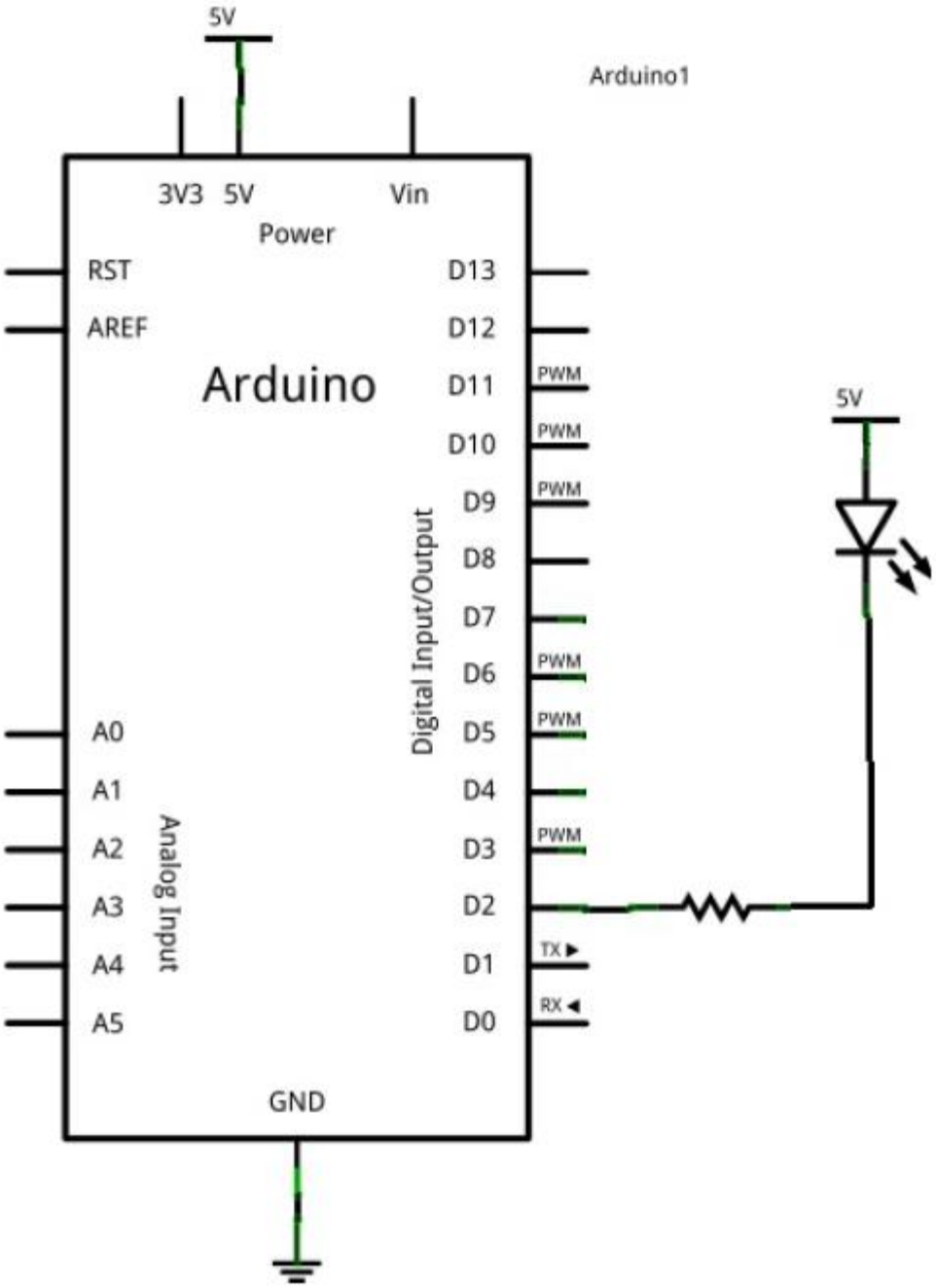


Schéma équivalent

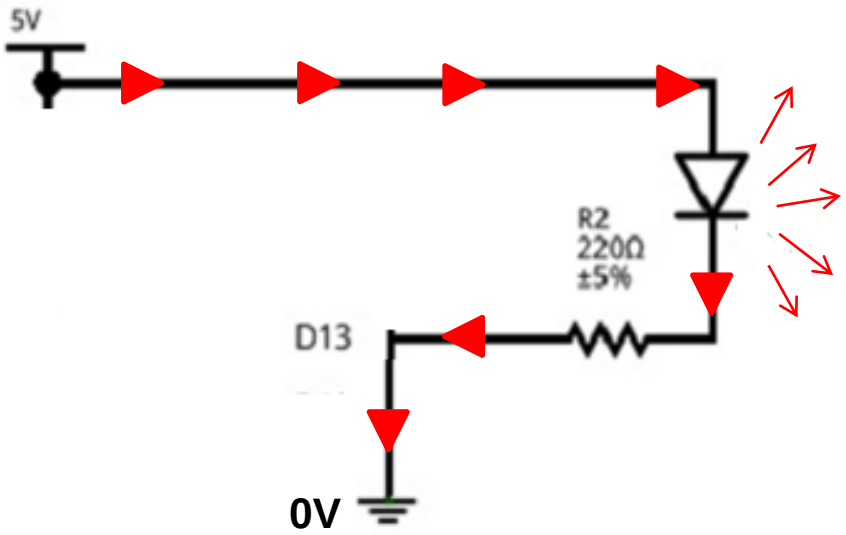


Question



Q: Quelle doit être la valeur de D2 pour que la LED s'allume ?

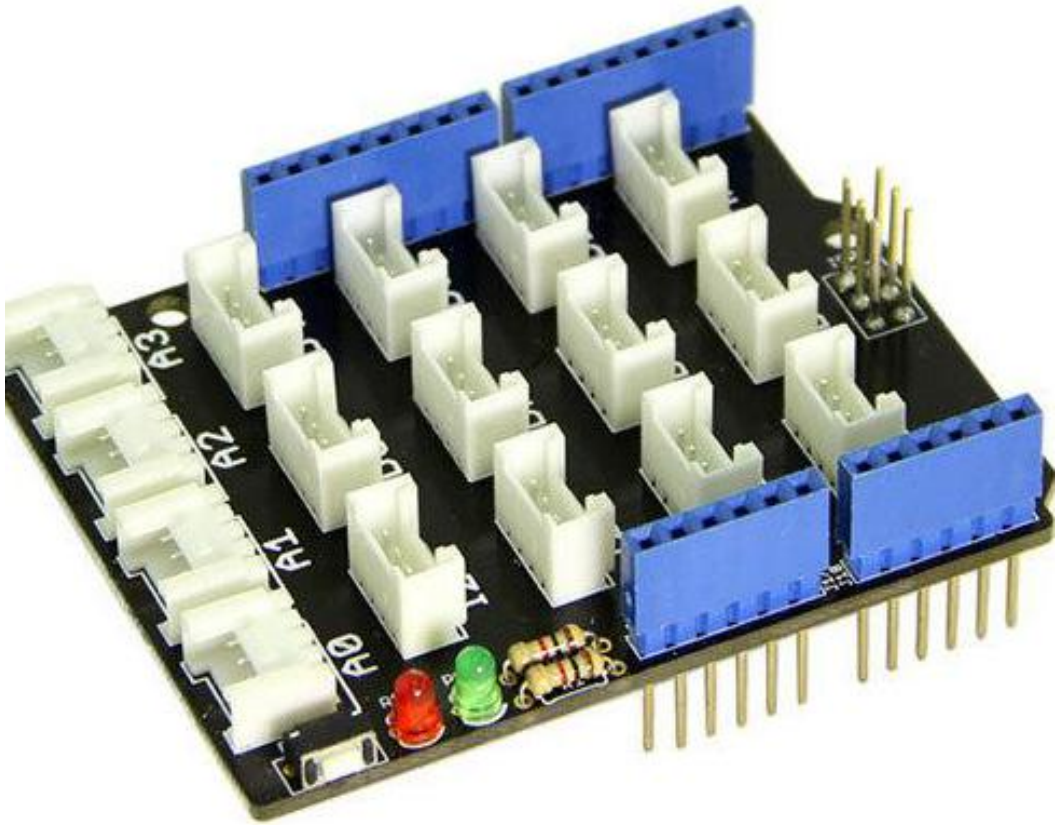
D2 → LOW



Programme:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

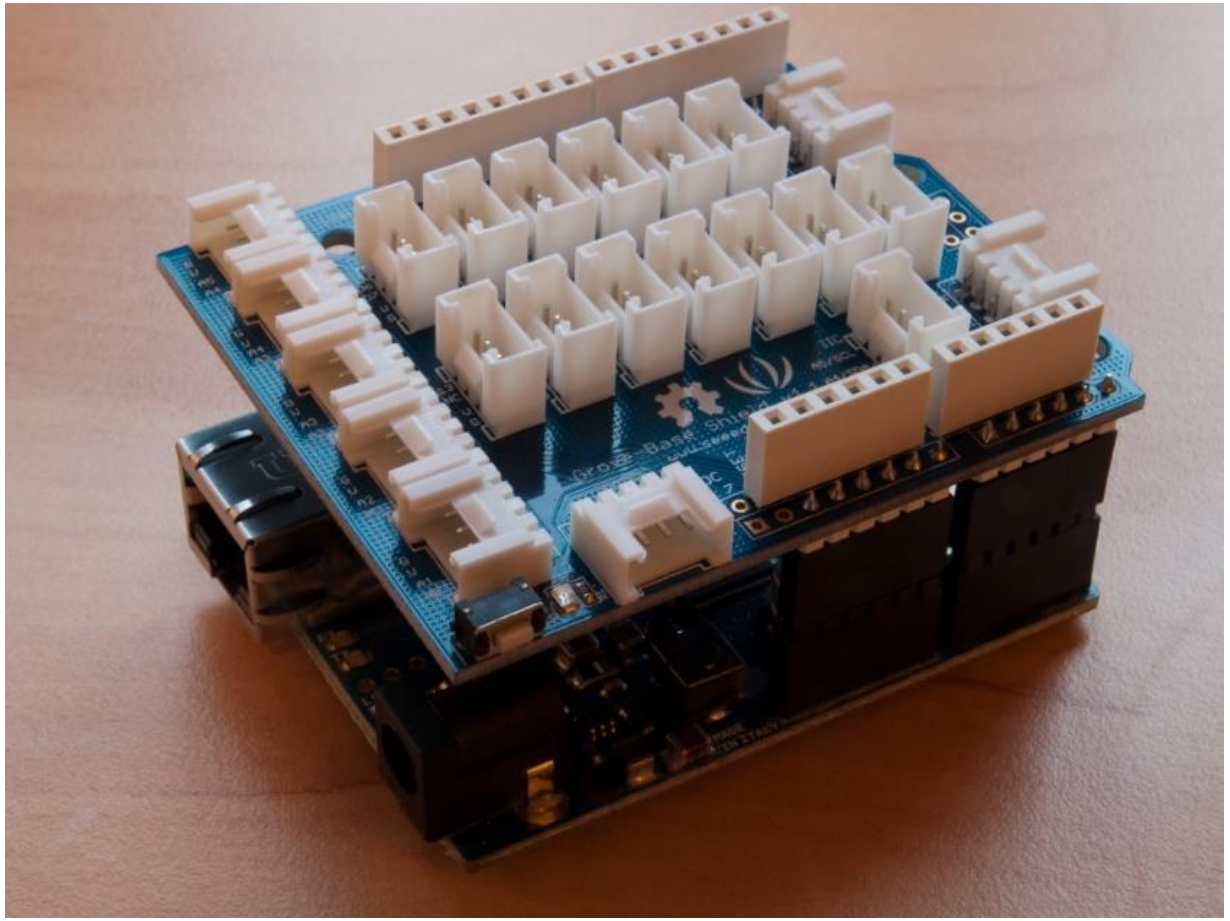
Montage Groove : Plus simple !!!



Utilisation d’un « shield » (bouclier)

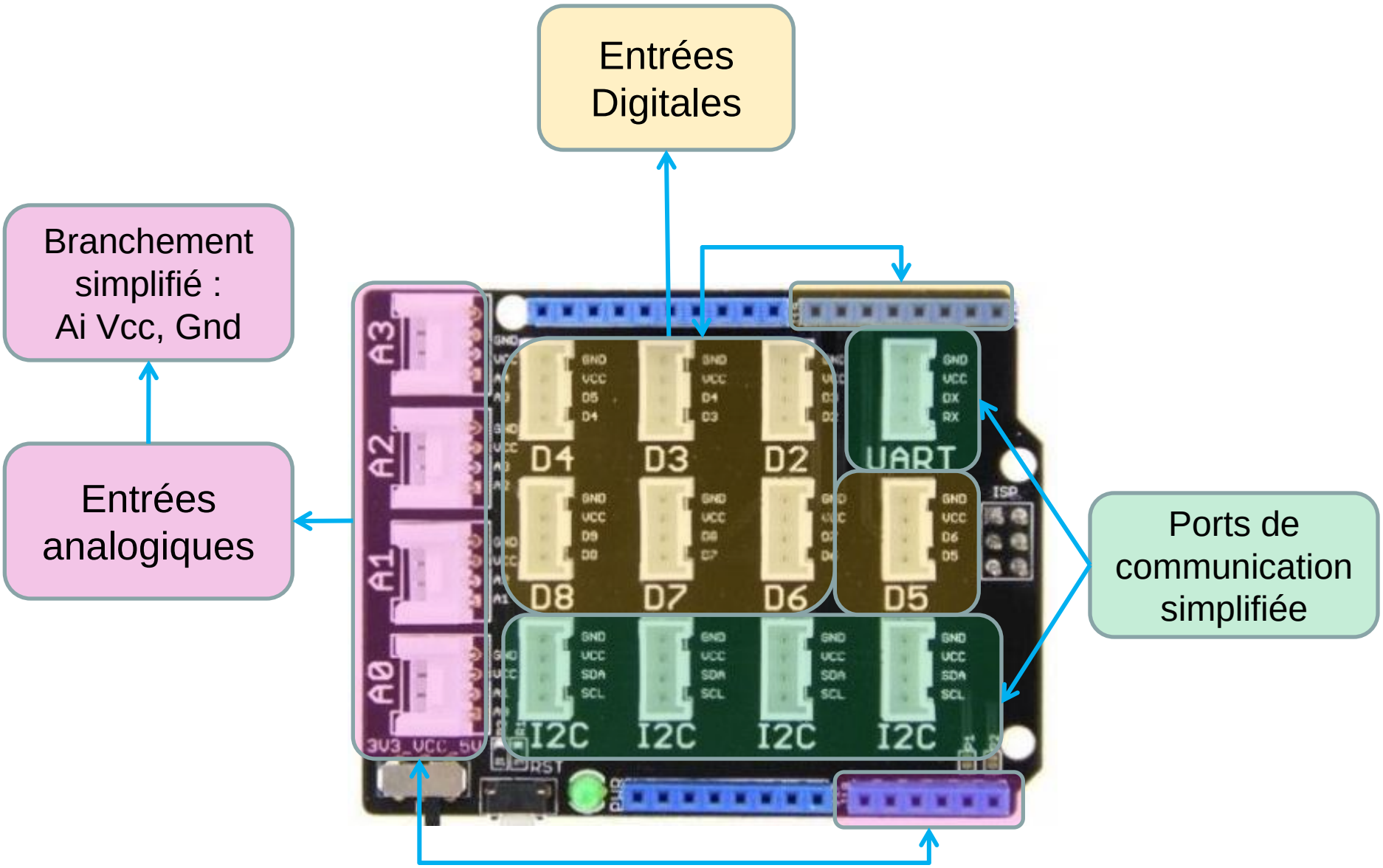
⇨ **Connections et montages plus simples…**

Montage Groove : Plus simple !!!

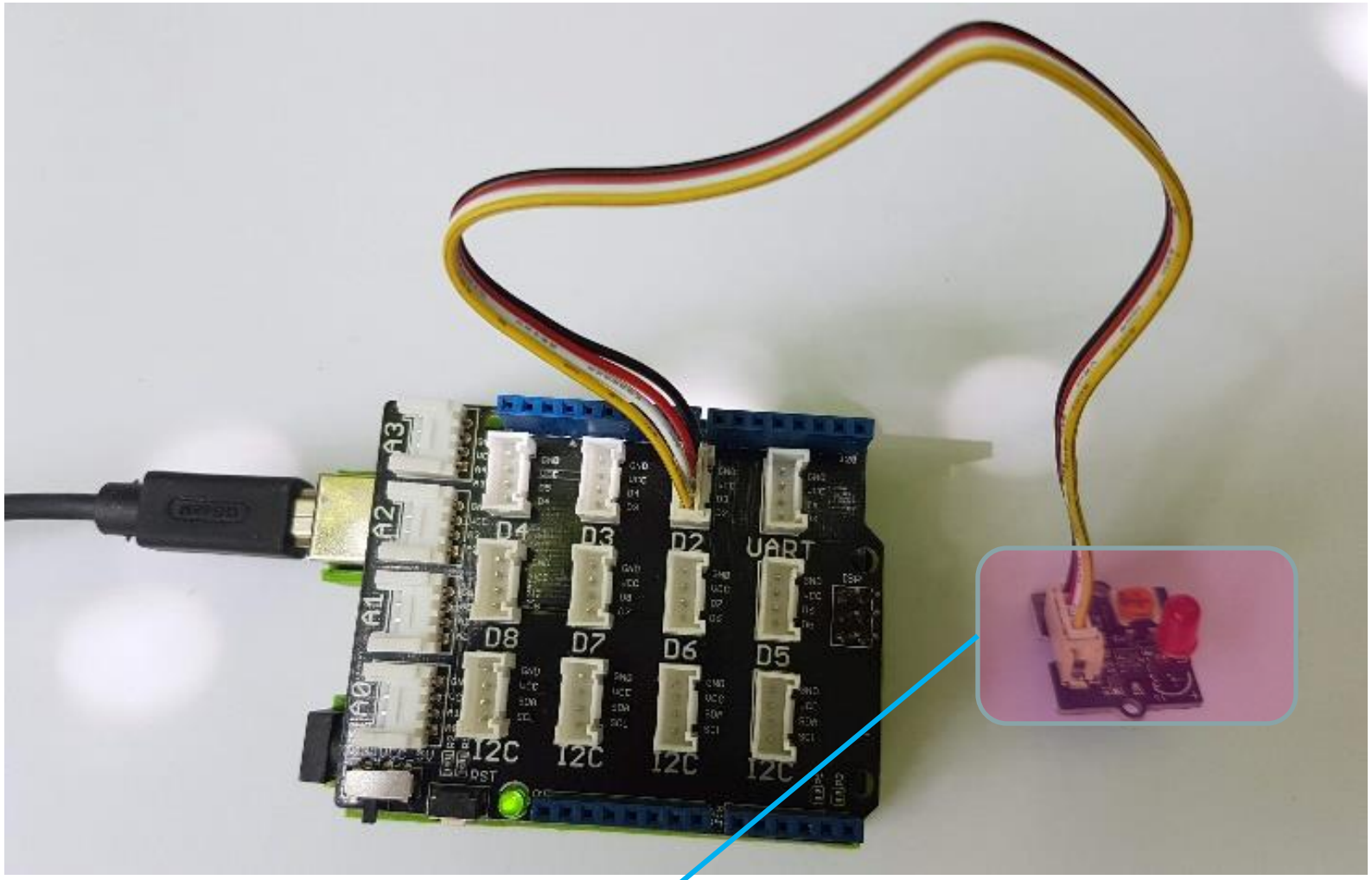


« plugger » (brancher) le Shield sur la carte Arduino

Description du module Groove:

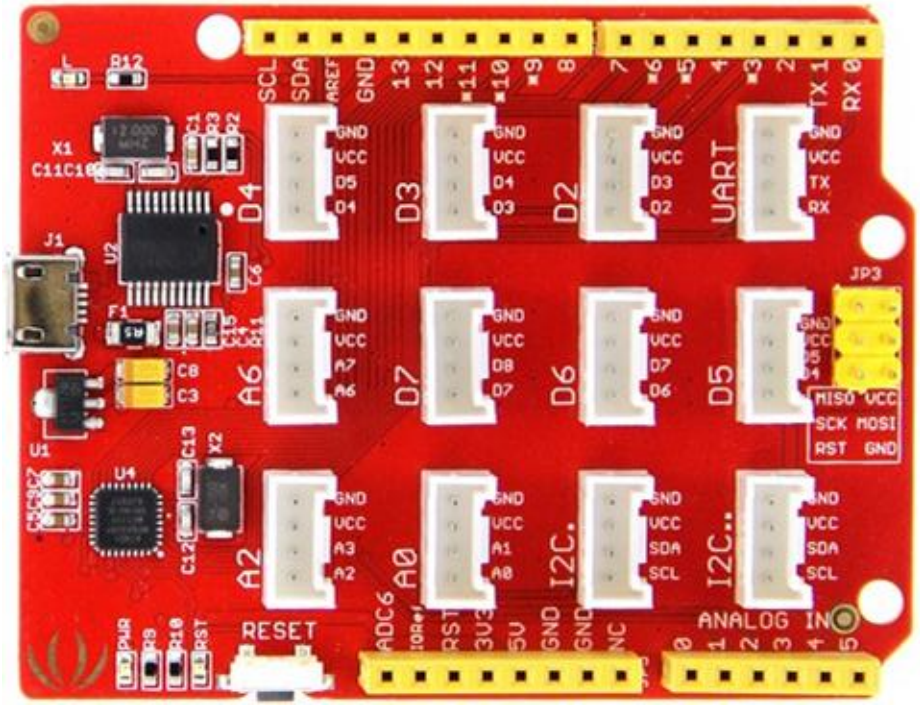


Retour sur le premier montage



LED Groove pré-câblée

Carte Seeduino LOTUS



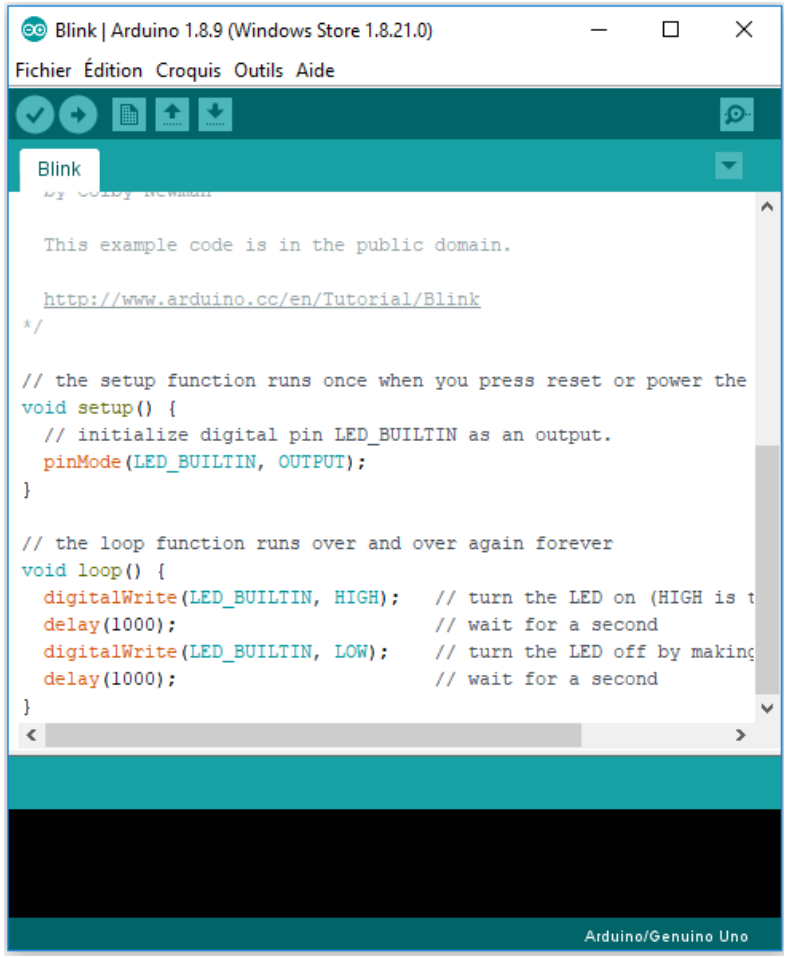
Mélange d'une carte Arduino et d'un module Groove

⇨ **Connections et montages plus simples…**

Tous les montages peuvent se réaliser en Groove

5

Le langage Arduino : C



5 Instructions et commentaires

Lignes de code comprises entre { }

Toute instruction se termine par un ;

Commentaires:

/* commentaires sur …
… plusieurs lignes */

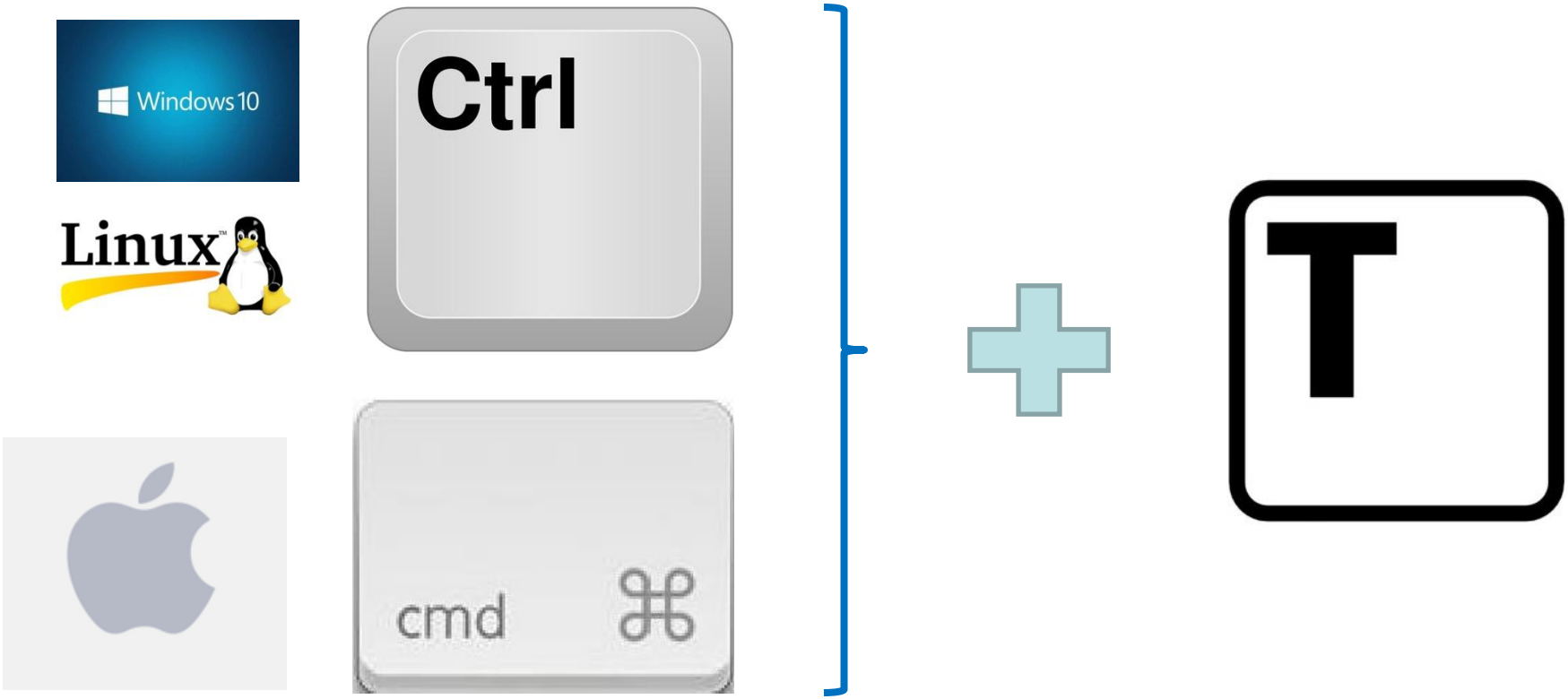
// commentaires sur une ligne

5 Structure générale d'un programme Arduino

```
// appel aux librairies
#include  < nom_de_la_bibliothèque.h >

// définition des constantes
# define    NOM_DE_LA_CONSTANTE     VALEUR_CONSTANTE

void setup() { // (initialisation)
     // instructions à n'exécuter qu'une seule fois;
}

void loop(){ // boucle infinie
    // instructions répétées indéfiniment;
 }
```

5 Indenter le code

Indentation automatique



↳ **Permet une meilleure lecture du code**

5 Déclaration des variable et définitions : en entête

Exemple: variante au programme précédent

Remplace à la compilation le mot « **LED** » par la valeur constante **13** dans tous le programme

```
// Exemple 01 : LED clignotante

# define LED 13

void setup() {
    pinMode(LED, OUTPUT);
}

void loop(){
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

Ecriture équivalentes:

```
// Exemple 01 : LED clignotante

# define LED 13

void setup() {
    pinMode(LED, OUTPUT);
}

void loop(){ … }
```

```
// Exemple 01 : LED clignotante

const byte LED = 2;

void setup() {
    pinMode(LED, OUTPUT);
}

void loop(){ … }
```

- Permet de préciser le type de la variable.
- Inconvénient: consomme de la mémoire RAM

```
// Exemple 01 : LED clignotante

int LED = 2;

void setup() {
    pinMode(LED, OUTPUT);
}

void loop(){ … }
```

Ecriture générale:

#define nom_de_la constante valeur  **Pas de « ; »**

5 Appel aux librairies : inclusion en entête

Pour quoi faire ?

Appel à des fonctionnalité déjà écrites

Ecriture générale:

Pas de « ; »

```
#include  < nom_de_la_bibliothèque.h >
```

Exemple: **Bibliothèque destinée à piloter un afficheur alphanumérique à cristaux liquides:**

```
#include  < LiquidCristal.h >
```

5 Bilan: structure générale d'un programme Arduino

```
// appel aux librairies
#include  < nom_de_la_bibliothèque.h >

// définition des constantes
# define    NOM_DE_LA_CONSTANTE     VALEUR_CONSTANTE

void setup() { // (initialisation)
     // instructions à n'exécuter qu'une seule fois;
}

void loop(){ // boucle infinie
    // instructions répétées indéfiniment;
 }
```

5 Les variables

- Les programmes manipulent un certain nombre de **variables**

  - **Données stockées dans l'espace mémoire réutilisable par la suite**

- Chaque variable est caractérisée par :
  - un **nom**
  - un **type**
  - une **place mémoire**

- Le nom:
  - Peut contenir des lettres MAJUSCULE/ minuscule,
  - chiffres,
  - caractères(-,_)

Type	Nombre stocké	Valeur extrêmes	Nb de bits	Nb d'octet
int	Entier	-32 768 à +32 767	16 bits	2 octets
long	entier	-2 147 483 648 à +2 147 483 647	32 bits	4 octets
char	Caractère ASCII	-128 à +127	8 bits	1 octet
float	Nombre décimal (à virgule)	-3.4×10^{-38} à $+3.4\times10^{38}$	32 bits	4 octets
double	Nombre décimal (à virgule)	-3.4×10^{-38} à $+3.4\times10^{38}$	32 bits	4 octets
unsigned char	Entier positif	0 à 255	8 bits	1 octet
unsigned int	Entier positif	0 à 65 535	16 bits	2 octets
unsigned long	Entier positif	0 à 4 294 967 295	32 bits	4 octets
byte	Entier positif	0 à 255	8 bits	1 octet
word	Entier positif	0 à 65 535	16 bits	2 octets
boolean	Entier positif	0 à 1 – false/true	1 bit	-

Autres variables plus complexes:

- Tableaux de données…
- Chaines de caractères…

Exemples:

```
int entrees[4]  ; // tableau d'entier de 4 éléments à initialiser

byte broches[] = {1, 4, 8, 6}  ; // tableau de 4 éléments

char texte[8]  = « bonjour » ; // chaine de caractères
```

5 Les opérateurs

Addition	Soustraction	Multiplication	Division	Modulo (reste d'une division euclidienne)
+	-	*	/	%

Attribution/ modification d'une valeur	Incrémentation	Décrémentation	Incrémenter et décrémenter de 1	Multiplication ou division par un facteur
=	+=	-=	++ et --	*= ou /=

Comparaison entre deux variables de même type (renvoie un booléen)					
supérieur	Inférieur	Supérieur ou égale	Inférieur ou égale	Égalité	Différence
>	<	>=	<=	==	!=

5 Opérateurs logiques (ET / OU)

ET	OU
&&	||

&&	true	false
true	true	false
false	false	false

||	true	false
true	true	true
false	true	false

Intérêt : permet le test simultané de plusieurs conditions

Condition « if »:

```
if (condition){
// instructions à exécuter
si la condition est vraie }
```

```
if (condition_1) {
// instructions à exécuter si la
condition_1 est vraie }
else if (condition_2) {
// instructions à exécuter si la
condition_2 est vraie}
else {
// instructions à exécuter si les
condition_1 et condittion_2
sont fausse}
```

```
if (condition_1) {
// instructions à exécuter si la
condition_1 est vraie }
else {
// instructions à exécuter si la
condition_1 est fausse}
```

Remarque: les conditions peuvent être multiples

Ex:

```
if (pousoirA == 1 && pousoirB == 1){
digitalWrite (LED,HIGH);
}
```

Condition « switch … case »:

```
switch (variable) {
    case valeur_1 :
        instructions à exécuter si la variable = valeur_1 ;
        break;
    case valeur_2 :
        instructions à exécuter si la variable = valeur_2 ;
        break;

                                .....


    case valeur_n :
        instructions à exécuter si la variable = valeur_n ;
        break;

}
```

Boucle « for »:

```
for (valeur_initiale ; condition_finale ; incrément) {
    instructions à exécuter dans la boucle tant que la condition finale
n'est pas atteinte ;
}
```

```
Ex:    for (byte i=0; i<=255 ; i++){
           analogWrite (2,i);
           delay(100);
       }
```

Boucle « while »:

```
while (condition) {
    instructions à exécuter dans la boucle tant que la condition finale
n'est pas atteinte ;
}
```

```
Ex:    byte i=0;
       while (i<256) {
           analogWrite (2,i);
           delay(100);
       }
```

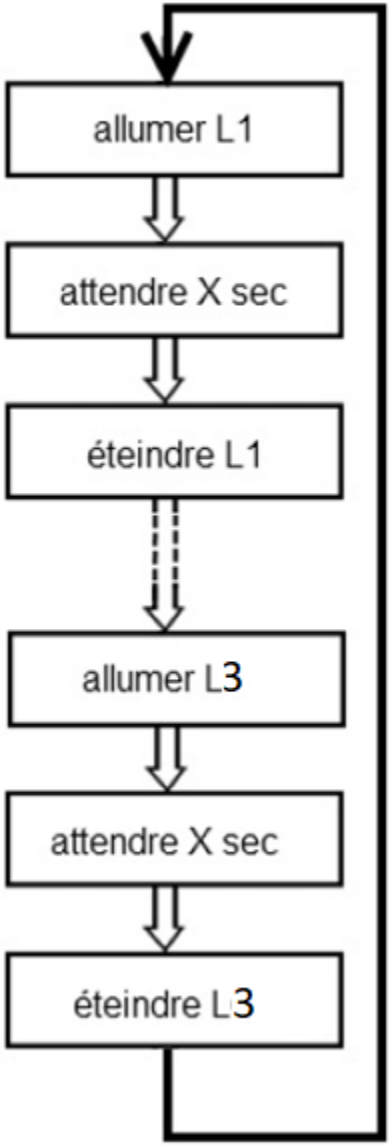
6

Des montages Simples

- Entrées / Sorties numériques
- Entrées Analogiques
- Potentiomètre
- Capteurs
- Actionneurs

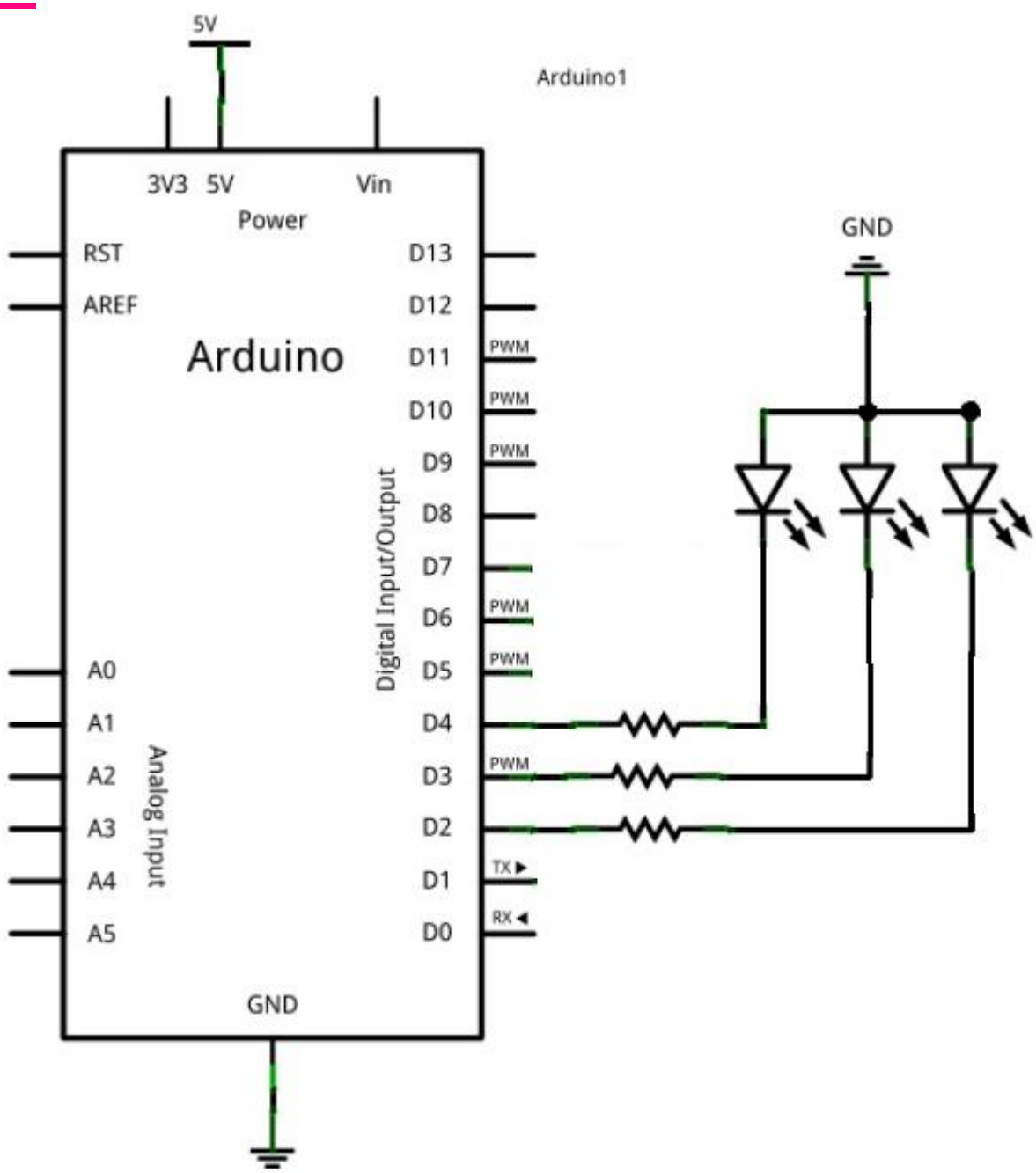
6 1. Chenillard à 3 LED : Utilisation des sorties digitales

Algorithme :

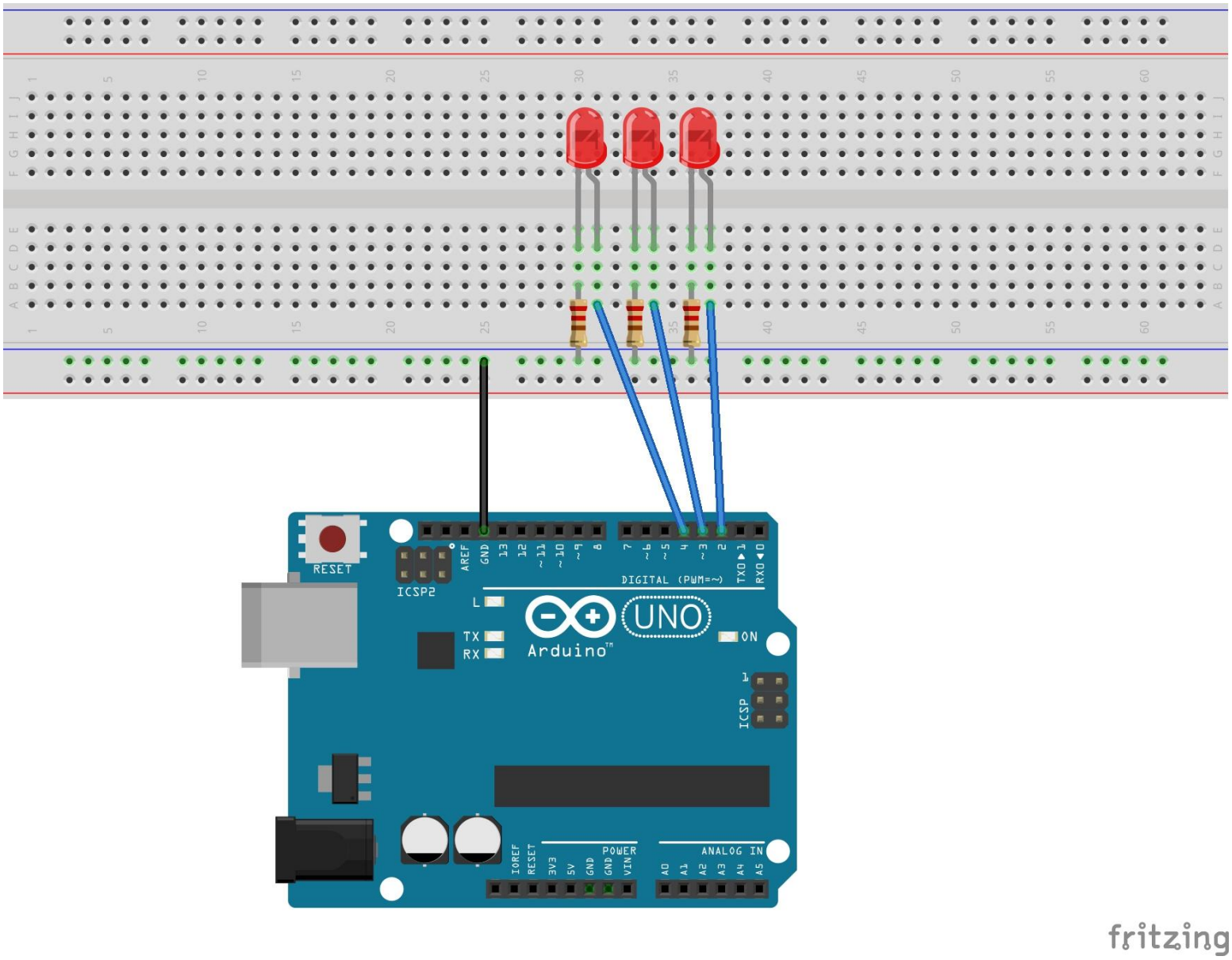


Boucler le programme

Schéma du montage :

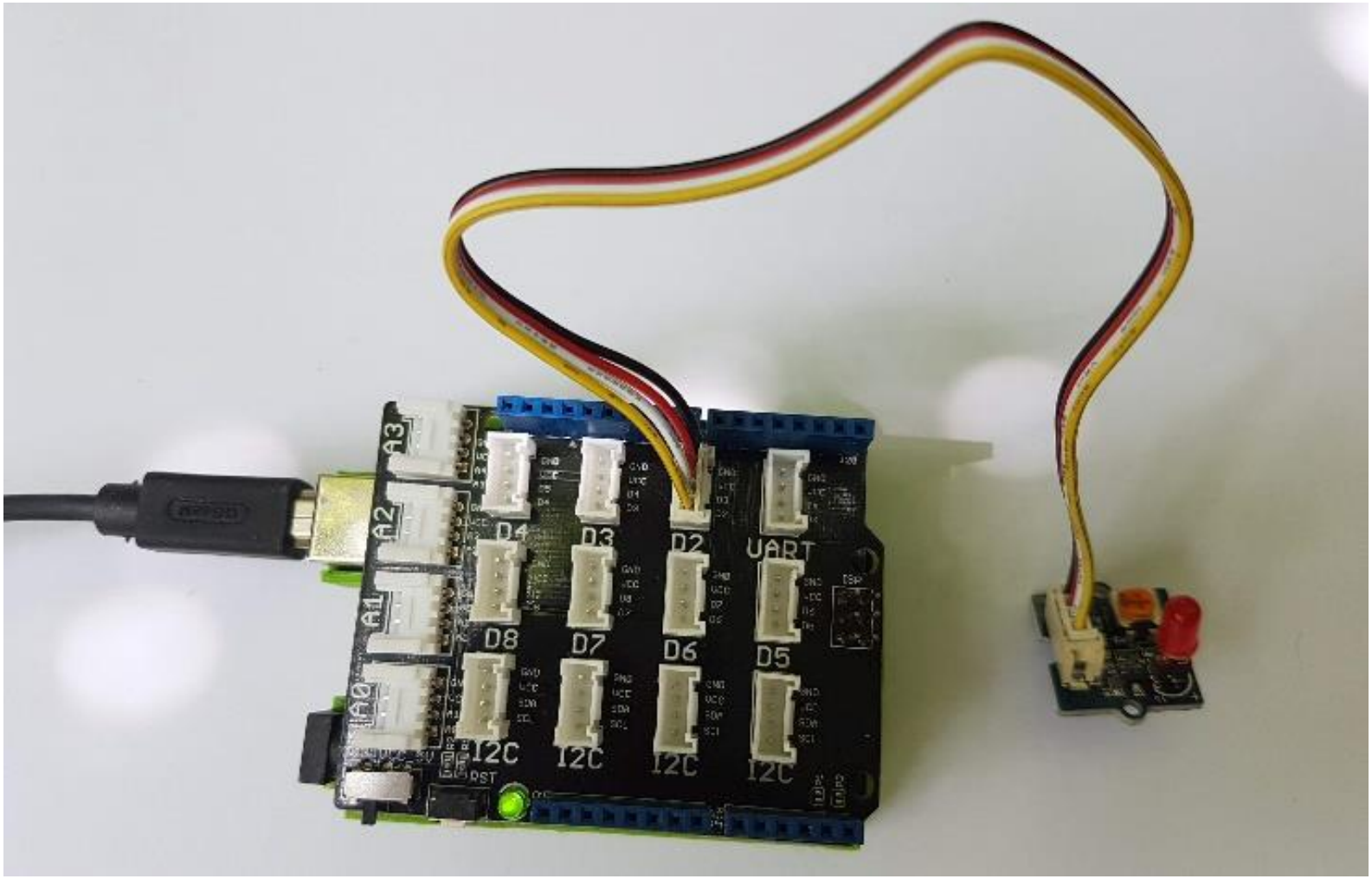


Montage câblé:



Montage Groove:

Brancher 3 LED Groove sur D2, D3 et D4



Programme:

```
// déclaration des variables globales
int led[ ] ={2, 3, 4} ;
// ici led[0] = 2, (on commence la numérotation à 0)
// led[1] = 3, …

void setup() {
   for (byte i=0; i<3 ; i++){
      pinMode(led[i],OUTPUT);
   }
} // ici i est de type « byte » car on ne compte que jusqu'à 6 donc il est
inutile d'utiliser 2 octets (int)

void loop(){
       for (byte i = 0; i<3; i++){
            // allumer la LED i
            digitalWrite(led[i], LOW);
           // attendre 1 secondes
           delay(1000);
           // éteindre la LED i
          digitalWrite(led[i], HIGH);
       // la LED i+1 sera allumée en même temps que la LED i sera
       éteinte
       delay(1000);
       }
}
```

Exercice:

Créer une fonction « clignoterLed() » permettant de faire clignoter une led :

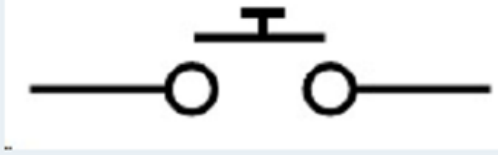
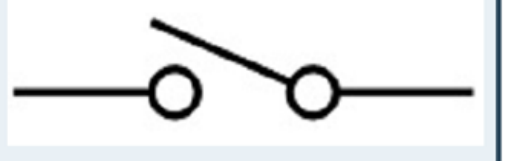
- d'indice « indiceLed »
- temps d'allumage : « Ta »
- temps où la Led est éteinte : « Te »

Aide : La fonction est le suivante

```
void clignoterLed(byte indiceLed, int Ta, int Te){

 ……

}
```

2. Bouton poussoir : Utilisation des entrées digitales

https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino/3285224-le-bouton-poussoir

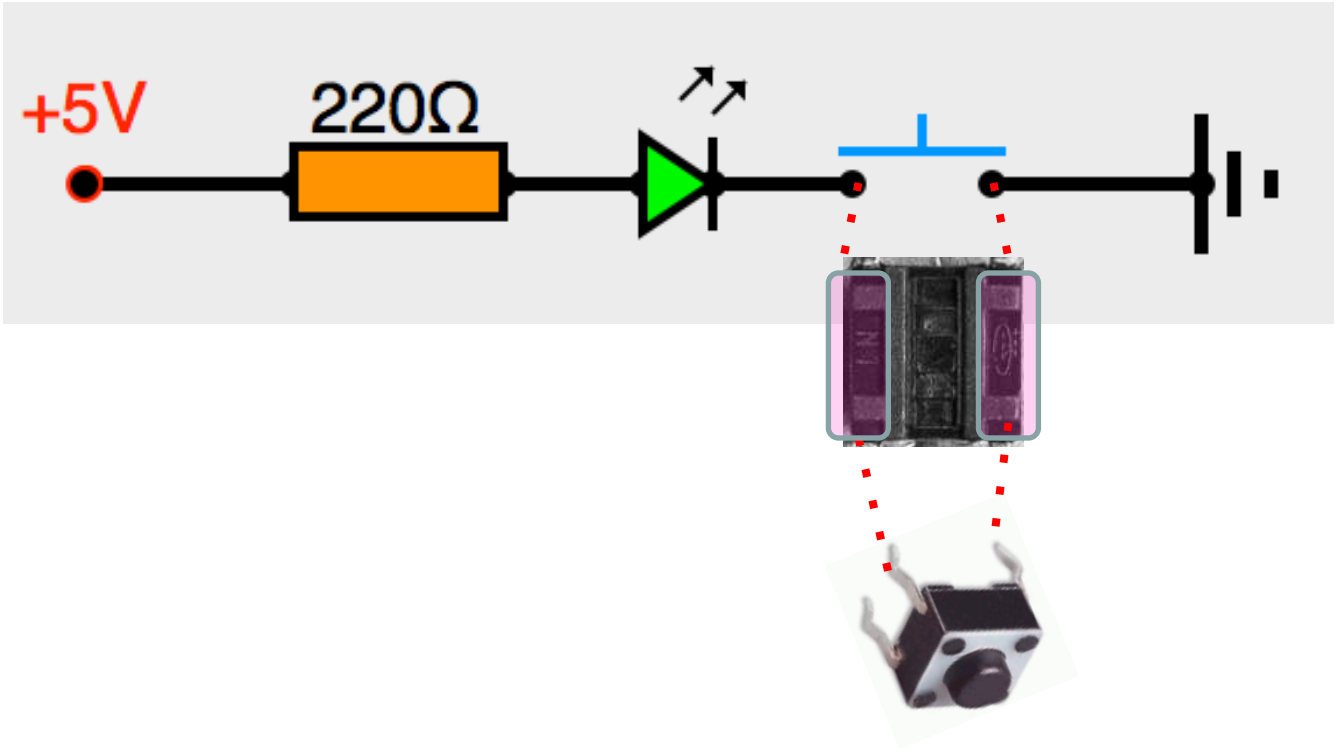
		
Bouton Poussoir NO	Bouton Poussoir NF	Interrupteur

- **<u>Circuit ouvert</u>: le courant ne passe pas.**
- **<u>Circuit fermé</u>: le courant passe**

<u>Exemple</u>:

Utiliser les pates opposées au canal central

Principe du bouton poussoir : un premier montage
Sans programmation

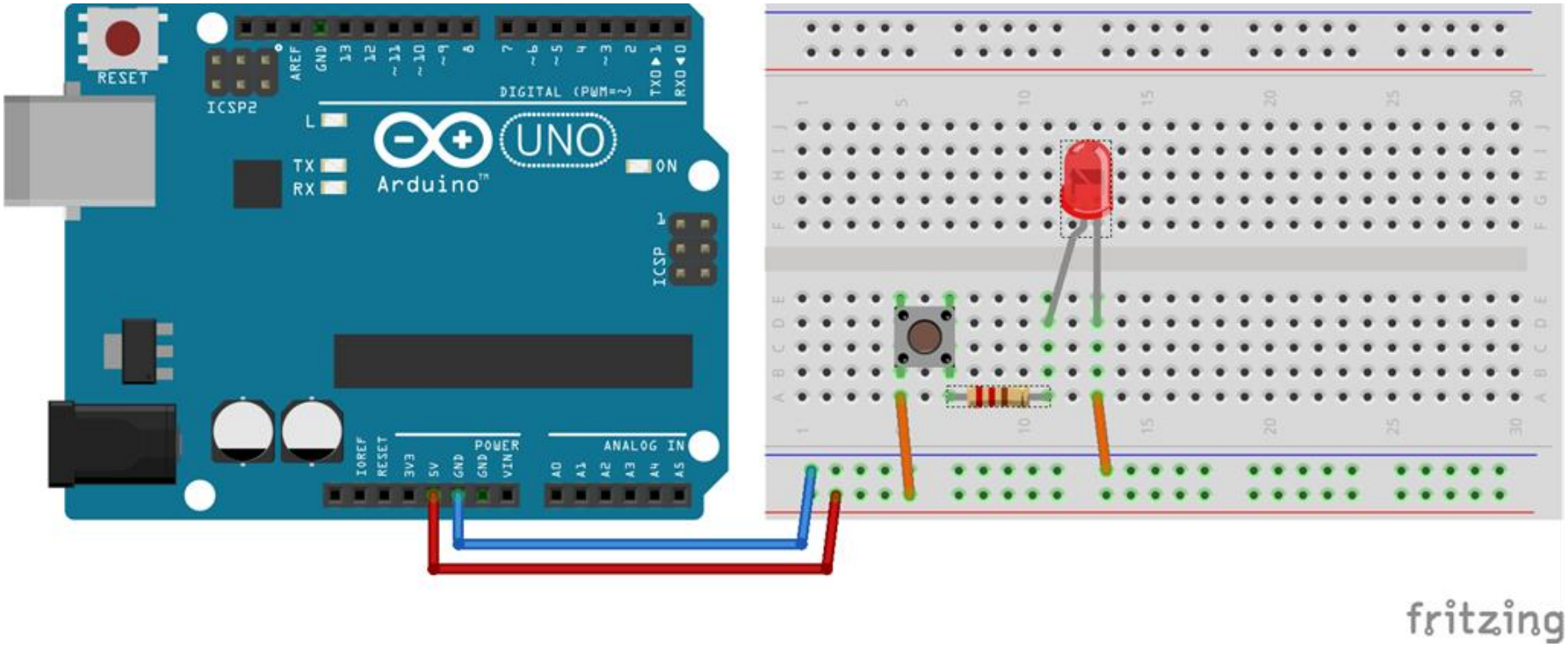


⇨ **Lorsque l'on appuie, le courant passe**

⇨ **Lorsque l'on relâche, le courant ne passe plus**

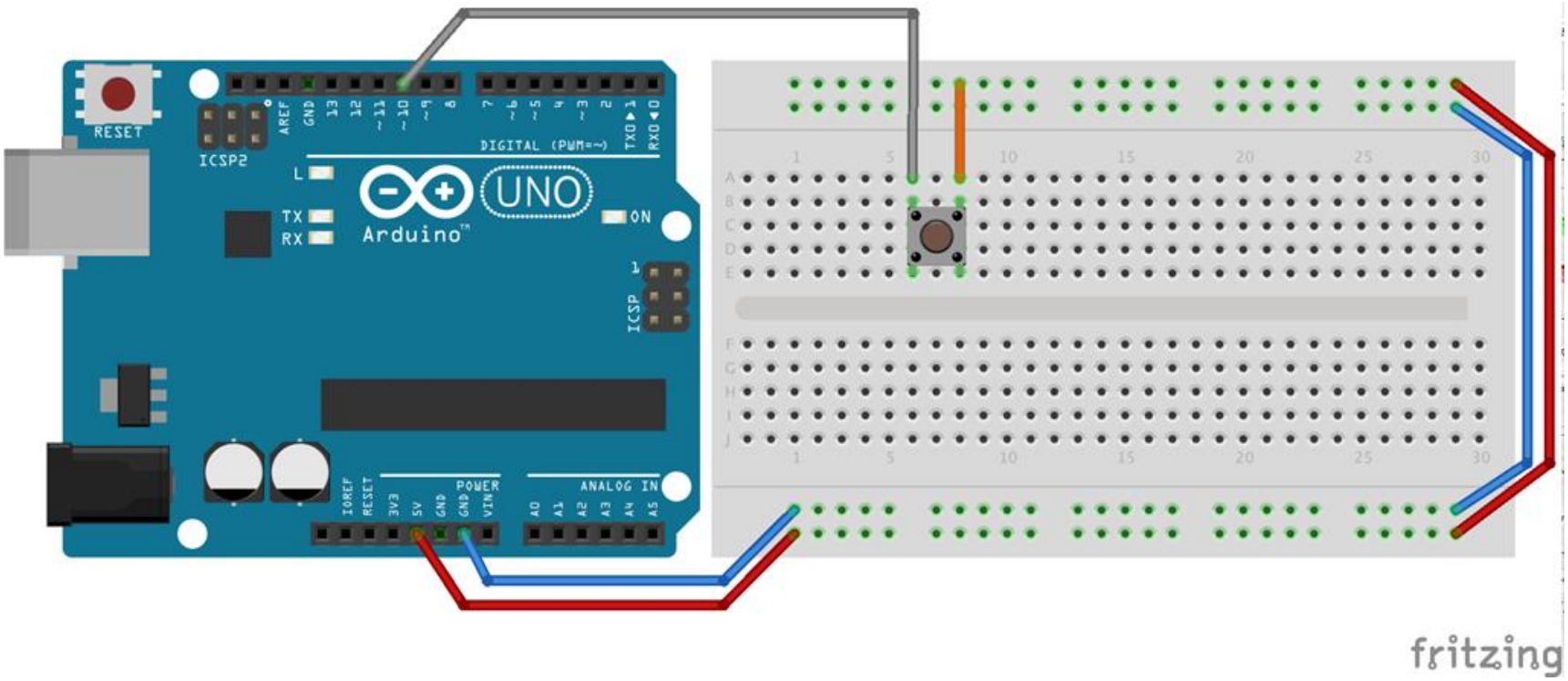
Montage câblé:

R ≈ 220Ω



- **Réaliser le montage**
- **Vérifier que la LED s'allume lorsque le bouton est pressé et qu'elle s'éteint lorsqu'on le relâche**

Utilisation du bouton poussoir sur les entrées digitales



Régler la broche digitale D_i en mode INPUT

Dans le setup():

```
pinMode(pin,INPUT);
```

Programme:

```
int pinBouton;
void setup() {
    Serial.begin(9600);
    pinBouton=10;
    pinMode(pinBouton, INPUT );
}
void loop() {
    boolean etatBouton=digitalRead(pinBouton);
    Serial.println(etatBouton);
}
```

Sur le moniteur série de l'IDE, observer le comportement du circuit en appuyant sur le bouton « loupe »

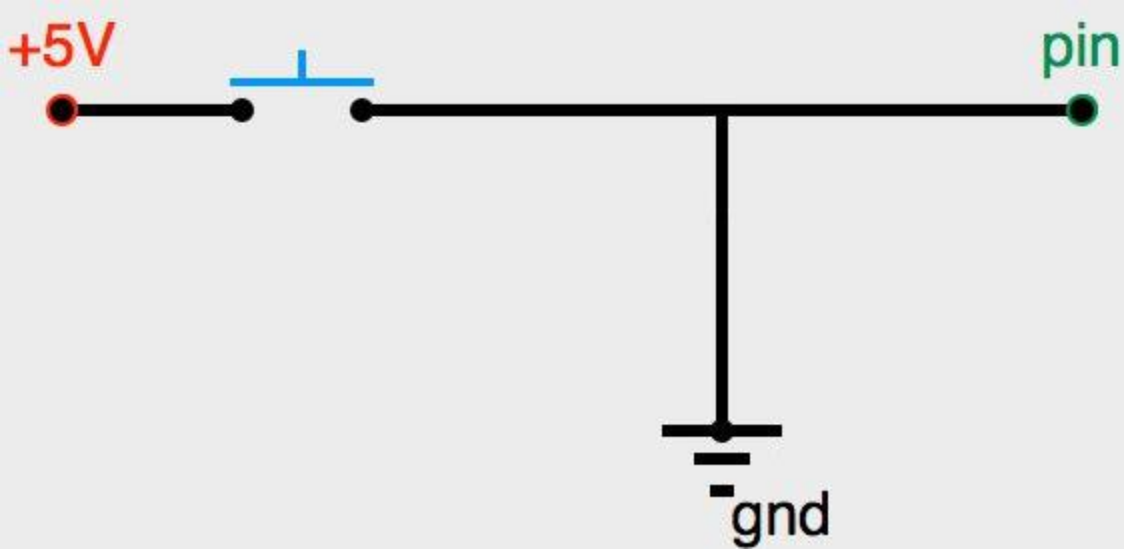
Résultat:

Le circuit a un comportement erratique :

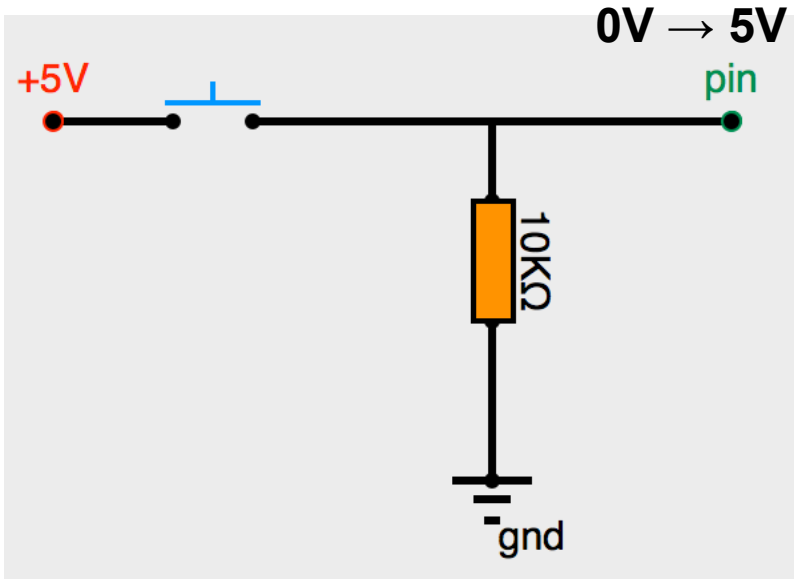
- **Bouton relâché : 0**
- **Bouton appuyé: 1**
- **Bouton relâché à nouveau: reste à 1**

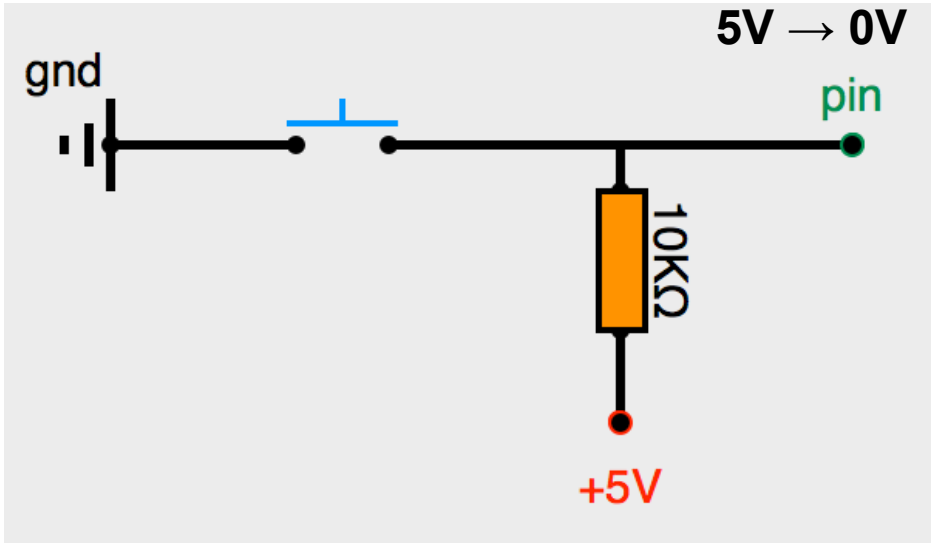
Pourquoi ?

court-circuit entre la masse (0V) et la tension d'alimentation (5V)



Solution:





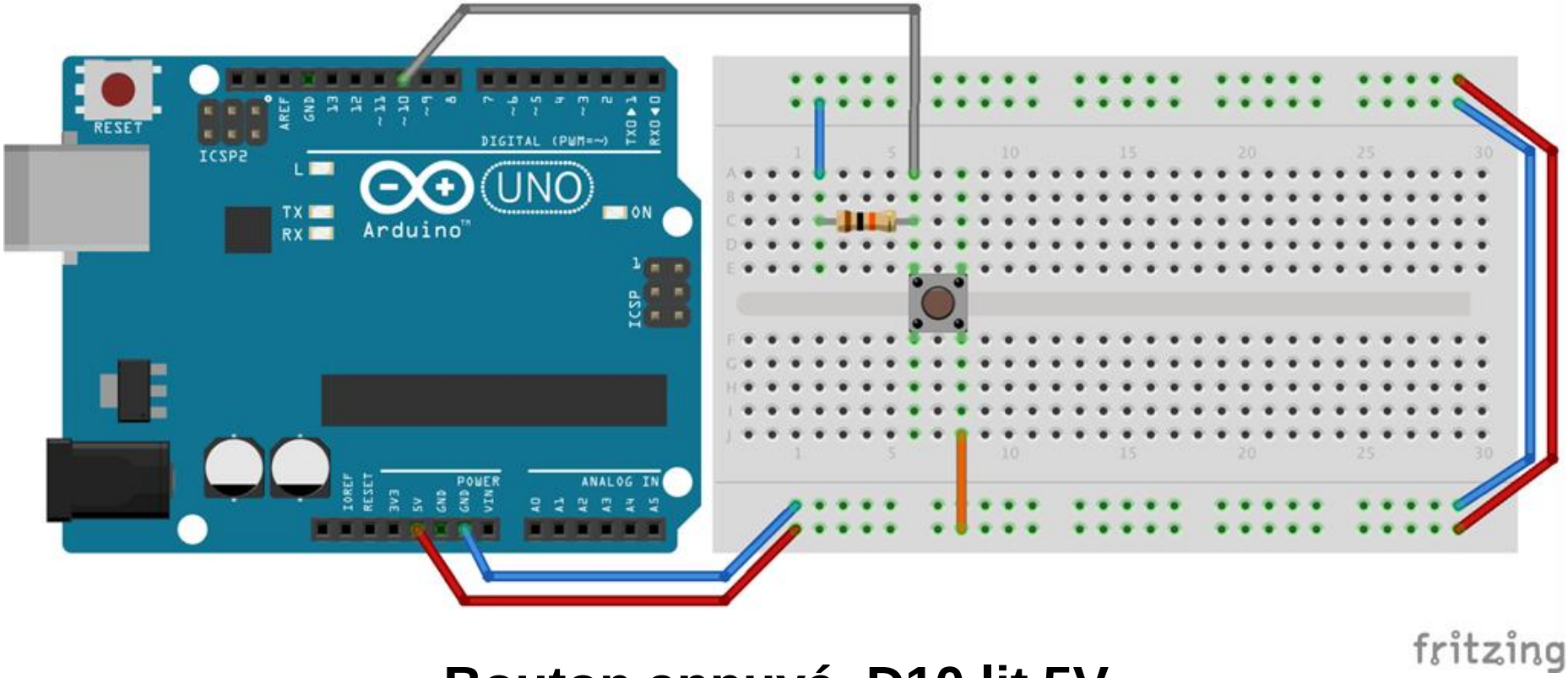
Résistance de PULL-DOWN

Résistance de PULL-UP

- Di se comporte comme un voltmètre entre le point de branchement et la masse
- La résistance permet d'éviter le court-circuit

Montage PULL-DOWN:

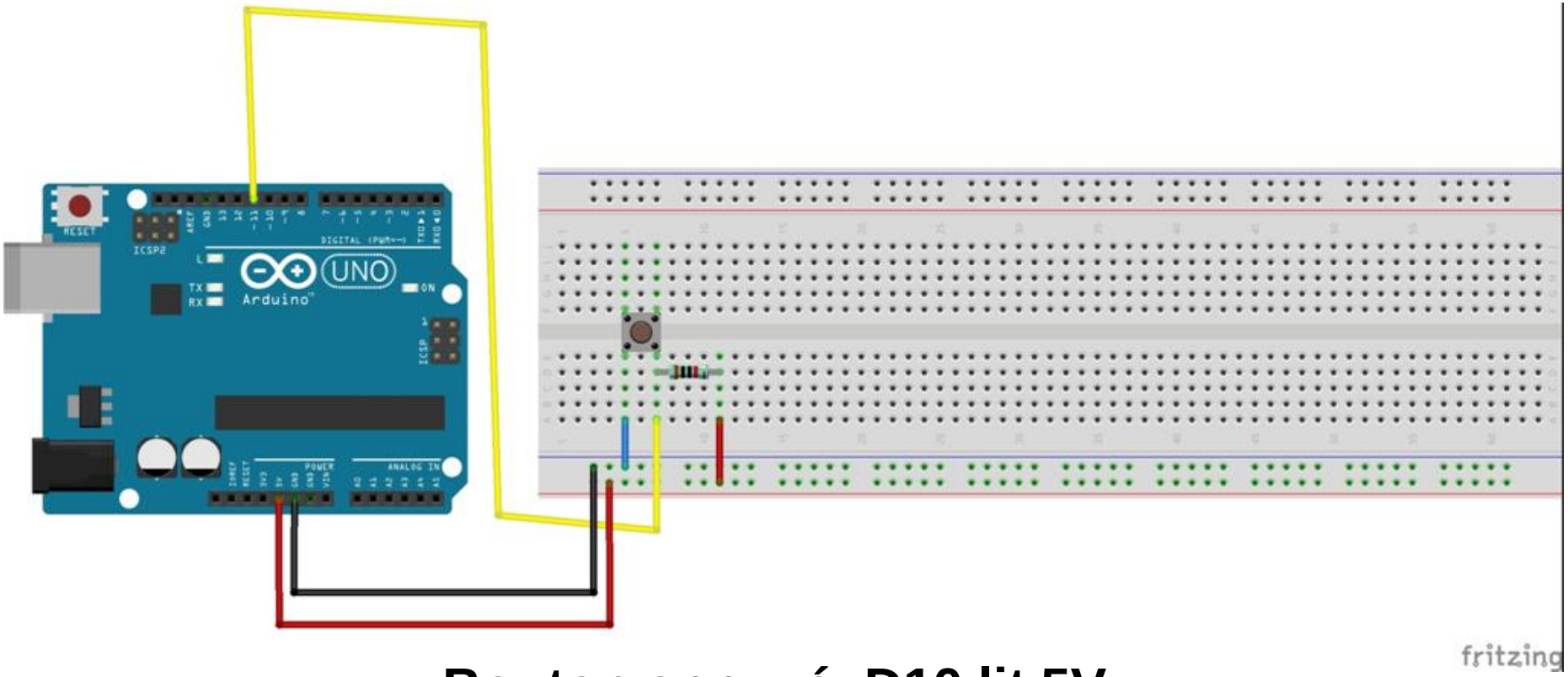
Bouton relâché, D10 lit 0V



Bouton appuyé, D10 lit 5V

Montage PULL-DOWN:

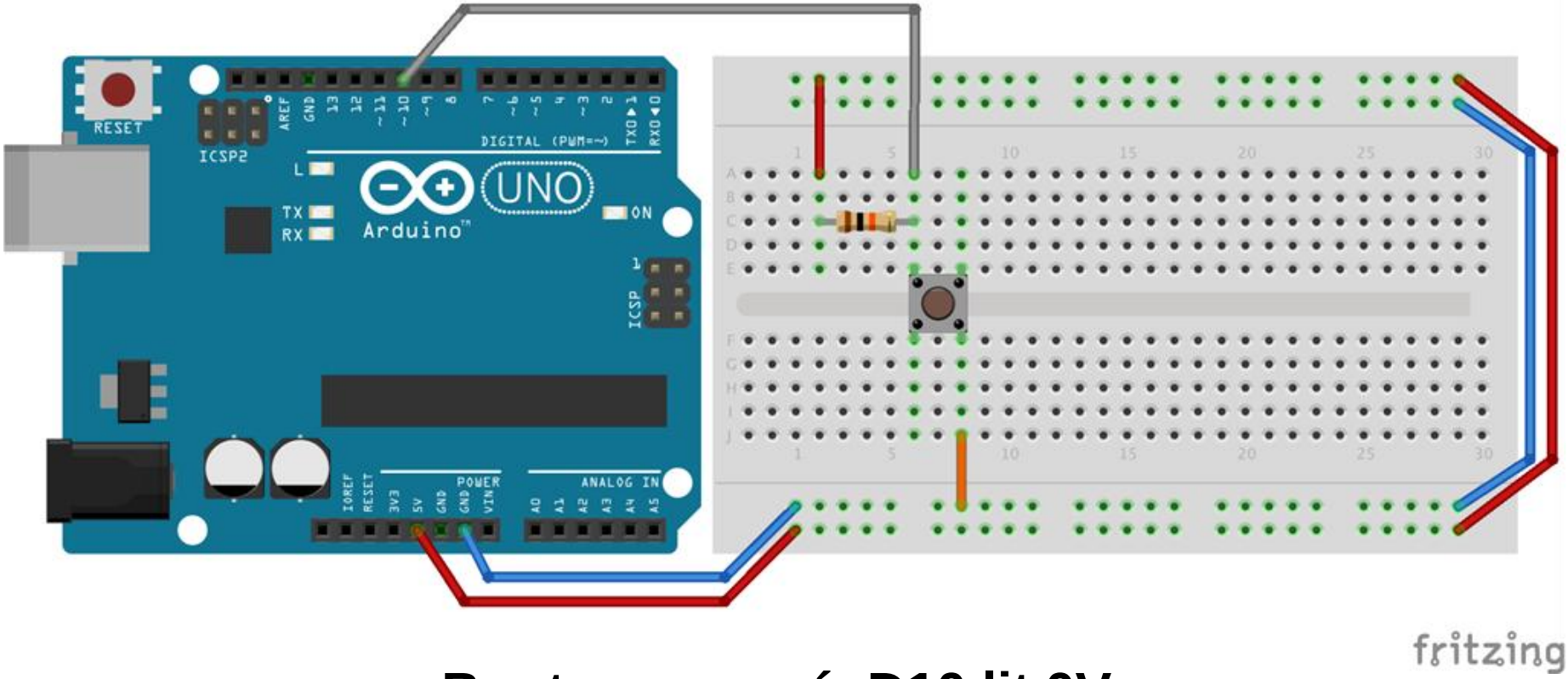
Bouton relâché, D10 lit 0V



Bouton appuyé, D10 lit 5V

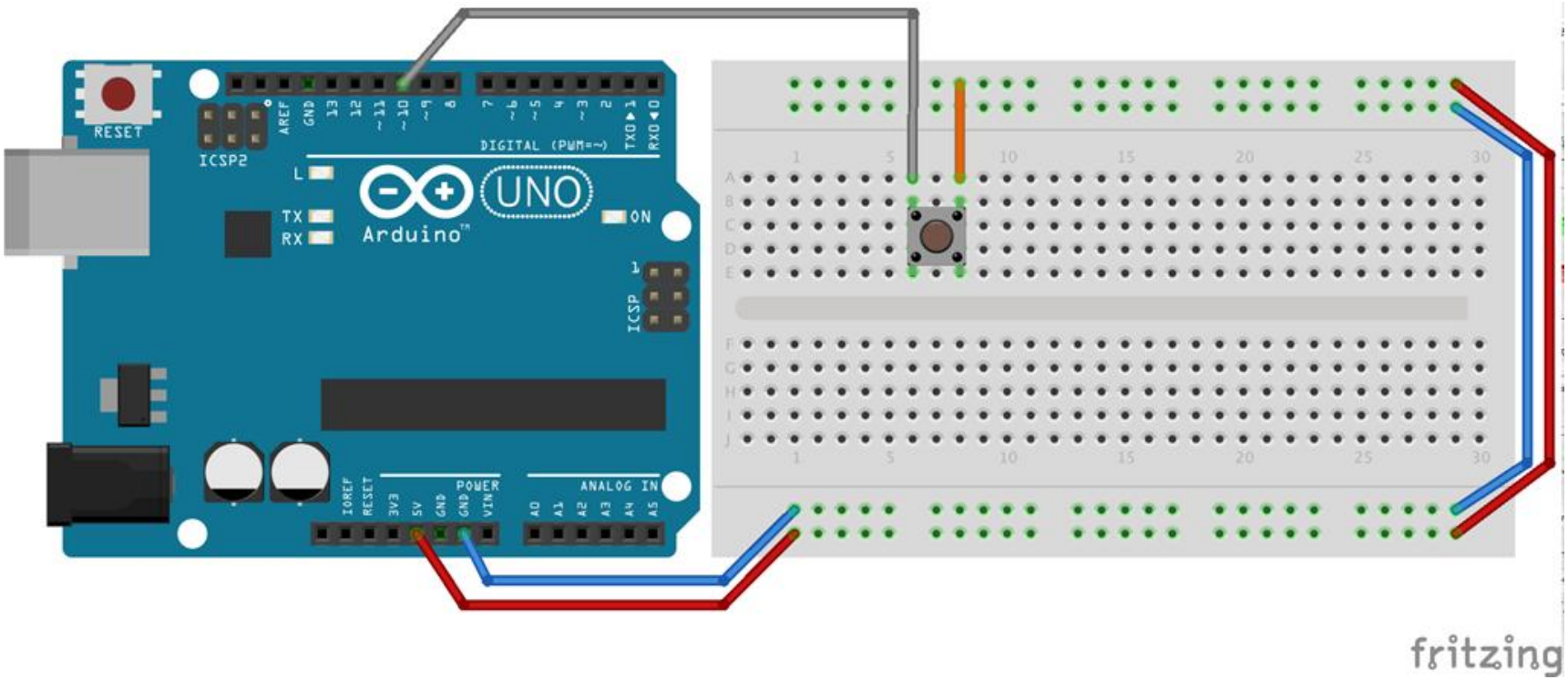
Montage PULL-UP:

Bouton relâché, D10 lit 5V



Bouton appuyé, D10 lit 0V

Solution de facilité : Mode INPUT_PULLUP



Activer la résistance interne de 20kΩ de la broche digitale D_i en mode INPUT_PULLUP

Dans le setup():

```
pinMode(pin,INPUT_PULLUP);
```

1er Montage (1LED + 1 bouton poussoir)

Objectif: Allumer la LED si le bouton est fermer et l'éteindre sinon.

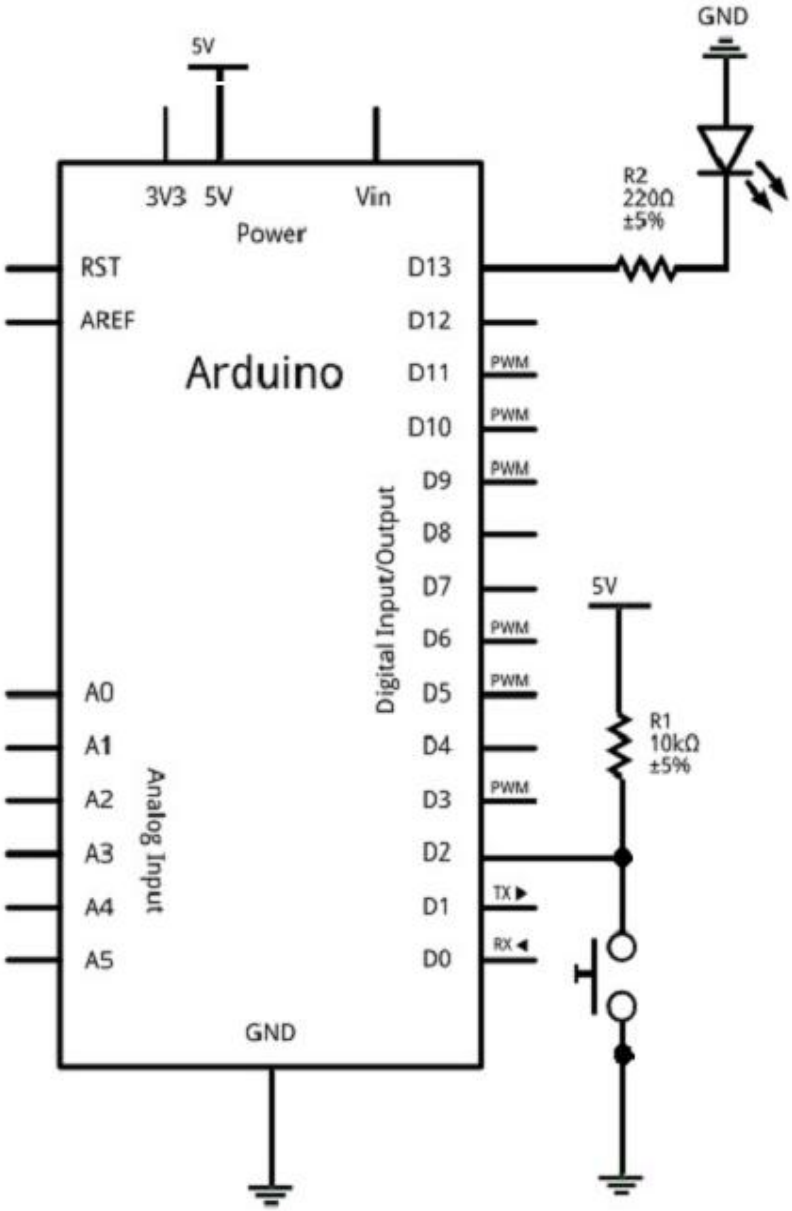
Algorithme: Si le bouton est fermé alors allumer la LED sinon éteindre la LED.

Fonction à utiliser pour tester l'état du bouton:

valeur = digitalRead(n°_de_pate)

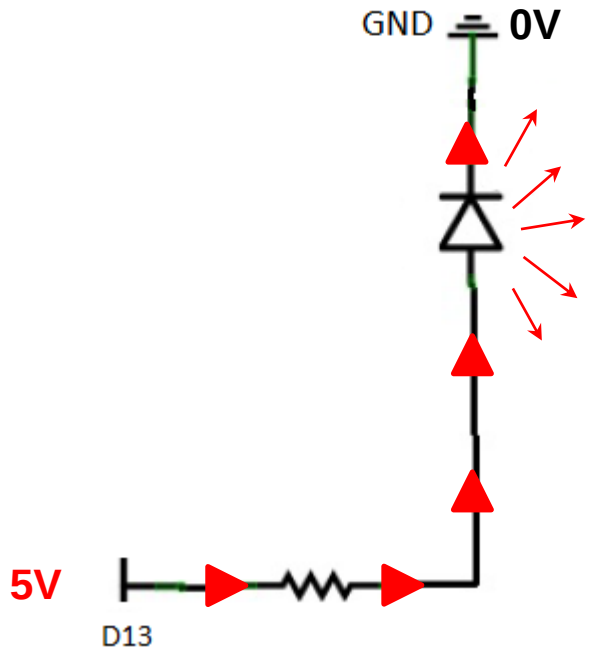
- **Teste la tension présente sur la pate «n°_de_pate »**
- **Retourne la valeur HIGH ou LOW suivant la tension lue**
  - **Ici valeur = HIGH ou LOW**

Montage

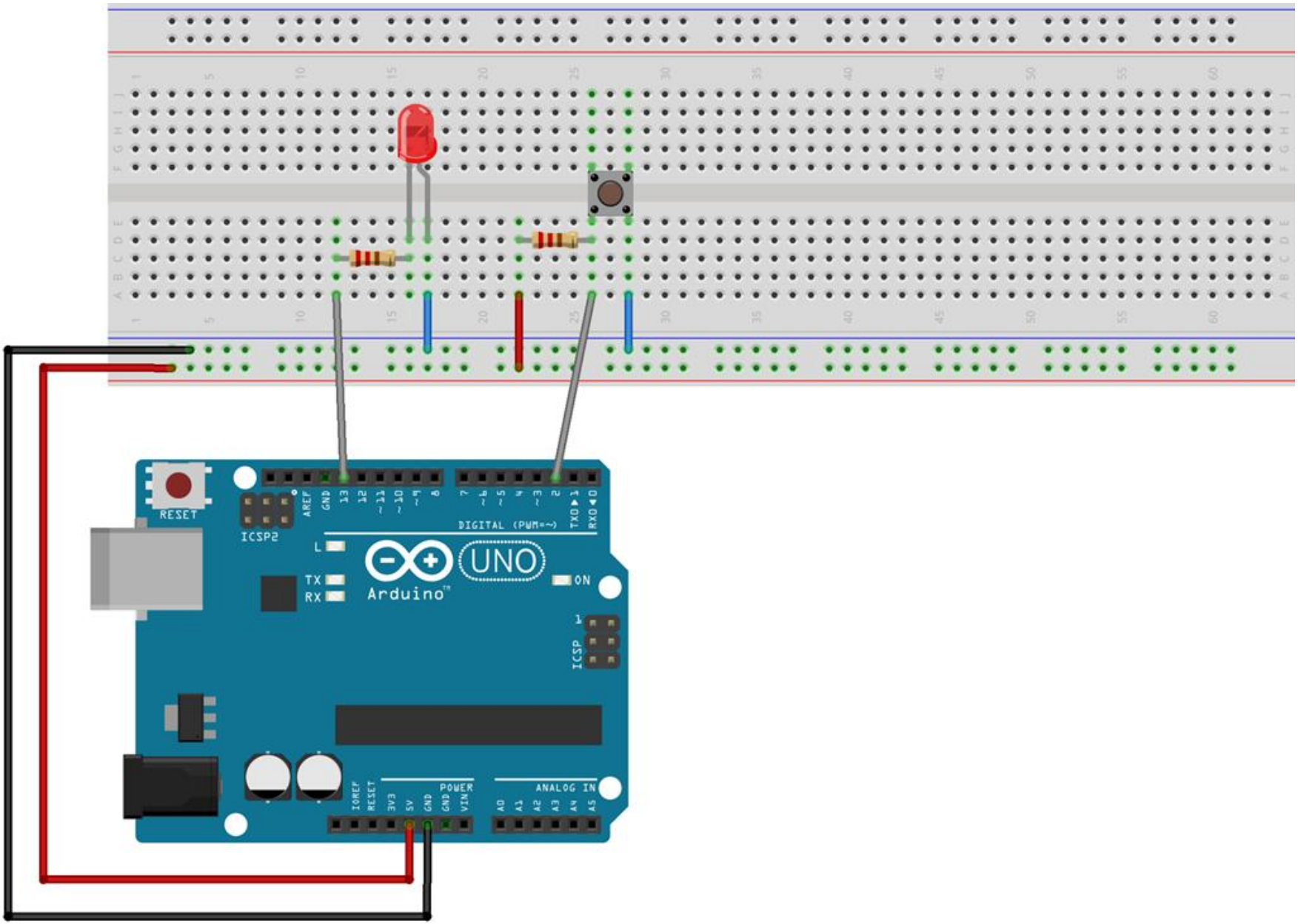


Q: Quelle doit être la valeur de D13 pour que la LED s'allume ?

D13 → HIGH

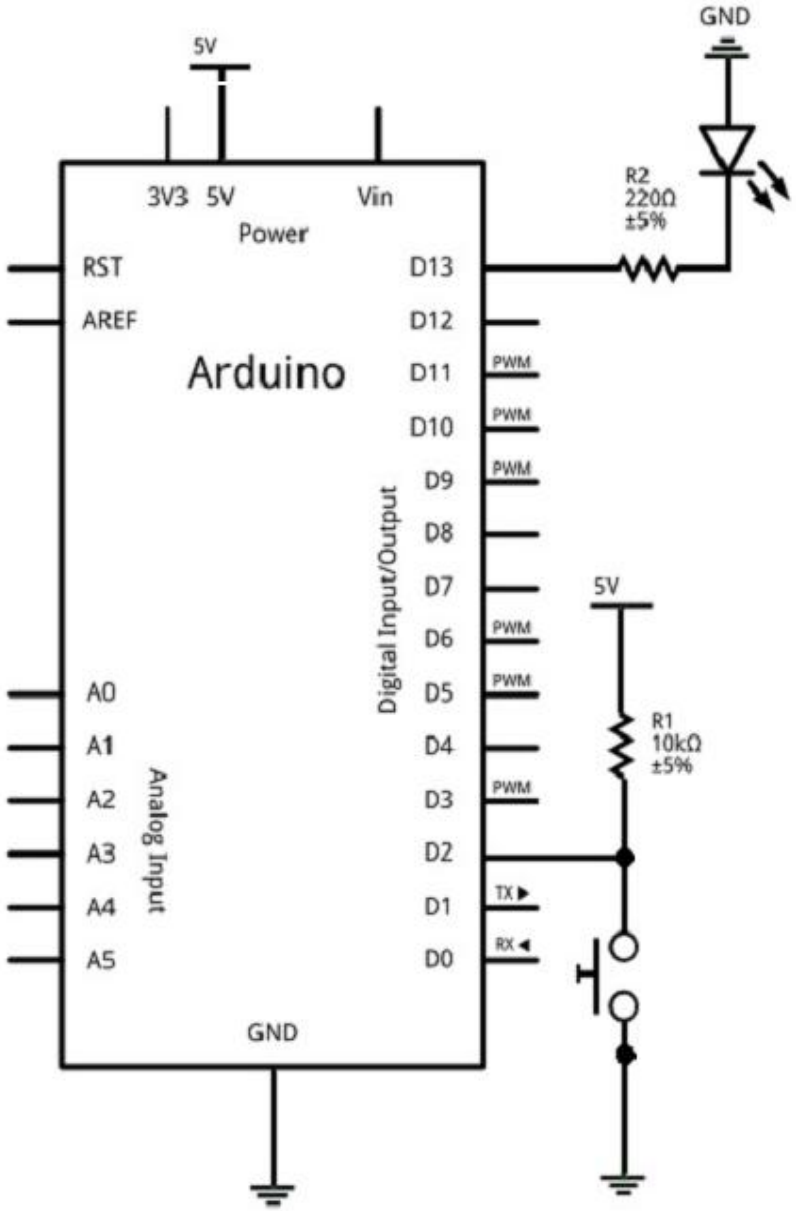


Montage



fritzing

Montage



Programme

```
//Déclaration des varables globales
#define LED 13
//broche 13 du micro-contrôleur se nomme maintenant  LED
#define BOUTON 2
//broche 2 du micro-contrôleur se nomme maintenant  BOUTON
int etat_bouton= 0;
//La variable « etat _bouton» mémorise l'état HIGH ou LOW de la pate d'entrée
```

```
void setup(){
    pinMode(LED,OUTPUT);
//la broche 13 est une sortie
    pinMode(BOUTON,INPUT);
//la broche 2 est une entrée
}
```

Programme (suite)

```
void loop(){

    etat_bouton = digitalRead(BOUTON);
    // lit et mémorise l'état en entrée de la pate 2

    if (etat_bouton == LOW){ // si l'entrée 2 est à l'état LOW (bouton appuyé)

        digitalWrite(LED, LOW); // allume  la LED
     }

    else {//si l'entrée 2 est à l'état LOW

        digitalWrite(LED, HIGH); // éteind la LED
     }

} // fin de loop()
```

Programme total

```
//Déclaration des varables globales
#define LED 13
//broche 13 du micro-contrôleur se nomme maintenant  LED
#define BOUTON 2
//broche 2 du micro-contrôleur se nomme maintenant  BOUTON
int etat_bouton= 0;
//La variable « etat _bouton» mémorise l'état HIGH ou LOW de la pate d'entrée

void setup(){
  pinMode(LED,OUTPUT);
//la broche 13 est une sortie
  pinMode(BOUTON,INPUT);
//la broche 2 est une entrée
}

void loop(){

        etat_bouton = digitalRead(BOUTON);
        // lit et mémorise l'état en entrée de la pate 2

        if (etat_bouton == LOW){ // si l'entrée 2 est à l'état LOW (bouton appuyé)

            digitalWrite(LED, LOW); // allume  la LED
         }

        else {//si l'entrée 2 est à l'état LOW

            digitalWrite(LED, HIGH); // éteind la LED
         }

} // fin de loop()
```

2ième Montage : Bouton poussoir à mémoire

Objectif: Réaliser une lampe qui garde en mémoire son état de fonctionnement…

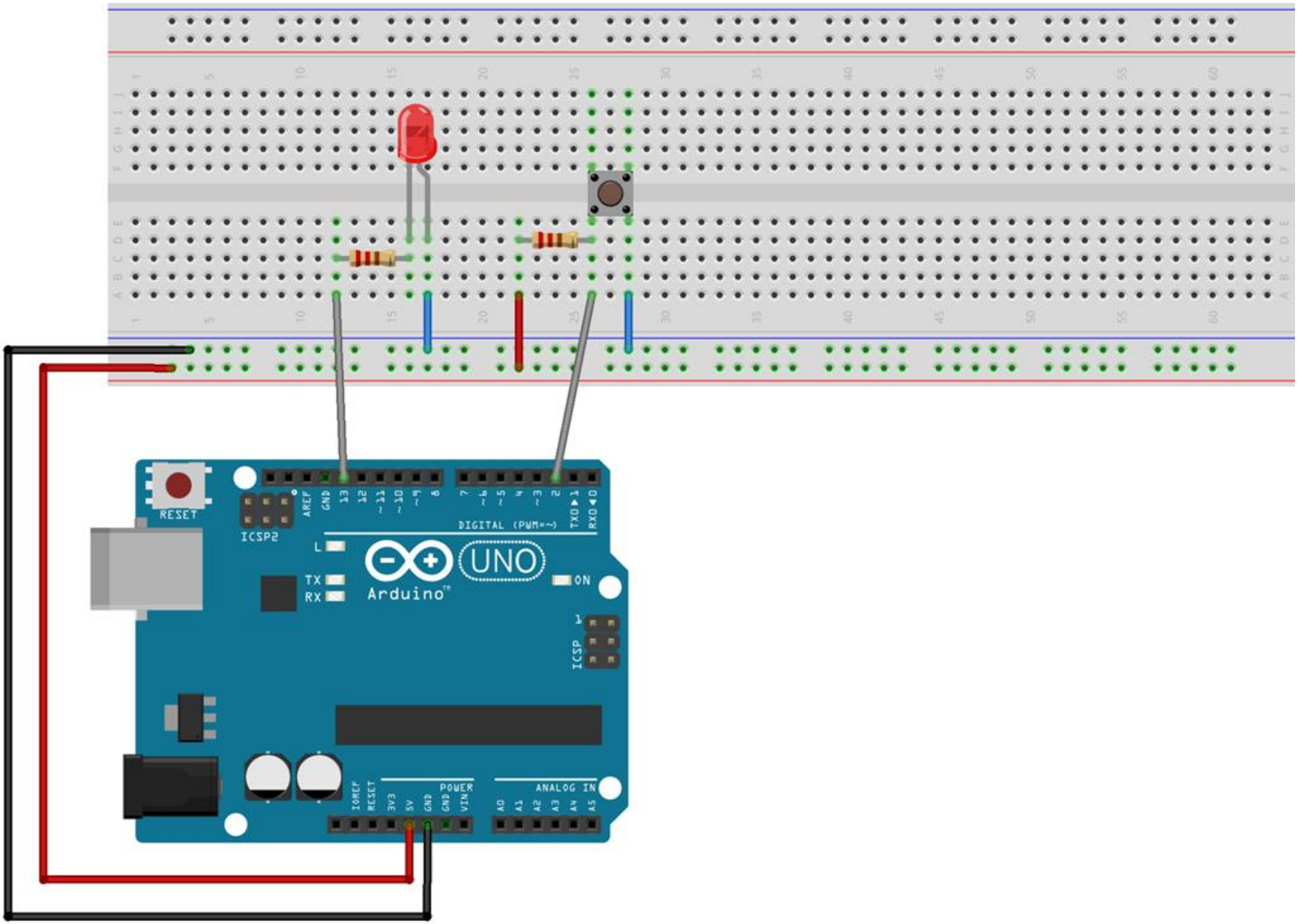
Algorithme:

Si le bouton est fermé alors :

- **Si la LED était éteinte alors allumer la LED et conserver l'état de la LED allumée tant que le bouton n'est pas appuyé à nouveau.**

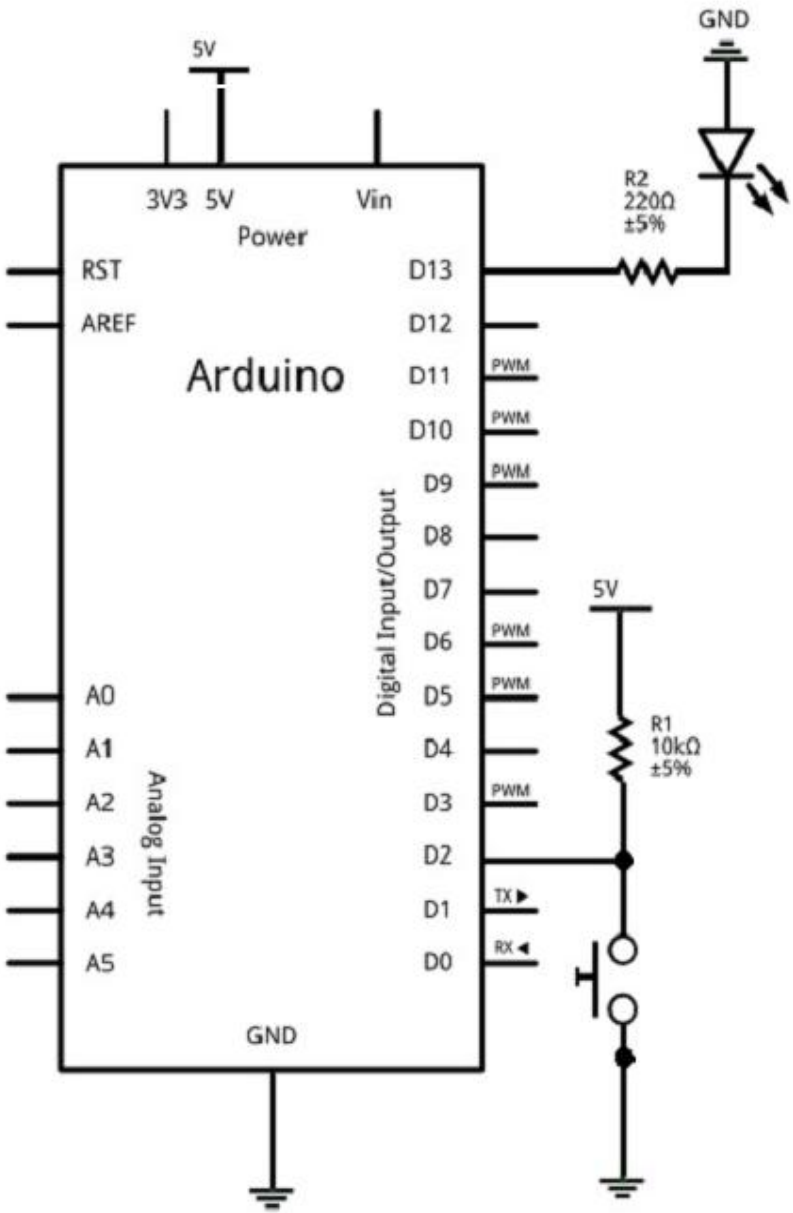
- **Si la LED était allumée alors éteindre la LED et conserver l'état de la LED éteinte tant que le bouton n'est pas appuyé à nouveau.**

Montage



fritzing

Montage



Programme

```
//Déclaration des varables globales
#define LED 13
//broche 13 du micro-contrôleur se nomme maintenant  LED
#define BOUTON 2
//broche 2 du micro-contrôleur se nomme maintenant  BOUTON
int etat_bouton= 0;
//La variable « etat_bouton » mémorise l'état HIGH ou LOW de la pate d'entrée
int etat_led= 0;
//La variable « etat_led » mémorise l'état 1 (allumée) ou 0 (éteinte) de la led.
```

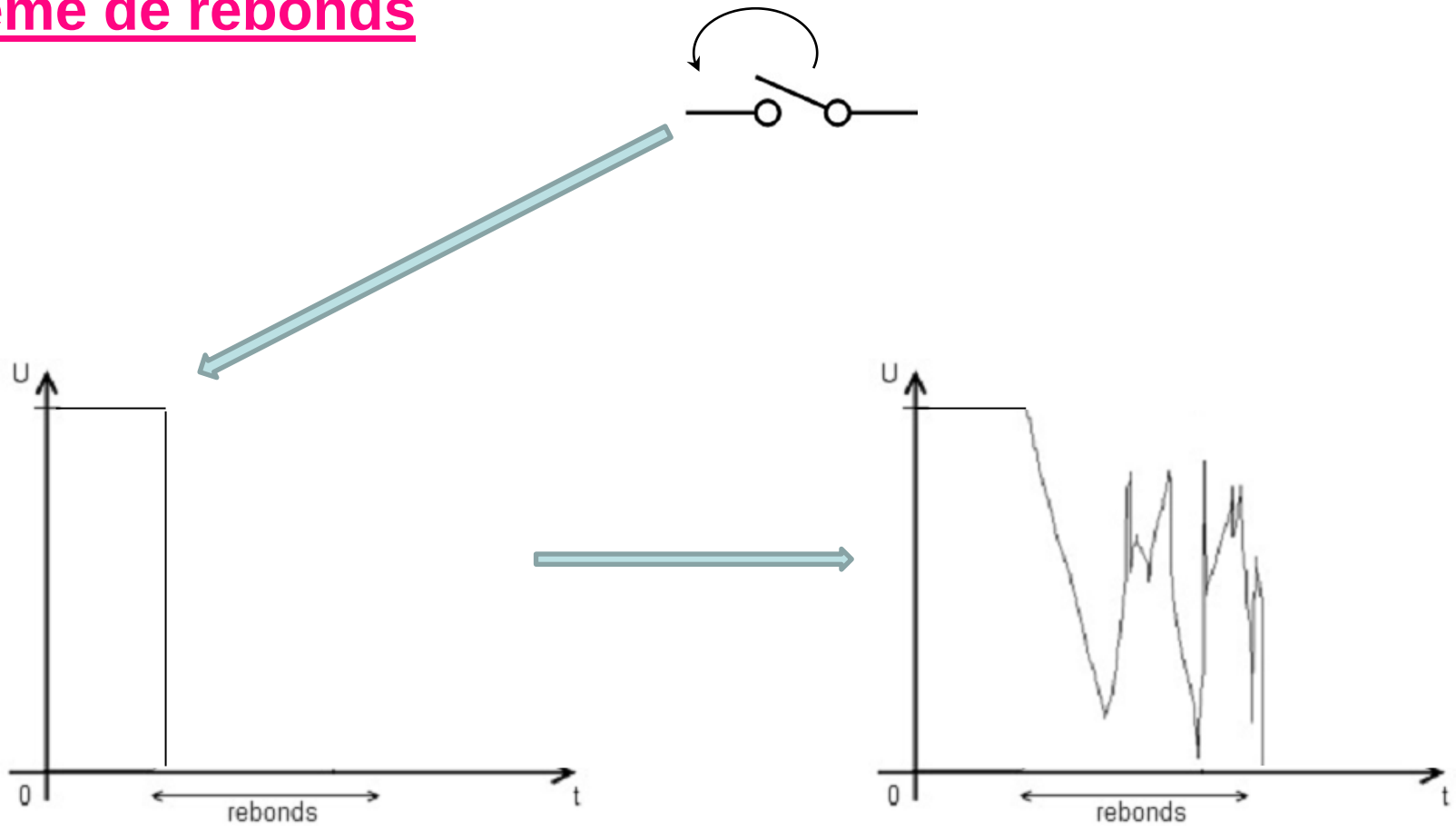
```
void setup(){
    pinMode(LED,OUTPUT);
//la broche 13 est une sortie
    pinMode(BOUTON,INPUT);
//la broche 2 est une entrée
}
```

Programme (suite)

```
void loop(){
  etat_bouton = digitalRead(BOUTON);
      // lit et mémorise l'état en entrée de la pate 2

      if (etat_bouton == LOW){ // si l'entrée 2 est à l'état LOW (bouton appuyé)

          etat_led = 1- etat_led ; // inverse l'état de la led
       }

      if (etat_led == 1) {//si la led doit être allumée
          digitalWrite(LED, LOW); // allumer la LED
       }
      else { // si la led est éteinte
          digitalWrite(LED, HIGH); // éteindre la LED
       }
} // fin de loop()
```

Problème: Fonctionnement hasardeux du bouton => lecture trop rapide des états !! (16MHz)

Problème de rebonds

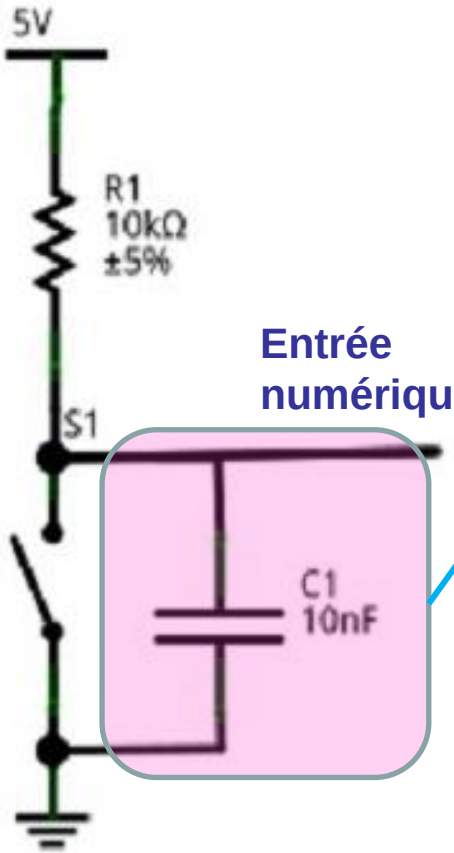


Signal idéal

Signal réel

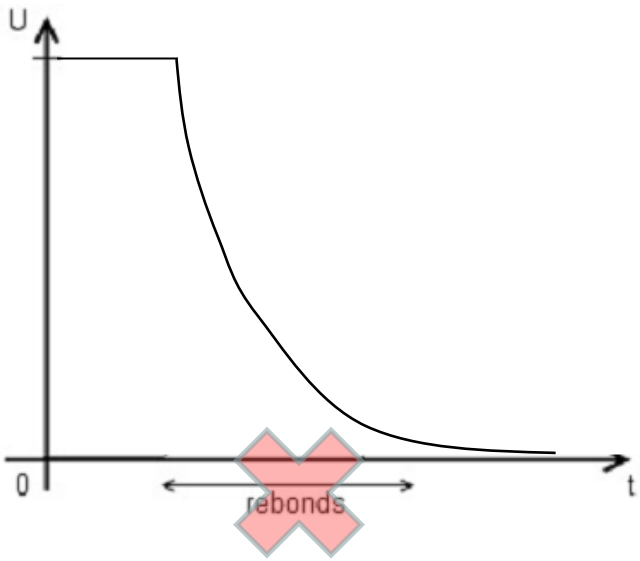
En quelque 10ième de sec => 1000 changements d'états (allumé/éteind) !!

Solution physique



Entrée numérique 2

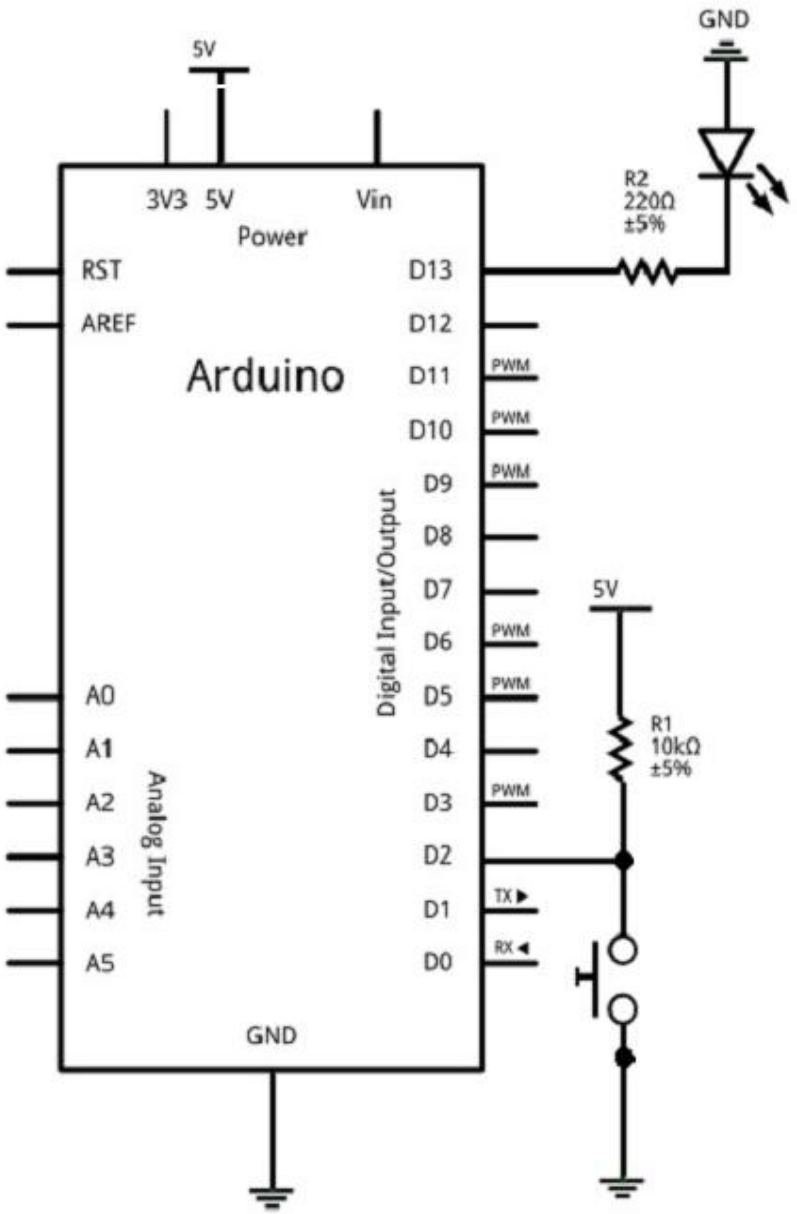
Condensateur en parallèle avec l'interrupteur



Solution numérique

- **Mémoriser l'ancienne valeur de l'état du bouton avant de lire la nouvelle valeur**
  - Comparer l'état actuel du bouton avec la précédente valeur.
- **Patienter ~ 10ms pour éviter les rebond avant d'allumer la LED et d'effectuer une nouvelle lecture du bouton.**

Montage



Programme

```
//Déclaration des varables globales
#define LED 13
//broche 13 du micro-contrôleur se nomme maintenant  LED
#define BOUTON 2
//broche 2 du micro-contrôleur se nomme maintenant  BOUTON
int etat_bouton= 0;
//La variable « etat_bouton » mémorise l'état HIGH ou LOW de la pate d'entrée
int old_etat_bouton= 0;
//La variable « old_etat_bouton » mémorise l'ancien état de la variable « etat_bouton »
int etat_led= 0;
//La variable « etat_led » mémorise l'état 1 (allumée) ou 0 (éteinte) de la led.
```

```
void setup(){
   pinMode(LED,OUTPUT);
//la broche 13 est une sortie
  pinMode(BOUTON,INPUT);
//la broche 2 est une entrée
}
```

Programme

```
void loop(){
  etat_bouton = digitalRead(BOUTON);
      // lit et mémorise l'état en entrée de la pate 2

      if ((etat_bouton == LOW) && (old_etat_bouton == HIGH)){
      // si l'entrée 2 est à l'état LOW (bouton appuyé) et que juste précédemment le
      bouton est ouvert
          etat_led = 1- etat_led ; // inverse l'état de la led
          delay(10); // patienter 10ms pour éviter les rebonds avant d'allumer la led
       }
      old_etat_bouton = etat_bouton; // sauvegarde la nouvelle valeur

      if (etat_led == 1) {//si la led doit être allumée
          digitalWrite(LED, LOW); // allumer la LED
       }
      else { // si la led est éteinte
          digitalWrite(LED, HIGH); // éteindre la LED
       }
} // fin de loop()
```

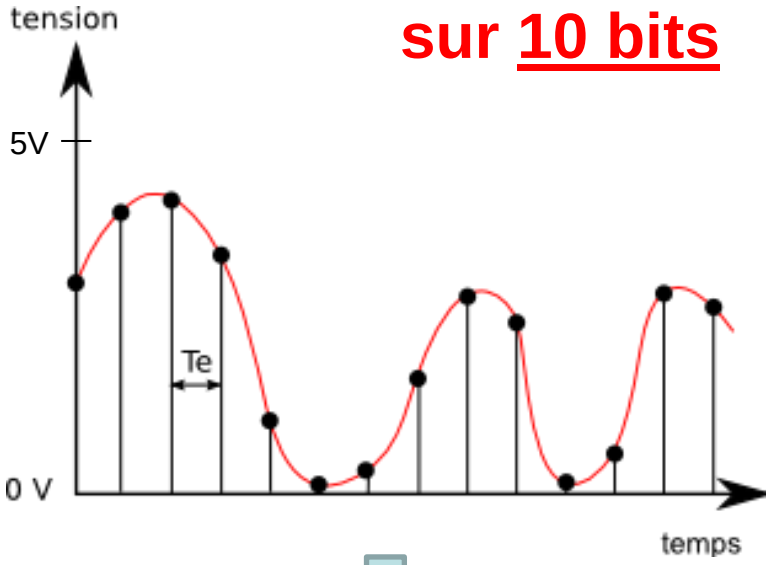
Programme complet

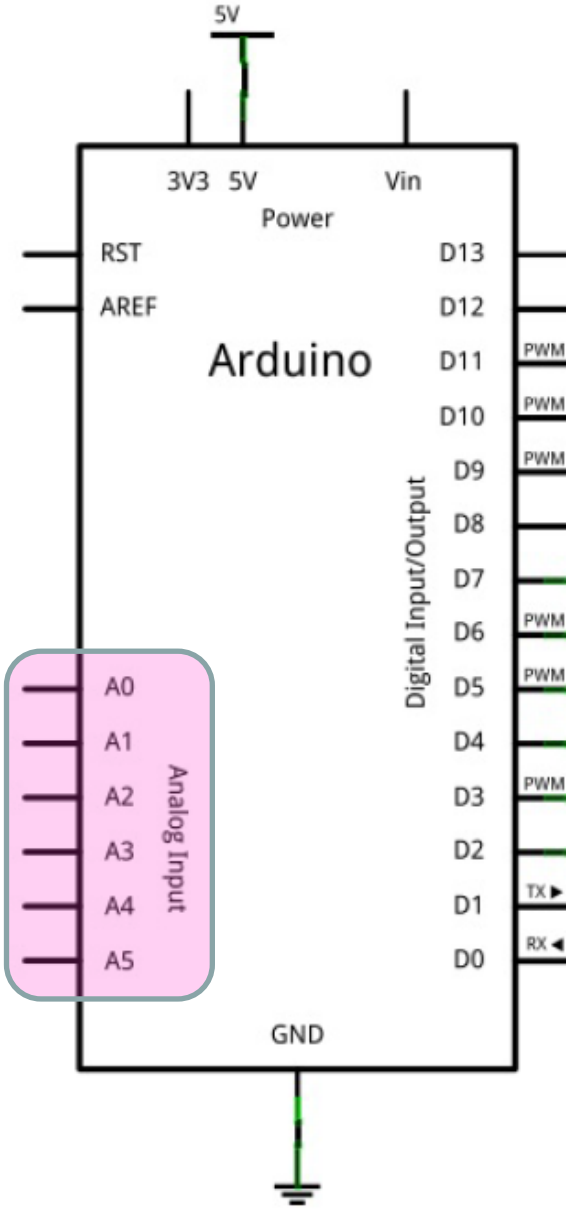
```
//Déclaration des varables globales
#define LED 13  //broche 13 du micro-contrôleur se nomme maintenant  LED
#define BOUTON 2 //broche 2 du micro-contrôleur se nomme maintenant  BOUTON
int etat_bouton= 0; //La variable « etat_bouton » mémorise l'état HIGH ou LOW de la pate d'entrée
int old_etat_bouton= 0; //La variable « old_etat_bouton » mémorise l'ancien état de la variable
« etat_bouton »
int etat_led= 0;
//La variable « etat_led » mémorise l'état 1 (allumée) ou 0 (éteinte) de la led.

void setup(){
   pinMode(LED,OUTPUT);
//la broche 13 est une sortie
 pinMode(BOUTON,INPUT);
//la broche 2 est une entrée
}
void loop(){
  etat_bouton = digitalRead(BOUTON); // lit et mémorise l'état en entrée de la pate 2
        if ((etat_bouton == LOW) && (old_etat_bouton == HIGH)){
        // si l'entrée 2 est à l'état LOW (bouton appuyé) et que juste précédemment le bouton est ouvert
            etat_led = 1- etat_led ; // inverse l'état de la led
             delay(10); // patienter 10ms pour éviter les rebonds avant d'allumer la led
         }
        old_etat_bouton = etat_bouton; // sauvegarde la nouvelle valeur
        if (etat_led == 1) {//si la led doit être allumée
            digitalWrite(LED, LOW); // allumer la LED
         }
        else { // si la led est éteinte
            digitalWrite(LED, HIGH); // éteindre la LED
         }
} // fin de loop()
```

3. Entrées Analogiques

Fréquence d'échantillonnage :
fe = 10kHz

Echantillonnage du signal sur 10 bits

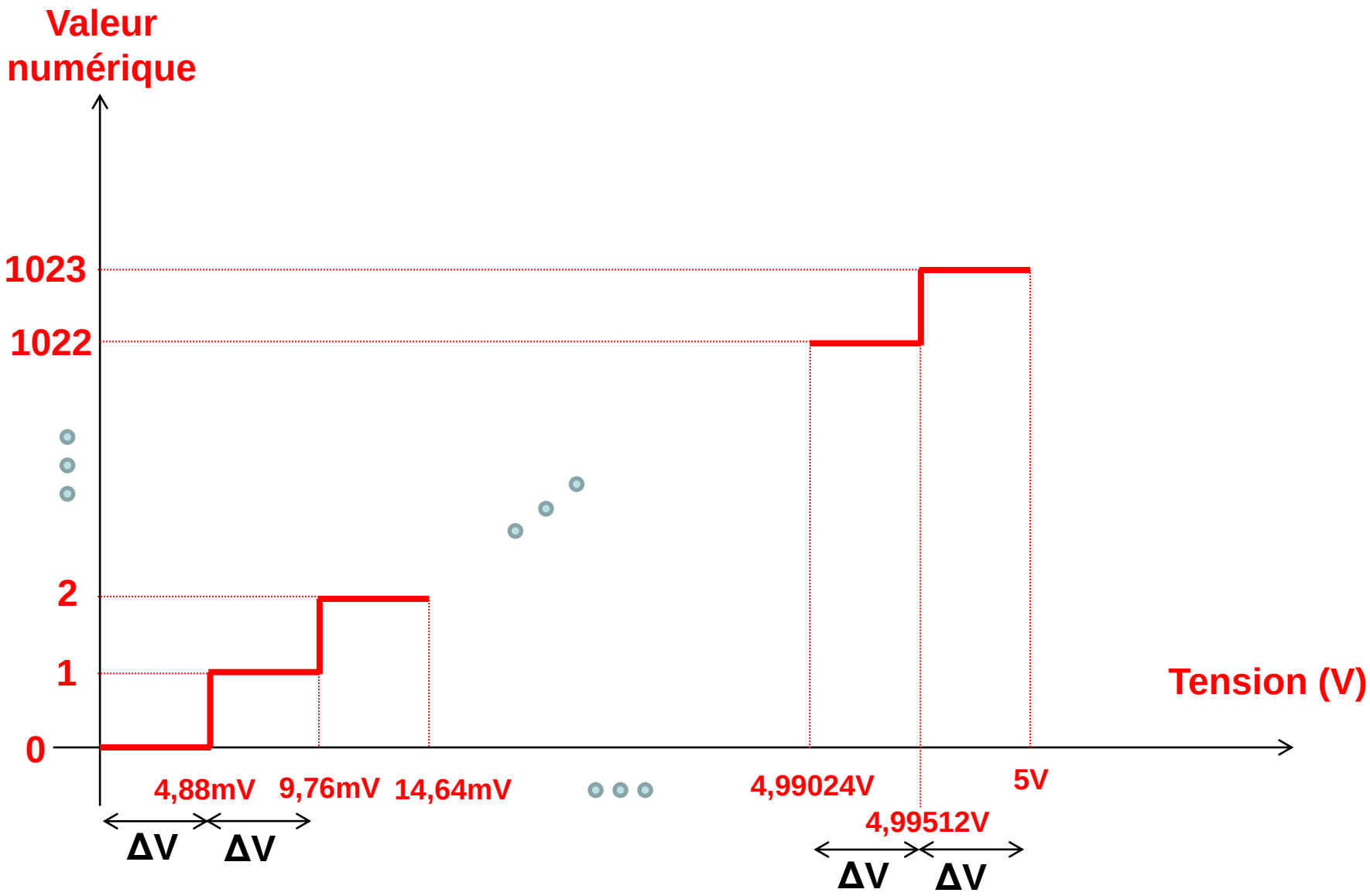




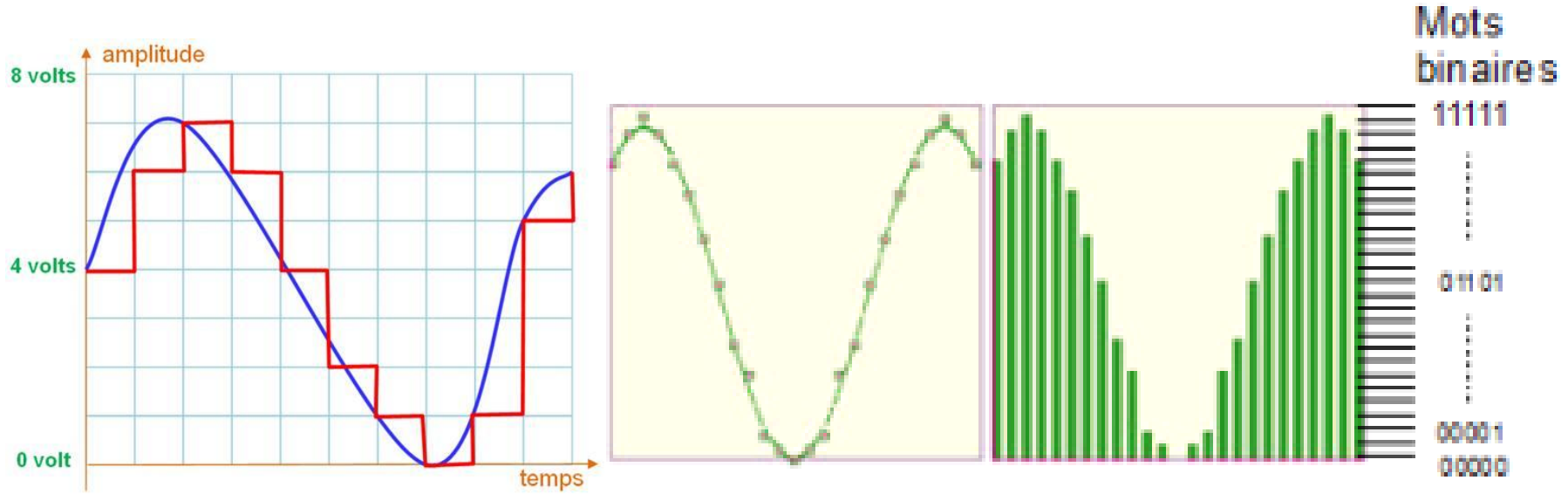
10 bits => 1024 valeurs entre 0 et 5V !!

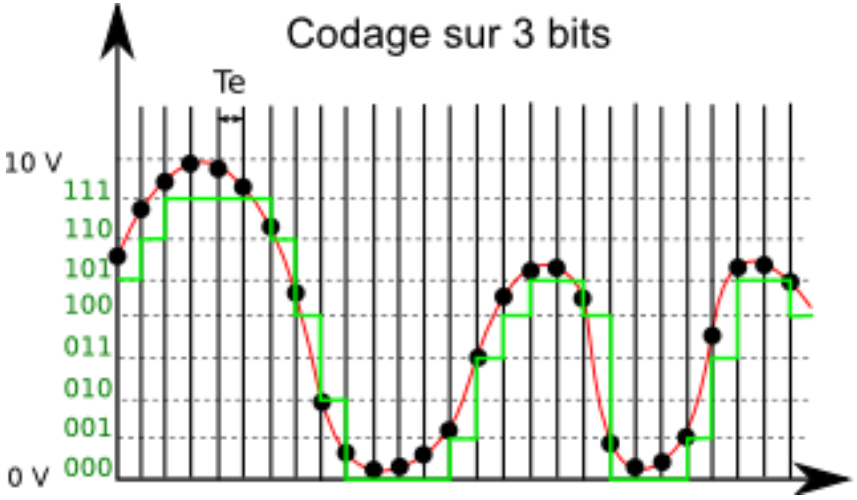
Sensibilité ΔV

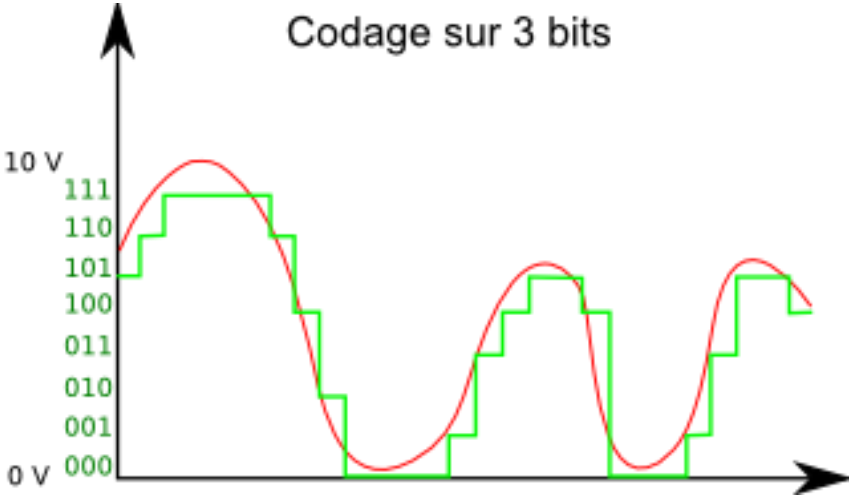
$$\Delta V = \frac{5}{1024} = 0{,}00488\ V = 4{,}88\times10^{-3}\ V = 4{,}88mV$$



Exemples:


amplitude
8 volts
4 volts
0 volt
temps
Mots
binaires
11111
01101
00001
00000


Codage sur 3 bits
Te
10 V
111
110
101
100
011
010
001
0 V
000


Codage sur 3 bits
10 V
111
110
101
100
011
010
001
0 V
000

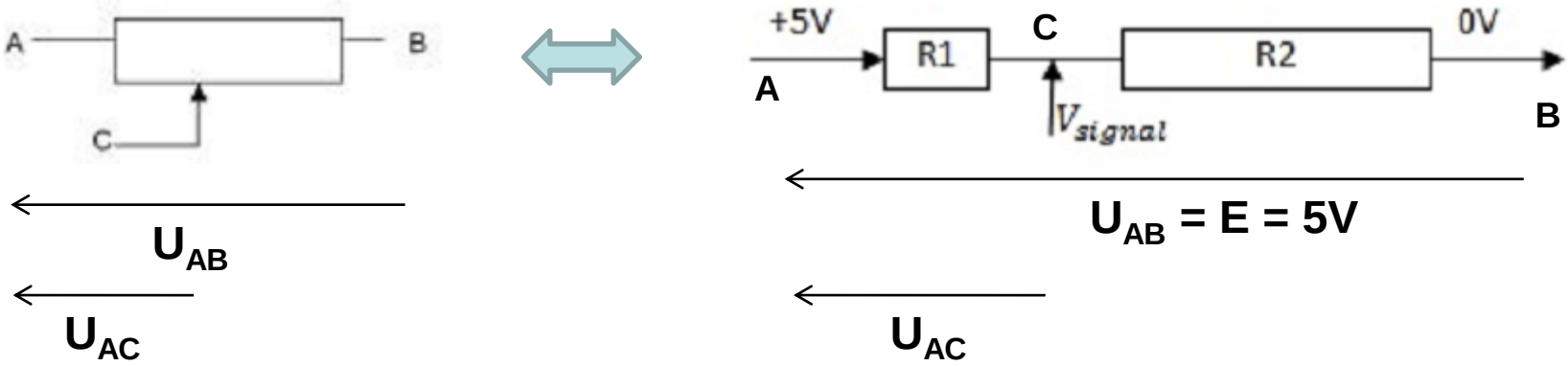
3.a. Utilisation d'un potentiomètre

Objectif 1:

faire varier la période de clignotement d'une LED à l'aide d'une résistance variable (rhéostat / potentiomètre).

Utiliser la fonction **analogRead()** sur une broche analogique.

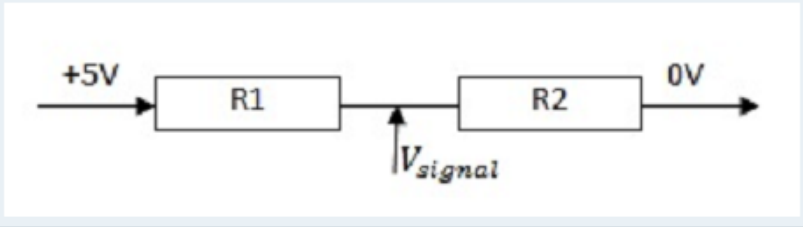
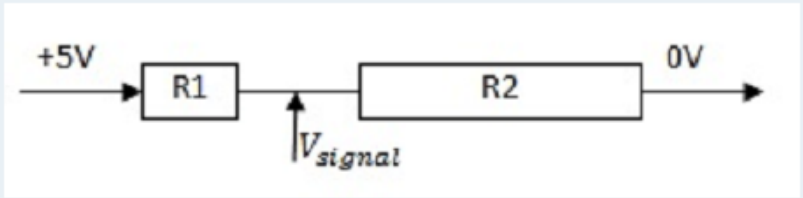
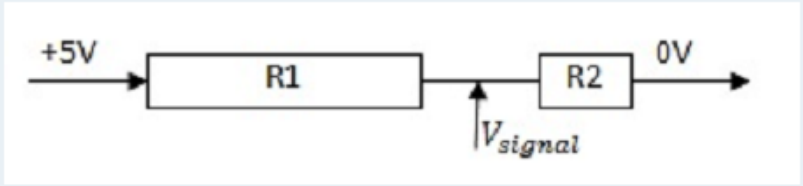
Principe:



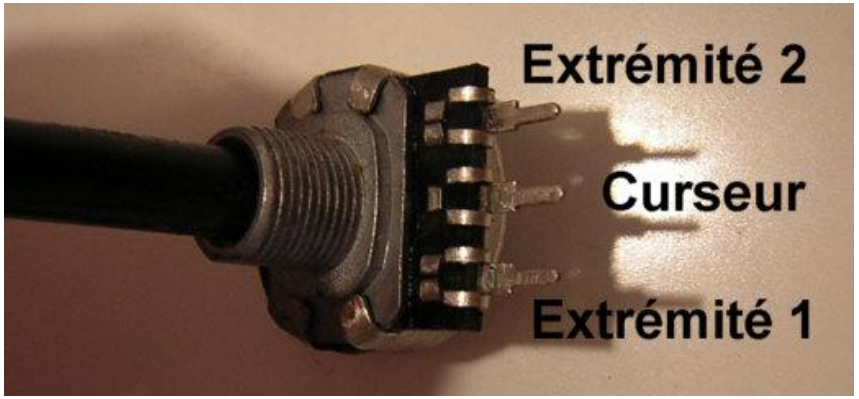
Pont diviseur de tension:

$$U_{AC} = \frac{R_1}{R_1 + R_2} U_{AB} = \frac{R_1}{R_1 + R_2} E$$

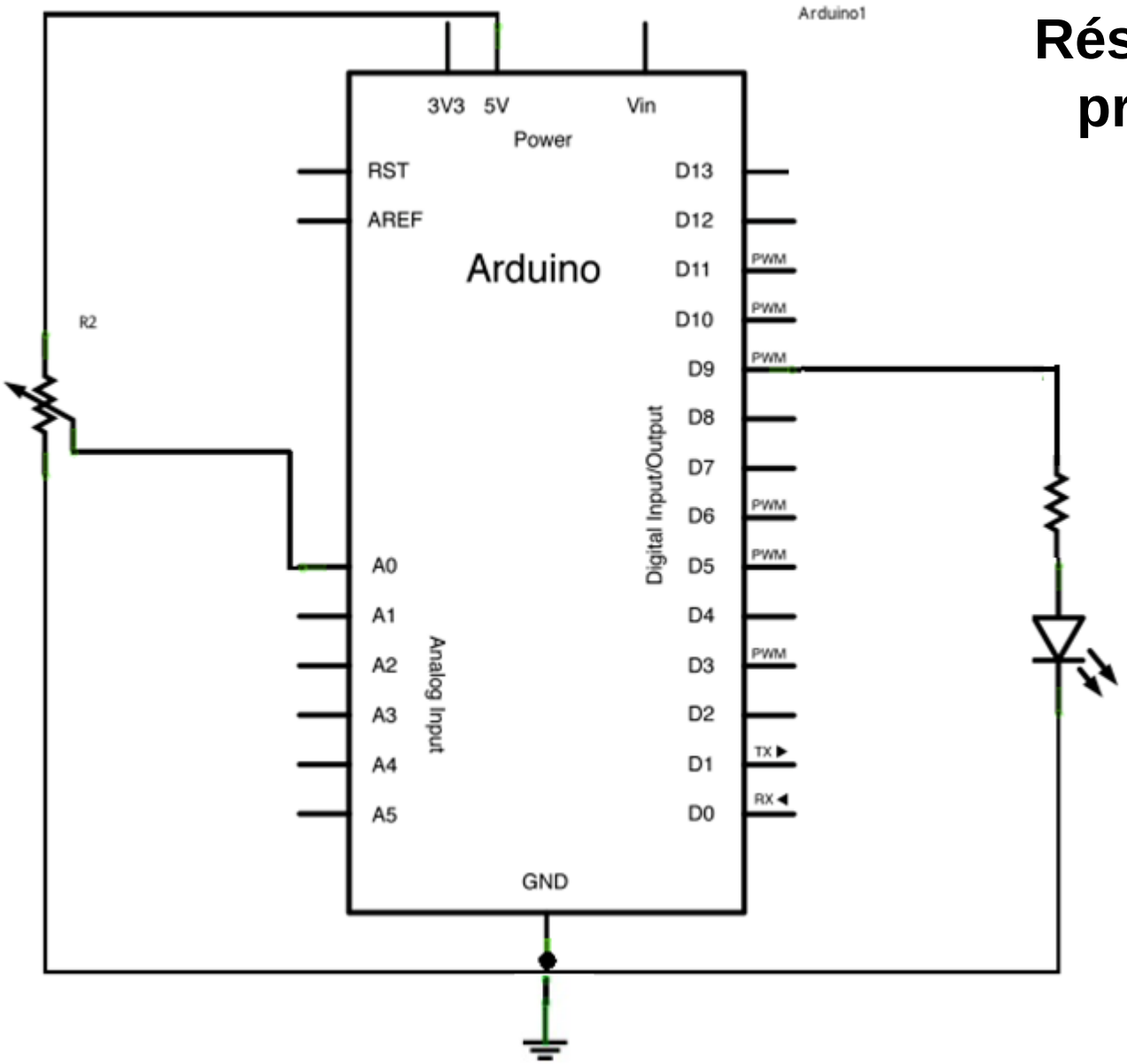
Exemple:

Schéma équivalent	Position du curseur	Tension sur la broche C
	Curseur à **la moitié**	$V_{signal} = (1 - \frac{50}{100}) \times 5 = 2.5V$
	Curseur à **25% du départ**	$V_{signal} = (1 - \frac{25}{100}) \times 5 = 3.75V$
	Curseur à **75% du départ**	$V_{signal} = (1 - \frac{75}{100}) \times 5 = 1.25V$

Composant:

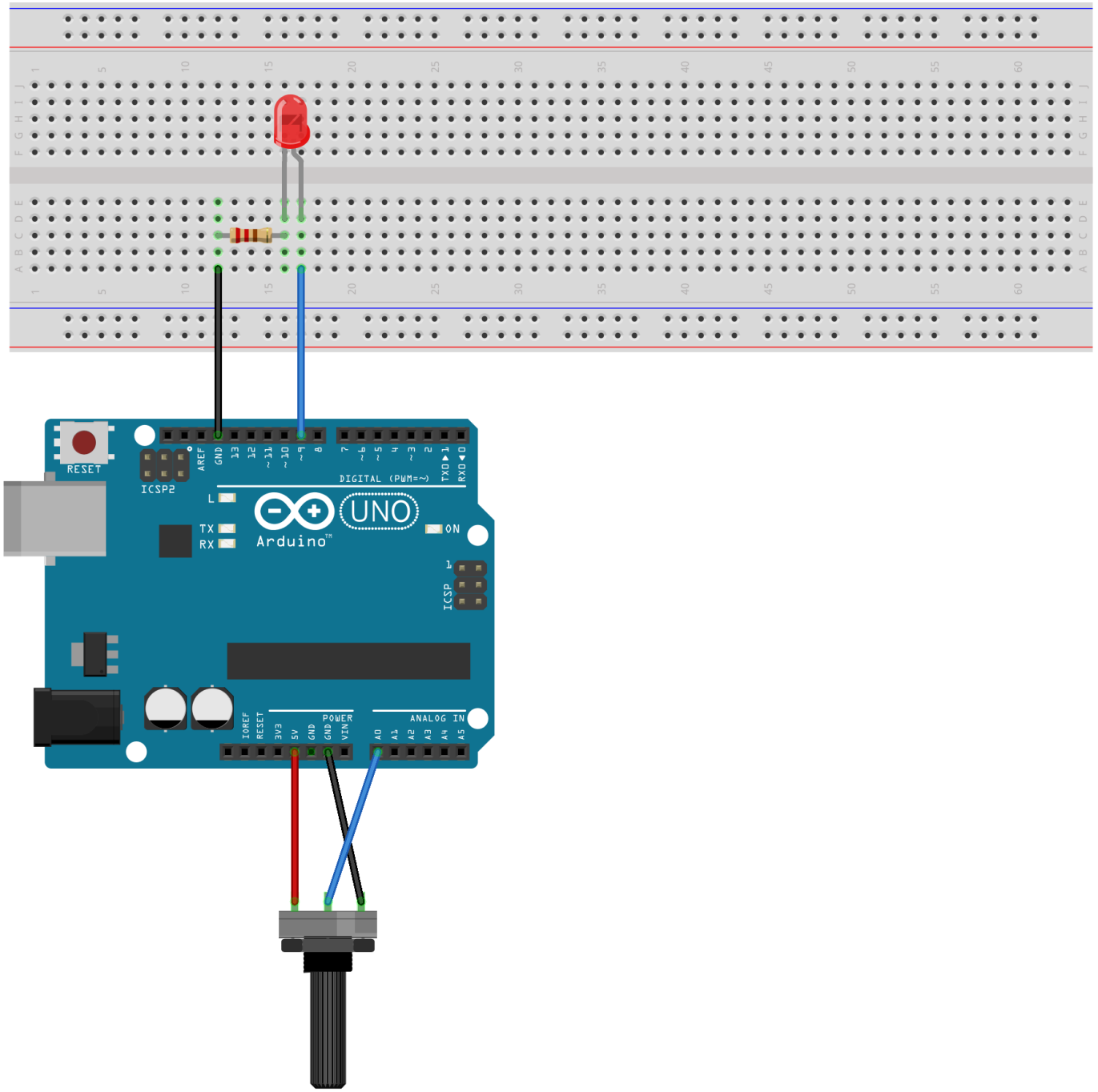


Montage



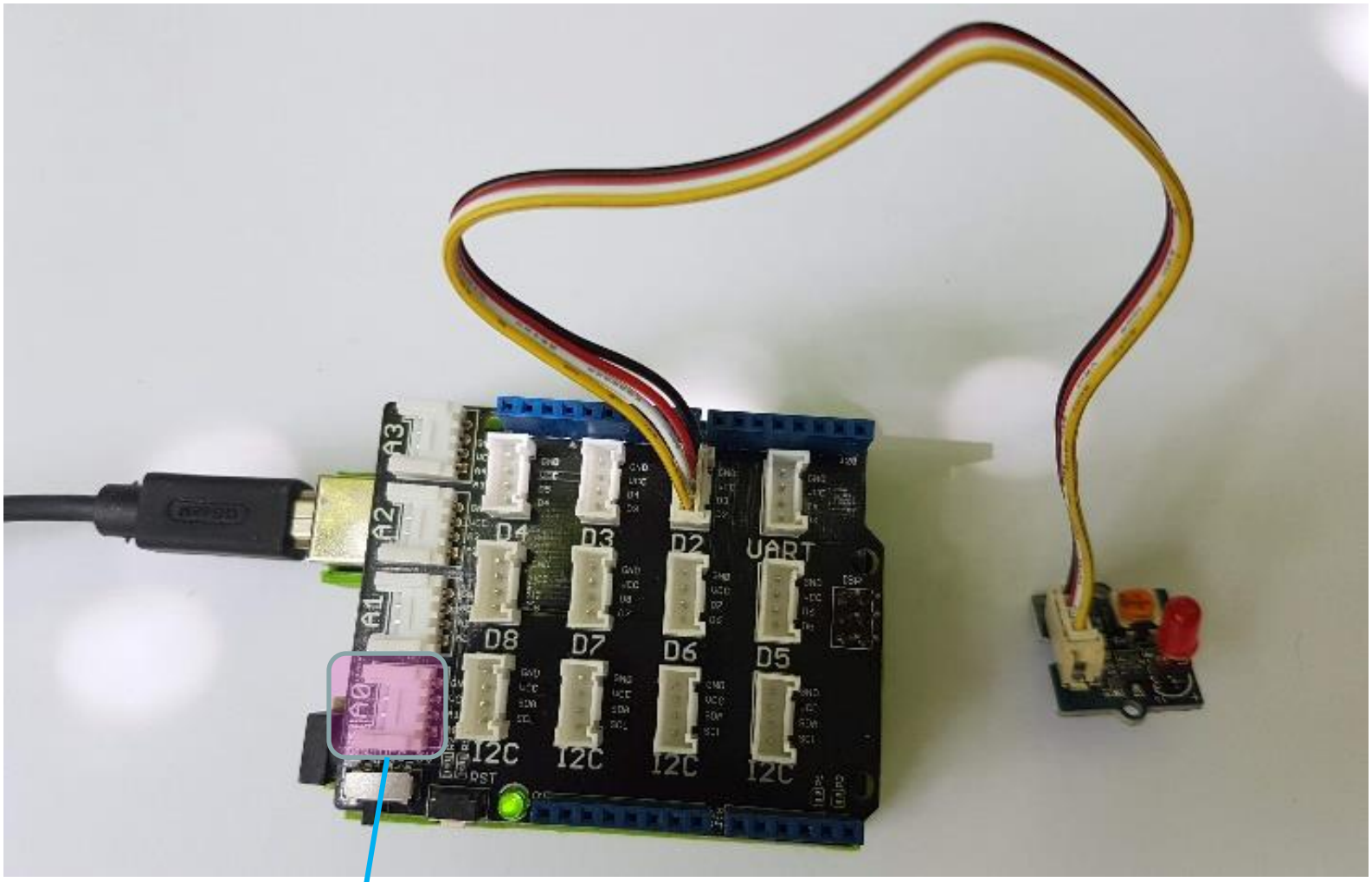
Résistance de protection: 220 Ω

Montage


RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino™
TX
RX
ON
ICSP
POWER
ANALOG IN
fritzing

Montage Groove:

Brancher 1 LED Groove sur D3 (PWM~)



Brancher le potentiomètre sur A0

Algorithme

- **Déclaration des variables globales:**
  - **valeurCapteur**
  - **LedPin et capteurPin**
- **Initialisation des variables et des paramètres de contrôles - setup():**
  - **Définir D13 comme une sortie numérique**
  - **Définir A0 comme une entrée analogique**
- **Boucle de contrôle - loop():**
  - **Lire la valeur de la tension à la pate A0 et mémoriser cette valeur dans la variable « valeurCapteur ».**
  - **Allumer la LED**
  - **Patienter un temps équivalent à « valeurCapteur »**
  - **Eteindre la LED**

Programme

```
int sensorPin = A0;  // A0 branchée à la pate du potentiomètre
int ledPin = 9;      // D9 pate de la LED
int sensorValue = 0;  // variable de mémorisation des valeurs provenant du potentiomètre
void setup() {
  // délare ledPin comme une sortie numérique
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Lecture des valeurs provenant du potentiomètre
  sensorValue = analogRead (sensorPin);
  // Allume la led
  digitalWrite(ledPin, HIGH);
  // patienter <sensorValue> millisecondes:
  delay (sensorValue);
  // éteindre la LED
  digitalWrite(ledPin, LOW);
  // patienter <sensorValue> millisecondes:
  delay (sensorValue);
}
```

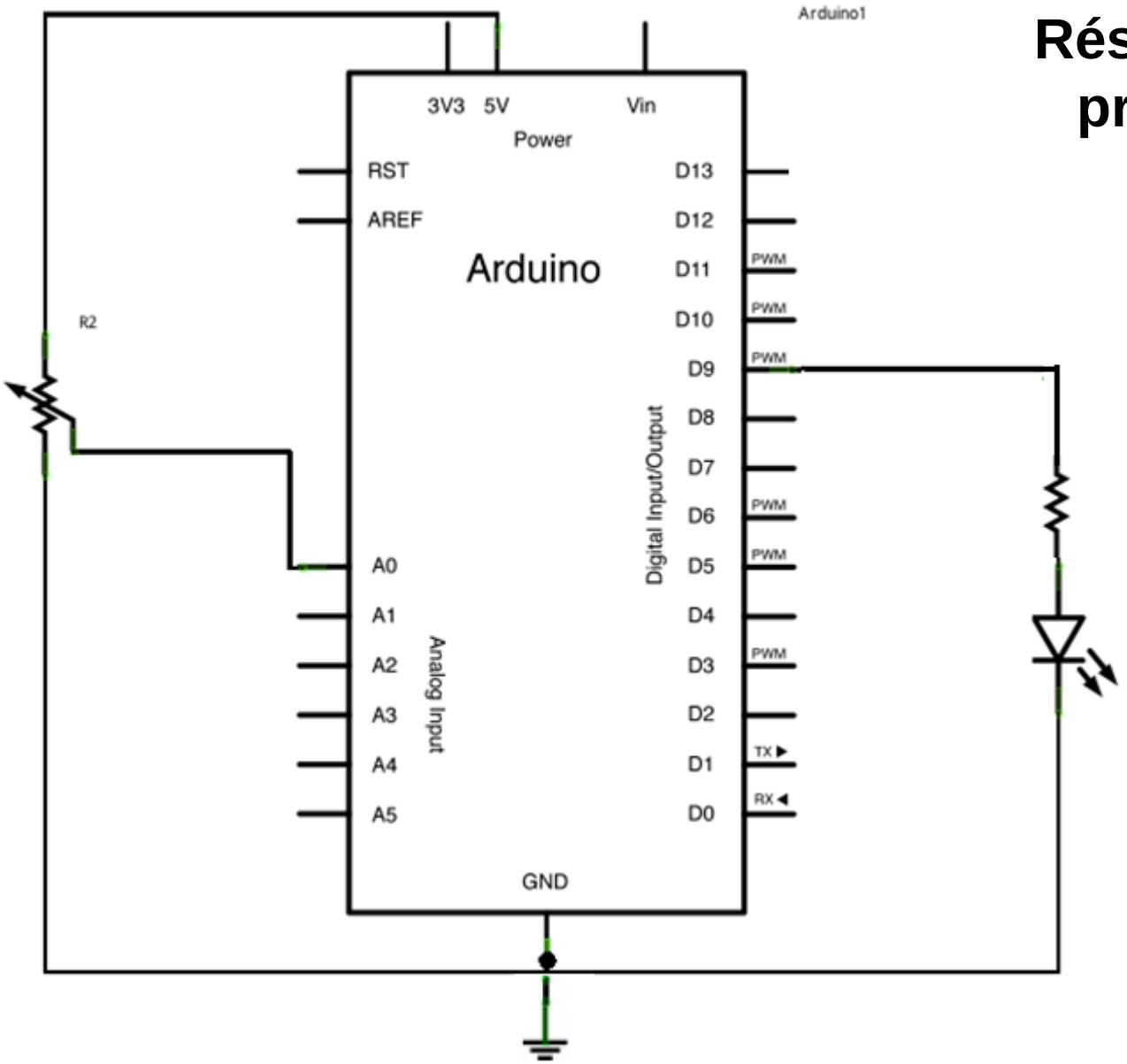
3.b. Utilisation d'un potentiomètre

Objectif 2:

Allumer une LED pour un seuil de tension particulier réglé avec un potentiomètre.

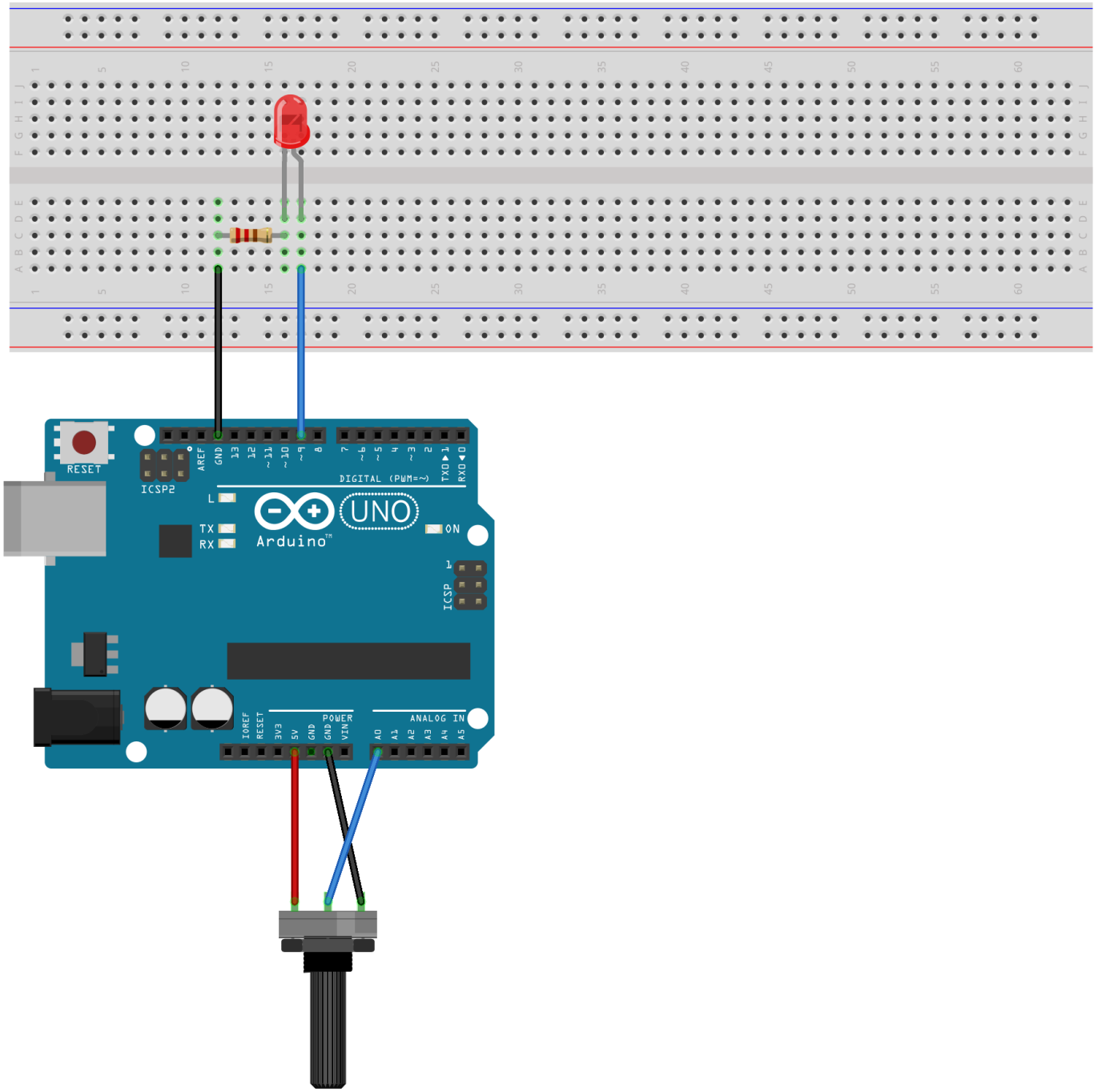
Montage identique!

Montage



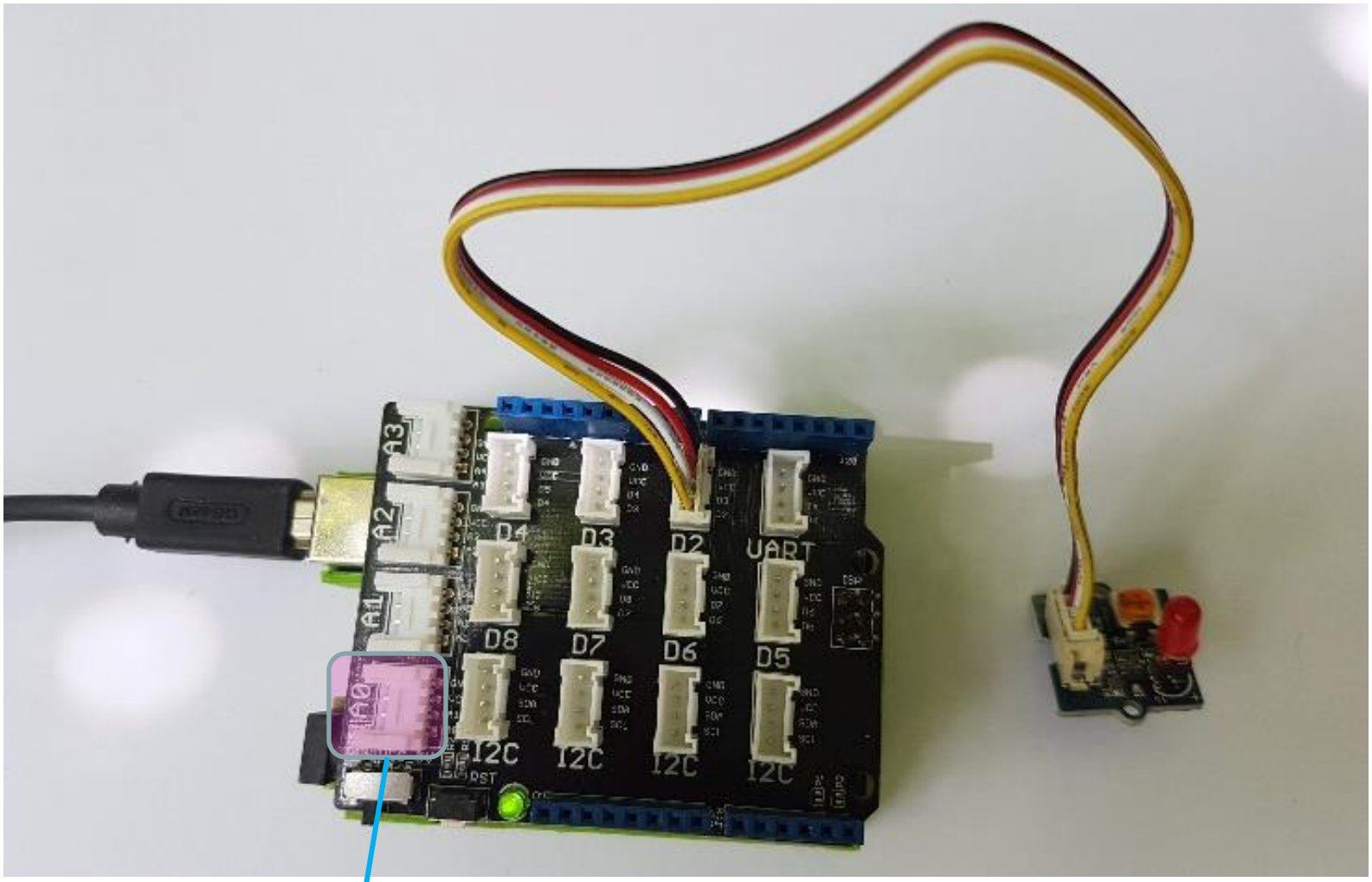
Résistance de protection: 220 Ω

Montage


RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino™
TX
RX
ON
ICSP
POWER
ANALOG IN
fritzing

Montage Groove:

Brancher 1 LED Groove sur D3 (PWM~)



Brancher le potentiomètre sur A0

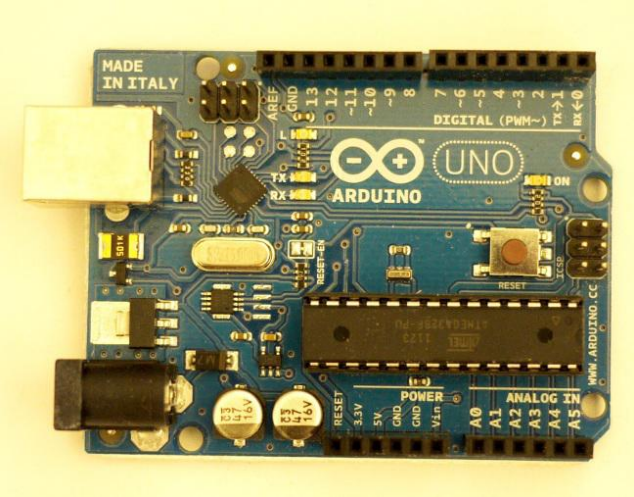
Algorithme

- **Déclaration des variables globales:**
  - **sensorValue, tensionSeuil et tension**
  - **LedPin et capteurPin**
- **Initialisation des variables et des paramètres de contrôles - setup():**
  - **Définir D13 comme une sortie numérique**
  - **Définir A0 comme une entrée analogique**
- **Boucle de contrôle - loop():**
  - **Lire la valeur de la tension à la pate A0 et mémoriser cette valeur dans la variable « sensorValue ».**
  - **Convertir sensorValue en sa tension correspondante**
  - **Tester le seuil de tension:**
    - **Si tension > tensionSeuil : allumer la LED**
    - **Sinon : éteindre la LED**

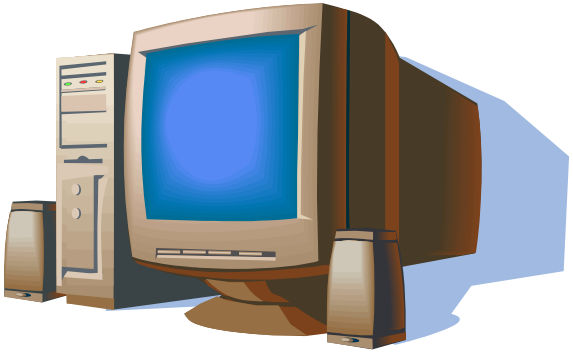
Programme

```
/////////////// variables globales
int sensorPin = A0;  // A0 branchée à la pate du potentiomètre
int ledPin = 9;      // D13 pate de la LED ////// ATTENTION :  MONTAGE GROOVE : D2
int sensorValue = 0;  // variable de mémorisation des valeurs provenant du potentiomètre
float tensionSeuil = 2.5; // tension seuil à partir de laquelle on allume la led ---- A modifier
float tension = 0; // tension mesurée

void setup() {
  // délare ledPin comme une sortie numérique
  pinMode (ledPin, OUTPUT);
}

void loop() {
  // Lecture des valeurs provenant du potentiomètre
  sensorValue = analogRead(sensorPin);
  // calcul de la tension mesurée correspondante
  tension = sensorValue * 5 / 1024;

  //test de la tension seuil
  if (sensorValue > tensionSeuil) {
    // Allumer la led
    digitalWrite(ledPin, HIGH);
  }
  else {
    // éteindre la LED
    digitalWrite(ledPin, LOW);
  }

  //attendre 2s avant de refaire une mesure
  delay(100);
}
```

6 Lire sur l'ordinateur des données envoyées par la carte Arduino

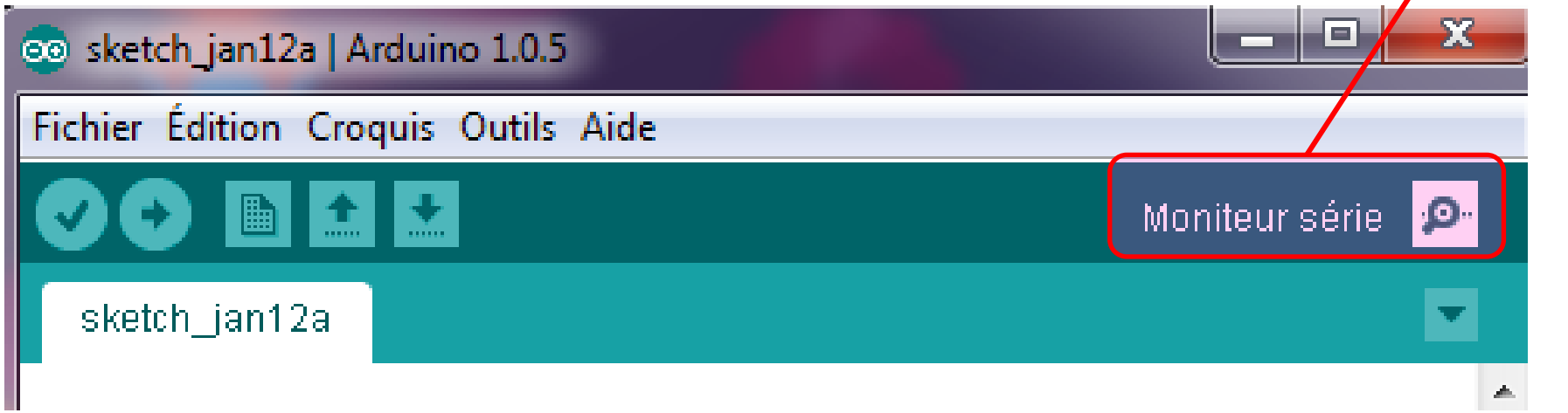


110010101110

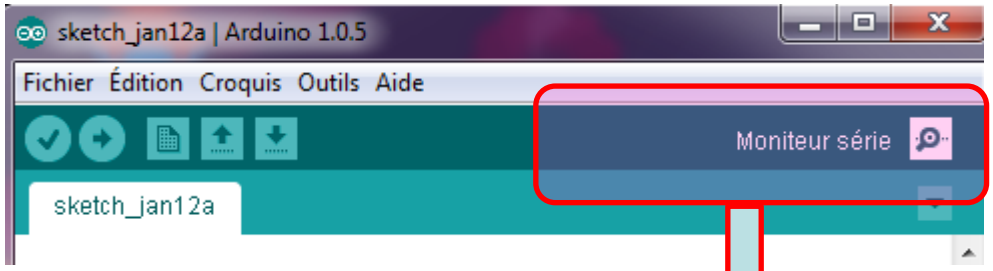


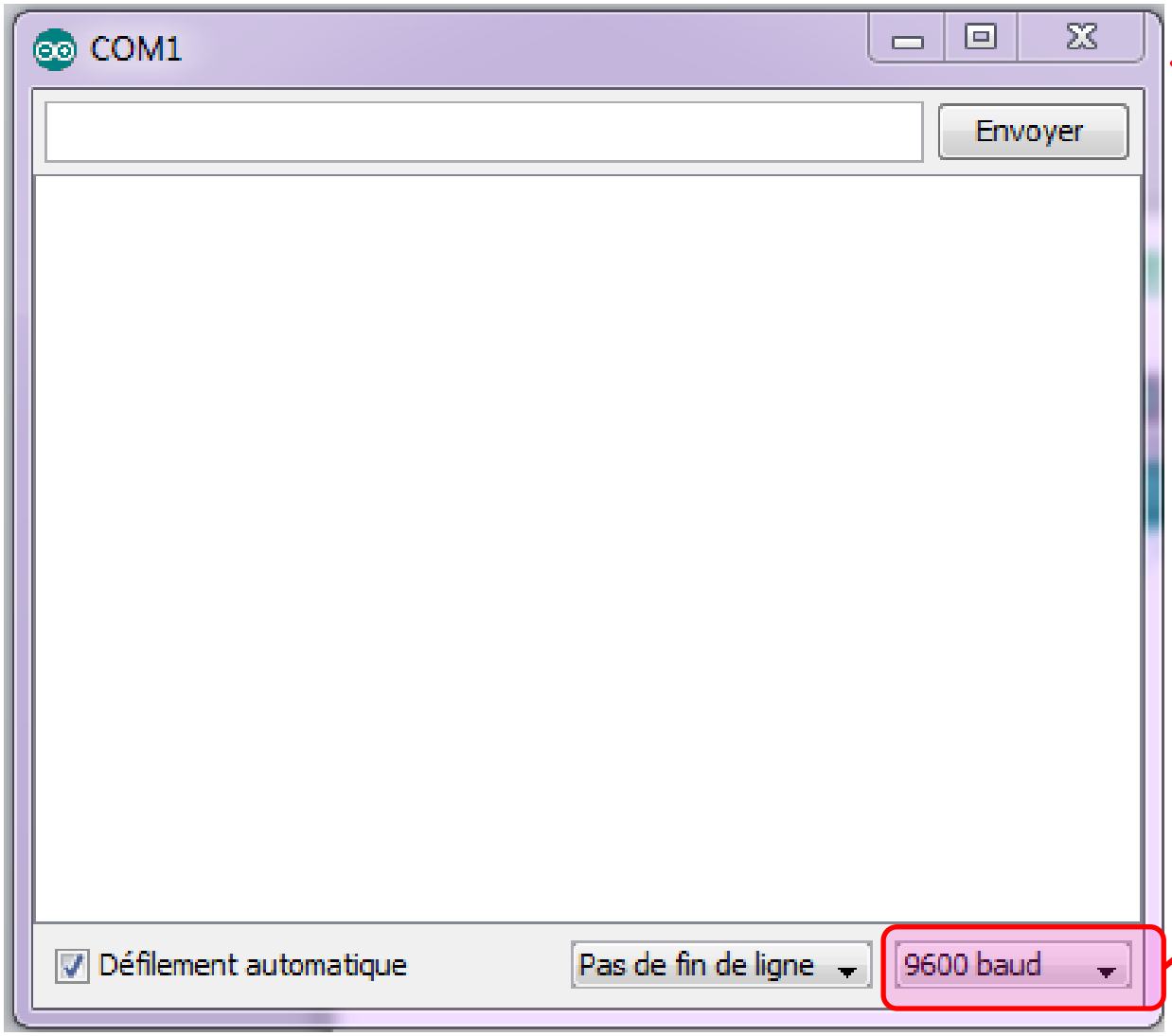
Transfert de données vers l'ordinateur

Lecture des données via l'IDE



Terminal Série





Vitesse de transfert des données : Ici 1 baud = 1 octet/s

Programmation de la communication : Objet « SERIAL »

Attributs ⇐ Classe « Serial » ⇒ Méthodes

- Vitesse
- Bits de données
- …

Voir l'aide en ligne

https://www.arduino.cc/reference/en/language/functions/communication/serial/

- if (Serial)
- available()
- begin()
- end()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- setTimeout()
- write()
- serialEvent()

1 Initialisation du port dans le setup()

↳ **Méthode : begin()**

↳ **Précision de la vitesse de transfert : 9600 bauds**

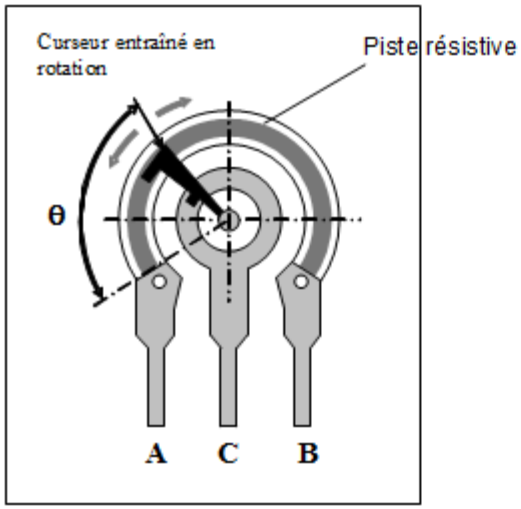
2 Ecriture des données sur le port

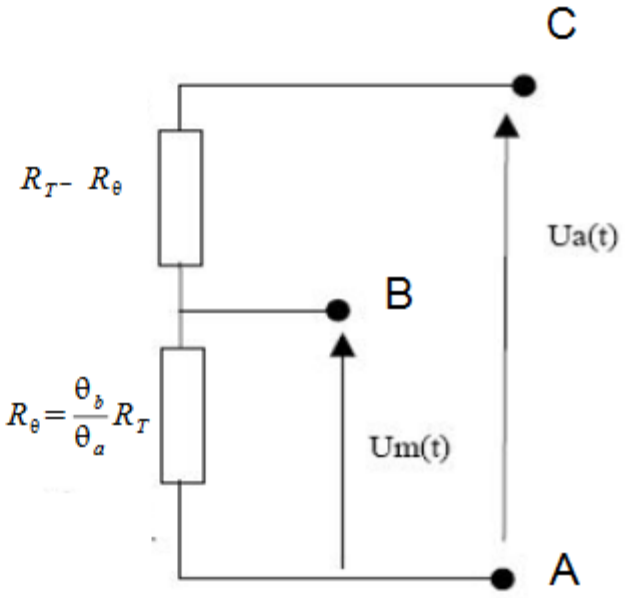
↳ **Méthode : println() ou print()**

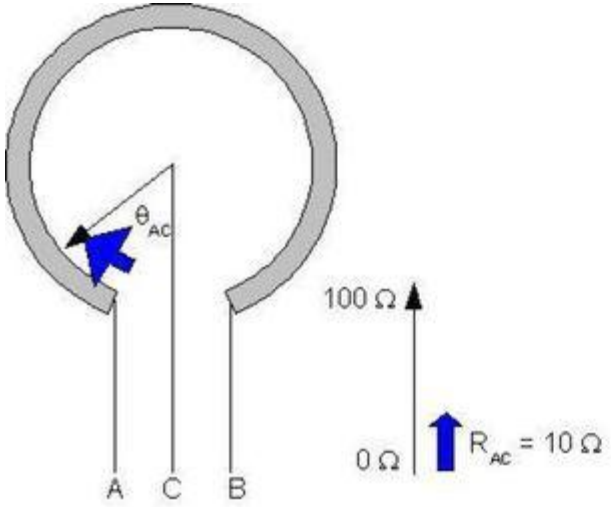
Modification programme précédent

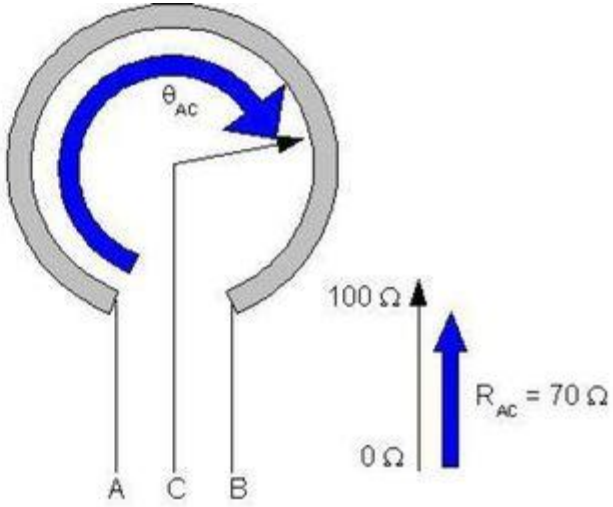
```
//////////////// variables globales
int sensorPin = A0;  // A0 branchée à la pate du potentiomètre
int ledPin = 13;      // D13 pate de la LED ////// ATTENTION :  MONTAGE GROOVE : D2
int sensorValue = 0;  // variable de mémorisation des valeurs provenant du potentiomètre
float tensionSeuil = 2.5; // tension seuil à partir de laquelle on allume la led ---- A modifier
float tension = 0; // tension mesurée

void setup() {
  // délare ledPin comme une sortie numérique
  pinMode (ledPin, OUTPUT);
// démarrer la communication série à 9600 bauds (valeur par défaut)
  Serial.begin(9600);
}

void loop() {
  // Lecture des valeurs provenant du potentiomètre
  sensorValue = analogRead(sensorPin);
  // calcul de la tension mesurée correspondante
  tension = sensorValue * 5 / 1024;

  //// Communication : envoie des données sur le port série
  Serial.print("valeur analogique : ");
  Serial.print(sensorValue);
  Serial.print("\t");
  Serial.print("valeur de la tension : ");
  Serial.print(sensorValue);
  Serial.println (" V"); // retour à la ligne

  //test de la tension seuil
  if (sensorValue > tensionSeuil) {
    // Allumer la led
    digitalWrite(ledPin, HIGH);
  }
  else {
    // éteindre la LED
    digitalWrite(ledPin, LOW);
  }

  //attendre 2s avant de refaire une mesure
  delay(100);
}
```

6 Afficher l'angle d'un potentiomètre







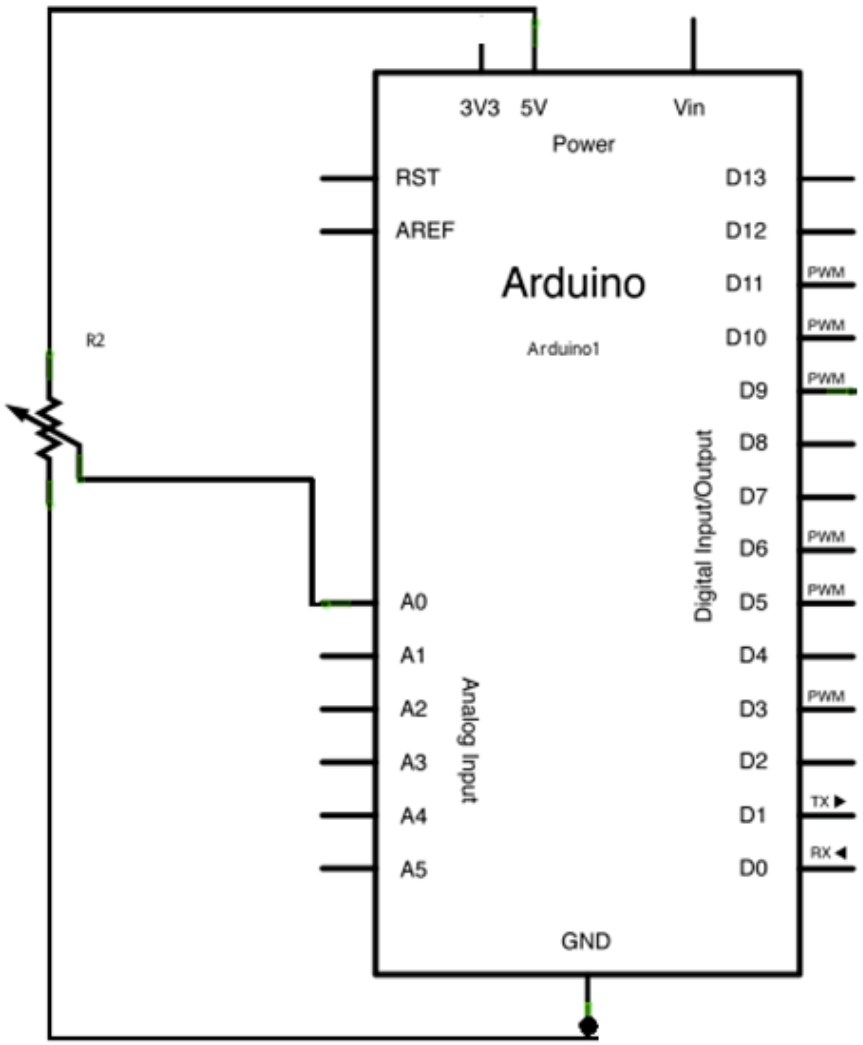


6 Afficher l'angle d'un potentiomètre

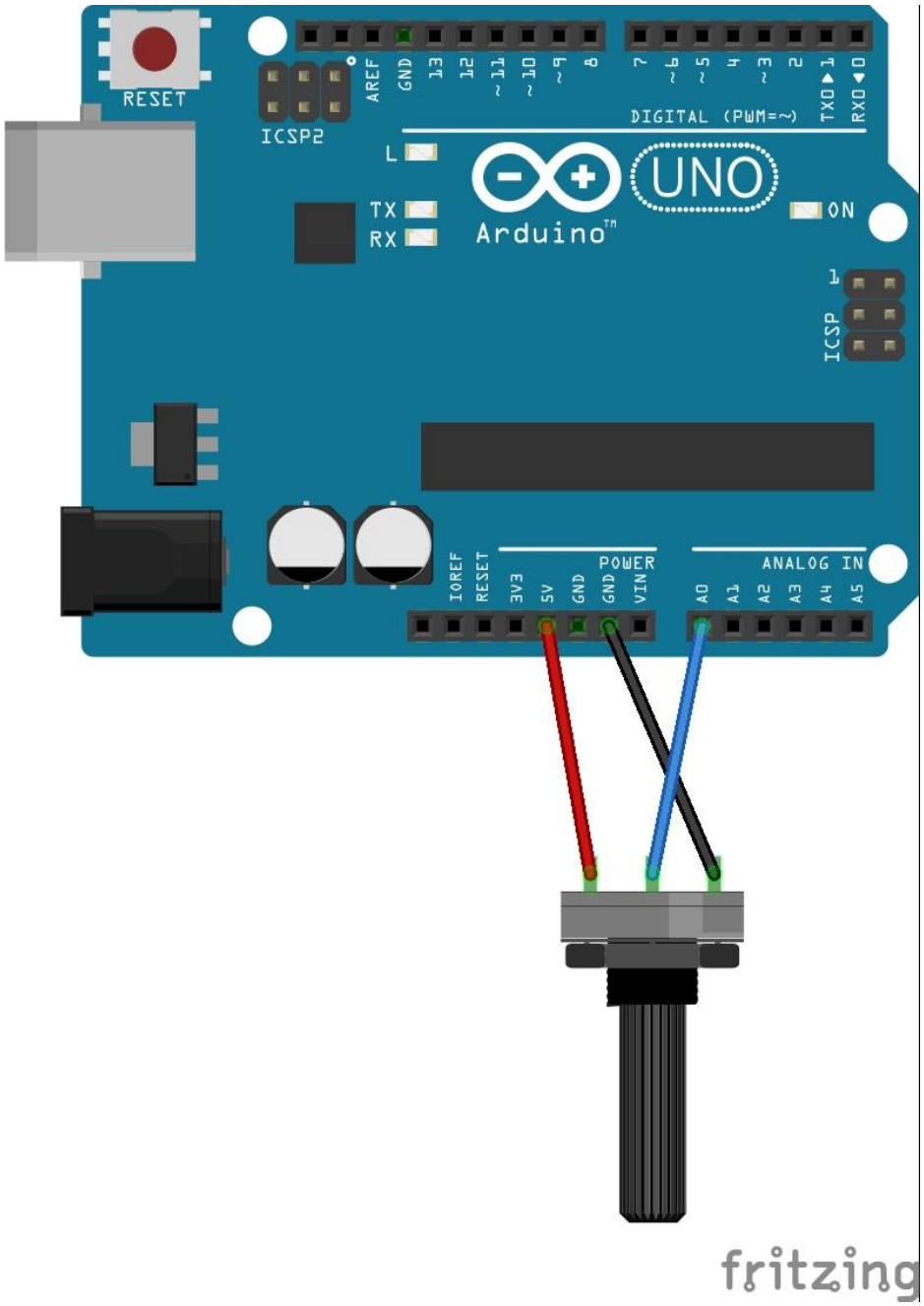
Objectif :

Afficher l'angle d'un potentiomètre grâce au montage potentiométrique

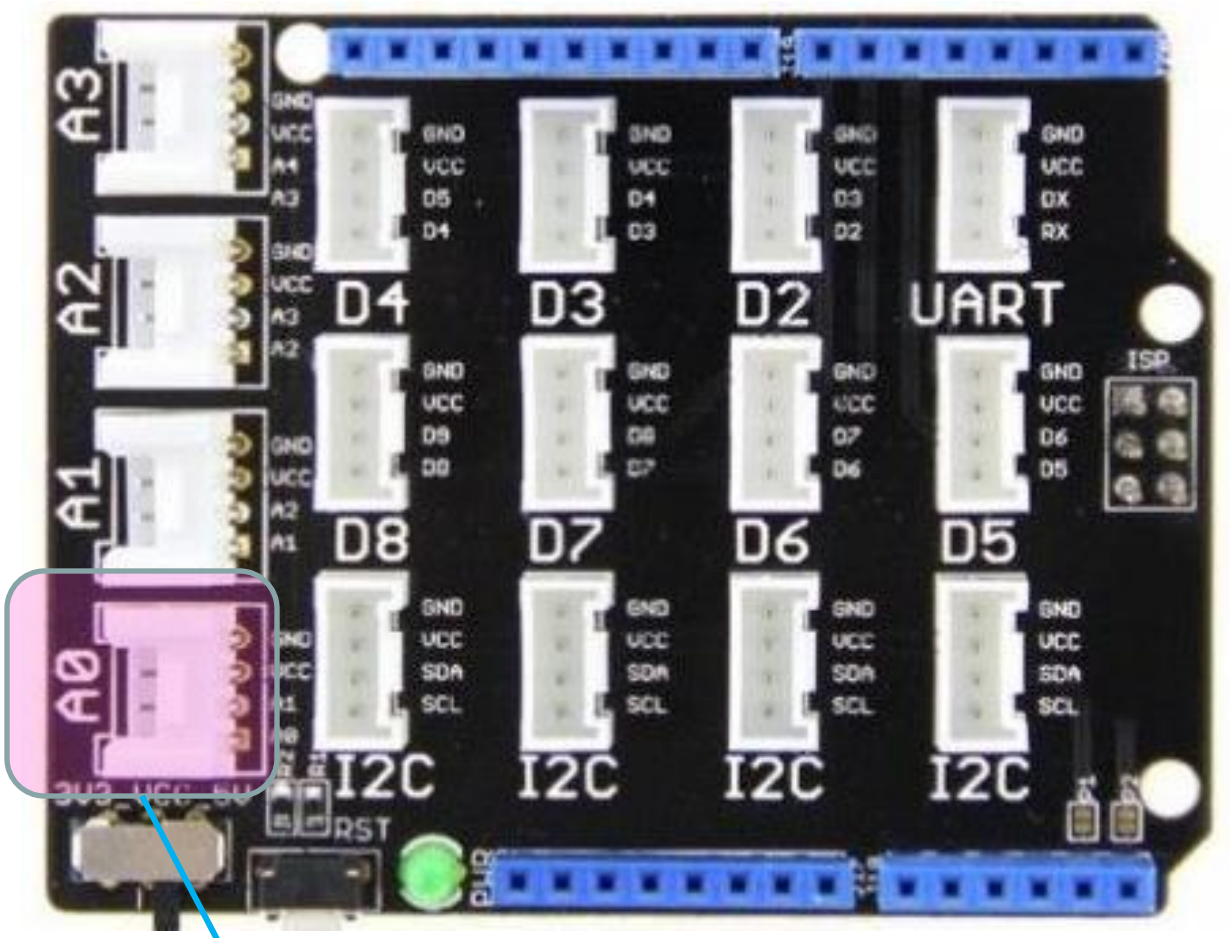
Montage


3V3
5V
Vin
Power
RST
AREF
Arduino
Arduino1
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
PWM
PWM
PWM
PWM
PWM
TX
RX
Digital Input/Output
R2
A0
A1
A2
A3
A4
A5
Analog Input
GND

Montage


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing

Montage Groove:



Brancher le potentiomètre sur A0

Algorithme

- **Déclaration des variables globales**
- **Initialisation des variables et des paramètres de contrôles - setup():**
  - **Définir A0 comme une entrée analogique**
- **Boucle de contrôle - loop():**
  - **Lire la valeur de la tension à la pate A0 et mémoriser cette valeur dans la variable « sensorValue ».**
  - **Convertir sensorValue en sa tension correspondante**
  - **Calculer la résistance correspondante**
  - **Calculer l'angle (modèle linéaire)**
  - **Envoyer la valeur de l'angle à la console SSI celui-ci a changé**

Programme

Potentiometre_5_Mesure_Angle.ino (télécharger)

```
/*****************************************************************
 *  Modèle pour mesure et modélisation de potentiometre linéaire
 *  pour mesurer un angle
 *    v1 : Olivier Boesch (c) 2019
 *****************************************************************/

/*************************** Paramètres à adapter au besoin du professeur *******/
// Broche de l'entrée analogique utilisée
int capteurPin = A0;
// valeur de la résistance totale du potentiomètre
float R = 50000.0;
/*******************************************************************************/
int valprev = -1; // initialisation de la valeur lue précédemment à -1

void setup() {
  // démarrer la communication série à 9600 bauds (valeur par défaut)
  Serial.begin(9600);
}
```

```
void loop() {
  //valeur du capteur de 0 à 1023
  int valeurCapteur;
  // tension calculée en V (de 0 à 5V)
  float tension;
  //résistance calculée en Ohm
  float resistance;
  //angle en degrés calculé
  float angle;

  // lecture capteur
  valeurCapteur = analogRead (capteurPin);
  // conversion en tension
  tension = valeurCapteur/1023.0*5.0; //valeur de 0 à 1023, tension de 0 à 5V
  //Calculer la résistance correspondant à la tension mesurée
  resistance = R*(1-tension/5.0);
  // résistance vers température
  /************************ Modele ********************************/
  /* potentiomètre linéaire
     1 segment de droite
     changer les valeurs de resistance et inscrire le modele correspondant */
    angle = resistance*300/50000.0;
  /*****************************************************************/
  //envoi de l'angle avec son unité
  if( valeurCapteur != (valprev+1) ){
    Serial.print (valeurCapteur);
    Serial.print ("\t");
    Serial.print(tension);
    Serial.print (" V\t");
    Serial.print (resistance);
    Serial.print (" Ohm\t");
    Serial.print (angle);
    Serial.println (" °");
    valprev = valeurCapteur;
  }
  //attendre 2s avant de refaire une mesure
  delay (100);
```

6 Afficher la température mesurée à l'aide d'une CTN

Objectif :

Afficher la température mesurée à l'aide d'un capteur CTN (Coefficient à Température Négative) placé dans un pont diviseur de tension

Capteur CTN ⇨ Thermistance

R diminue lorsque la température augmente, et inversement.

Liens de références

- https://fr.wikipedia.org/wiki/Thermistance
- https://fr.wikipedia.org/wiki/Relation_de_Steinhart-Hart
- http://fablab-robert-houdin.org/wiki/doku.php?id=les_sondes_de_temperature_thermistances_ctn

Modèle

(Voir : https://fr.wikipedia.org/wiki/Thermistance)

Relation de Steinhart–Hart (si l'effet Joule est négligeable)

$$\frac{1}{T} = A + B\ln(R) + C\left(\ln(R)\right)^3$$

Avec *A*, *B*, *C* les coefficients de Steinhart–Hart

T en K

R en Ω

Calcul des coefficients de Steinhart–Hart : *A, B, C*

https://fr.wikipedia.org/wiki/Relation_de_Steinhart-Hart

$$\begin{cases} A + (\ln R_1).B + (\ln R_1)^3.C = \dfrac{1}{T_1} \\ A + (\ln R_2).B + (\ln R_2)^3.C = \dfrac{1}{T_2} \\ A + (\ln R_3).B + (\ln R_3)^3.C = \dfrac{1}{T_3} \end{cases}$$

Avec R_1, R_2 et R_3 les valeurs de la résistance aux températures respectives T_1, T_2 et T_3

On pose $Y_1 = \dfrac{1}{T_1}$ $Y_2 = \dfrac{1}{T_2}$ et $Y_3 = \dfrac{1}{T_3}$ ainsi que $L_1 = \ln R_1$ $L_2 = \ln R_2$ et $L_3 = \ln R_3$

On définit :

$$a = \left(\frac{L_2 - L_3}{L_1 - L_2}\right) \times \left(L_2^3 - L_1^3\right) + \left(L_2^3 - L_3^3\right)$$

$$b = Y_2 - Y_3 - \left(\frac{L_2 - L_3}{L_1 - L_2}\right) \times (Y_1 - Y_2)$$

On obtient :

$$C = \frac{b}{a}$$

$$B = \left(\frac{1}{L_1 - L_2}\right) \times \left[Y_1 - Y_2 - \left(L_1^3 - L_2^3\right) \times C\right]$$

$$A = Y_1 - L_1 \cdot B - L_1^3 \cdot C$$

Méthode expérimentale:

- Effectuer 3 mesures de résistances aux températures T1, T2 et T3 encadrant la plage de mesure d'une salle de classe.
  Ex: sur [0°C, 100°C], choisir T2 = 50°C

- Calculer directement les coefficient A, B et C à l'aide des formules précédentes :

$$Y_1 = \frac{1}{T_1} \quad Y_2 = \frac{1}{T_2} \text{ et } Y_3 = \frac{1}{T_3}$$

$$L_1 = \ln R_1 \quad L_2 = \ln R_2 \text{ et } L_3 = \ln R_3$$

$$a = \left(\frac{L_2 - L_3}{L_1 - L_2}\right) \times \left(L_2^3 - L_1^3\right) + \left(L_2^3 - L_3^3\right)$$

$$b = Y_2 - Y_3 - \left(\frac{L_2 - L_3}{L_1 - L_2}\right) \times (Y_1 - Y_2)$$

$$C = \frac{b}{a}$$

$$B = \left(\frac{1}{L_1 - L_2}\right) \times \left[Y_1 - Y_2 - \left(L_1^3 - L_2^3\right) \times C\right]$$

$$A = Y_1 - L_1 \cdot B - L_1^3 \cdot C$$

Modèle simplifié

Sur une plage de températures [T1,T2] limitée

https://fr.wikipedia.org/wiki/Thermistance

$$\frac{R_T}{R_0} = \exp\left(\beta \times \left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$$

Avec sur [T1,T2] :

$$\beta = \frac{T_1 \cdot T_2}{T_2 - T_1} \times \ln\left(\frac{R_1}{R_2}\right)$$

R_0 : Résistance à une température T_0 donnée (souvent 25°C)

On mesure R_T sur la plage de températures [T1,T2] et on en déduit T

$$T = \frac{1}{\frac{1}{\beta}\ln\left(\frac{R_T}{R_0}\right) + \frac{1}{T_0}}$$

Les valeurs de β, R_0 *et* T_0 se trouvent sur les fiches technique des CTN

Fiche technique d'une CTN

Coefficient β pour
T1 = 25°C et T2 = 100°C :

β = 4300K
R0 = 10KΩ
T0 = 25°C = 298K

EPCOS

Temperature Measurement	B57164
Leaded Disks	K 164

R_{25} Ω	No. of R/T characteristic	$B_{25/100}$ K	Ordering code
1 k	1011	3730	B57164K0102+000
1,5 k	1013	3900	B57164K0152+000
2,2 k	1013	3900	B57164K0222+000
3,3 k	4001	3950	B57164K0332+000
4,7 k	4001	3950	B57164K0472+000
6,8 k	2903	4200	B57164K0682+000
10 k	2904	4300	B57164K0103+000
15 k	1014	4250	B57164K0153+000
22 k	1012	4300	B57164K0223+000
33 k	1012	4300	B57164K0333+000
47 k	4003	4450	B57164K0473+000
68 k	2005	4600	B57164K0683+000
100 k	2005	4600	B57164K0104+000
150 k	2005	4600	B57164K0154+000
220 k	2007	4830	B57164K0224+000
330 k	2006	5000	B57164K0334+000
470 k	2006	5000	B57164K0474+000

+: J for $\Delta R_N/R_N = \pm$ 5 %
K for $\Delta R_N/R_N = \pm$ 10 %

Voir https://www.gotronic.fr/pj-691.pdf

Pour la CTN précédente sur la plage de T° [25°C,100°C] la température *T* estimée (en K), ayant mesuré R_T, est donnée par :

$$T = \frac{1}{\frac{1}{4300}\ln\left(\frac{R_T}{10^4}\right) + \frac{1}{298}}$$

Méthode expérimentale:

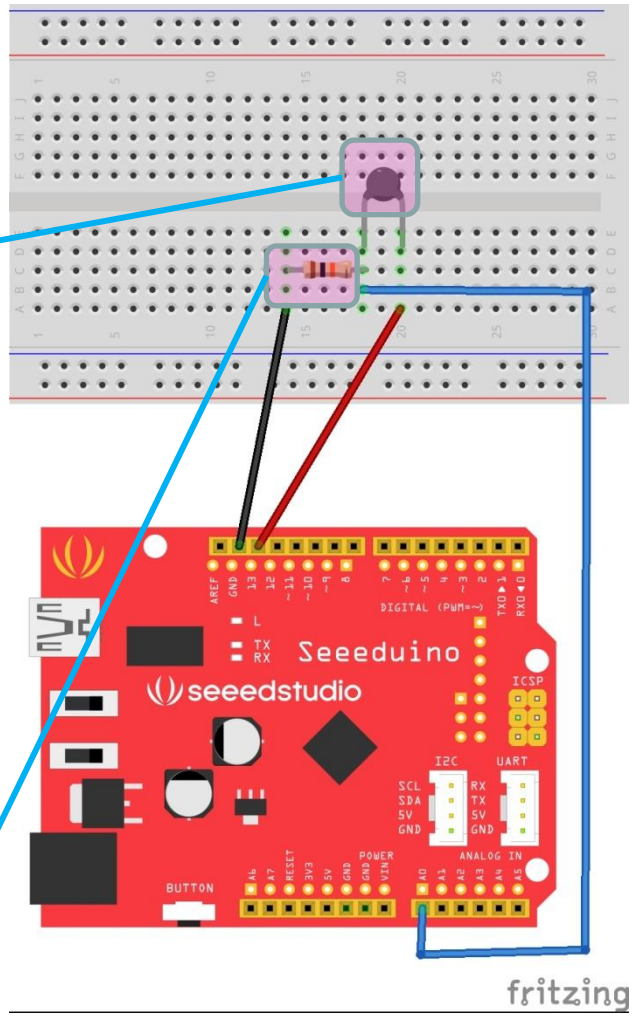
- Effectuer deux mesures de résistances aux températures T1 et T2 encadrant la plage de mesure d'une salle de classe.
  Ex: [0°C, 100°C]

- Calculer directement le coefficient β :

$$\beta = \frac{T_1 \cdot T_2}{T_2 - T_1} \times \ln\left(\frac{R_1}{R_2}\right)$$

Montage : Choix de la résistance fixe

CTN sur $[T_1, T_2]$:
$R1 \in [R_{1min}, R_{1max}]$



$A0 = E \times R0/(R0+R1)$

$\Leftrightarrow$

$R1 = (E/A0-1) \times R0$

Choisir une résistance R' telle que
$R0 = \sqrt{(R_{1min}.R_{1max})}$

Voir : http://www.tonerkebab.fr/wiki/doku.php/wiki:tutoriels:arduino:arduino_resist_2

Algorithme

- **Déclaration des variables globales**
- **Initialisation des variables et des paramètres de contrôles - setup():**
  - **Définir A0 comme une entrée analogique**
- **Boucle de contrôle - loop():**
  - **Alimenter la LED et le montage à CTN**
  - **Lire la valeur de la tension à la pate A0 et mémoriser cette valeur dans la variable « sensorValue ».**
  - **Convertir sensorValue en sa tension correspondante**
  - **Calculer la résistance correspondante**
  - **Calculer la température correspondante (discuter le modèle)**
  - **Cesser d'alimenter le montage**
  - **Envoyer la valeur de la T° à la console**
  - **Patienter quelques secondes (2s)**

Programme

ctn_Modele_Steinhart_Hart.ino (télécharger)

Utiliser la librairie <maths.h> pour calculer les logarithmes

https://www.tutorialspoint.com/c_standard_library/c_function_log.htm

Librairies et variables globales

```
//// Librairies
 #include <math.h> // pour le calcul de la fonction ln(), ici noté log()

/***************************** Paramètre à adapter au besoin du professeur *******/
// Broche de l'entrée analogique utilisée
int capteurPin = A0;
// Broche qui alimente le montage
int alimPin = 2;
// Broche de la led
int ledPin = 13;
// valeur de la résistance du pont en Ohms
// tension d'alimentation en V
int E = 5; //V
/*****************************************************************************/
/***************************** Variables associées aux mesures ******************/
//valeur du capteur de 0 à 1023
  int valeurCapteur;
  // tension calculée en V (de 0 à 5V)
  float tension;
  //résistance calculée en Ohm
  float resistance;
  //température TC calculuée en °C et TK en K
  float TK, TC;
  // Coefficient B de Steinhart–Hart, R0 et T0 : résistance de la CTN à la température T0 (généralement 25°C  =298K)
  float B = 4300; //K
  float R0 = 10000.0; //Ohm
  float T0= 298; //K
```

Initialisation de la carte

```
void setup() {
  // démarrer la communication série à 9600 bauds (valeur par défaut)
  Serial.begin(9600);
  // pin d'alimentations et de led à configurer en sortie
  pinMode (ledPin, OUTPUT);
  pinMode (alimPin, OUTPUT);
}
```

Corps du programme

```
void loop() {
  // allumer la led 13 -> montrer qu'on fait une mesure
  digitalWrite (ledPin, HIGH);
  // alimenter le montage
  digitalWrite  (alimPin, HIGH);
  // attendre un peu (pour stabiliser la mesure)
  delay (100);
  // lecture capteur
  valeurCapteur = analogRead (capteurPin);
  // conversion en tension
  tension = valeurCapteur / 1023.0 * 5.0; //valeur de 0 à 1023, tension de 0 à 5V
  //valeur vers résistance
  resistance = (float)R0 * (E/ tension - 1);

  // résistance vers température
  /************************ Modele Steinhart – Hart ********************************/
  TK =  1 / ((1 / B) * log (résistance / R0) + 1 / T0) ; // température en K
  TC = TK - 273; // température en °C

  /*********************************************************************/
  // arrêter d'alimenter le montage
  digitalWrite (alimPin, LOW);
  // eteindre la led 13 -> mesure finie
  digitalWrite (ledPin, LOW);
  //envoi de la température par lien série avec l'unité (°C)
  Serial.print (TC);
  Serial.println (" °C");
  //attendre 2s avant de refaire une mesure
  delay (2000);
}
```

Programme total (copier/coller)

```
//// Librairies
 #include <math.h> // pour le calcul de la fonction ln(), ici noté log()

/***************************** Paramètre à adapter au besoin du professeur *******/
// Broche de l'entrée analogique utilisée
int capteurPin = A0;
// Broche qui alimente le montage
int alimPin = 2;
// Broche de la led
int ledPin = 13;
// valeur de la résistance du pont en Ohms
// tension d'alimentation en V
int E = 5; //V
/*****************************************************************************/
/***************************** Variables associées aux mesures ******************/
//valeur du capteur de 0 à 1023
  int valeurCapteur;
  // tension calculée en V (de 0 à 5V)
  float tension;
  //résistance calculée en Ohm
  float resistance;
  //température TC calculuée en °C et TK en K
  float TK, TC;
  // Coefficient B de Steinhart–Hart, R0 et T0 : résistance de la CTN à la température T0 (généralement 25°C  =298K)
  float B = 4300; //K
  float R0 = 10000.0; //Ohm
  float T0= 298; //K

void setup() {
  // démarrer la communication série à 9600 bauds (valeur par défaut)
  Serial.begin(9600);
  // pin d'alimentations et de led à configurer en sortie
  pinMode (ledPin, OUTPUT);
  pinMode (alimPin, OUTPUT);
}

void loop() {
  // allumer la led 13 -> montrer qu'on fait une mesure
  digitalWrite (ledPin, HIGH);
  // alimenter le montage
  digitalWrite  (alimPin, HIGH);
  // attendre un peu (pour stabiliser la mesure)
  delay (100);
  // lecture capteur
  valeurCapteur = analogRead (capteurPin);
  // conversion en tension
  tension = valeurCapteur / 1023.0 * 5.0; //valeur de 0 à 1023, tension de 0 à 5V
  //valeur vers résistance
 resistance = (float)R0 * (E/ tension - 1); // si la tension A0 est prise sur la résistance R0

  // résistance vers température
  /************************* Modele Steinhart – Hart ***********************************/
  TK =  1 / ((1 / B) * log (résistance / R0) + 1 / T0) ; // température en K
  TC = TK - 273; // température en °C

  /*************************************************************************/
  // arrêter d'alimenter le montage
  digitalWrite (alimPin, LOW);
  // eteindre la led 13 -> mesure finie
  digitalWrite (ledPin, LOW);
  //envoi de la température par lien série avec l'unité (°C)
  Serial.print (TC);
  Serial.println (" °C");
  //attendre 2s avant de refaire une mesure
  delay (2000);
}
```

6 Afficher l’intensité lumineuse à l’aide d’une photorésistance

Objectif :

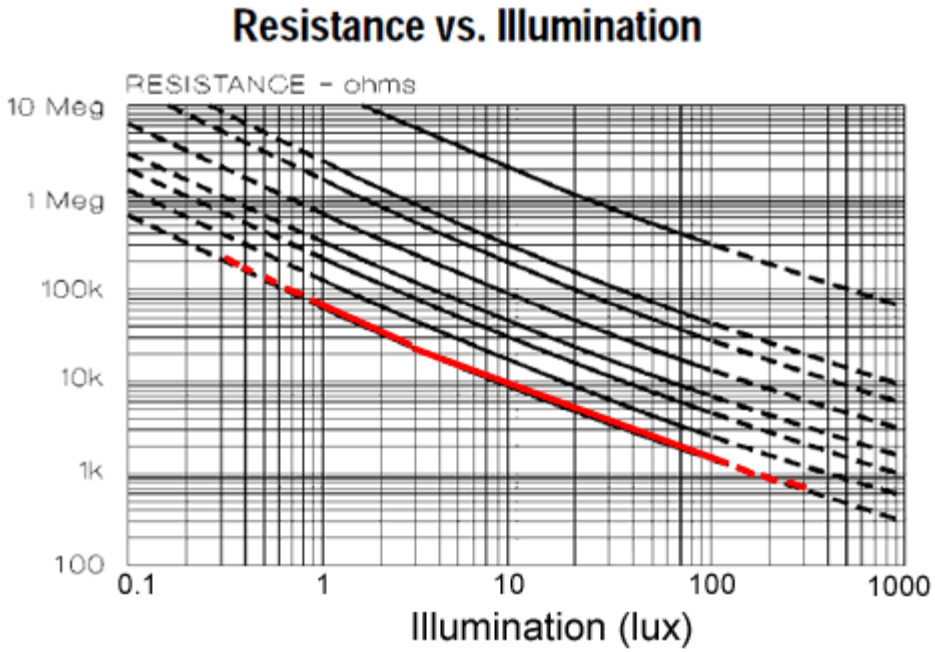
Afficher l’intensité lumineuse à l’aide d’une photorésistance placé dans un pont diviseur de tension

Capteur de lumière: photorésistance

LDR04

http://www.electronic-direct.be/product-details/ldr04/shop.htm?lng=en

Principe de fonctionnement:



La résistance varie ~ entre 100Ω et 2M Ω

Modèles

https://fr.wikipedia.org/wiki/Photor%C3%A9sistance

Lorsque le photon incident est suffisamment énergétique, la production des paires électron-trou est d'autant plus importante que le flux lumineux est intense. La résistance évolue donc comme l'inverse de l'éclairement.

$$R(L) = R_0 L^{-k}$$

Avec *R(L)* la résistance de la photorésistance pour un éclairement *L*.

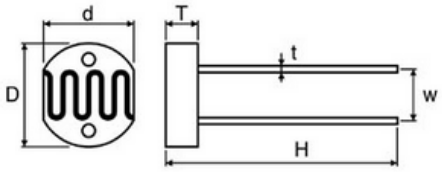
L en Lux

R en Ω

k réel positif

Modèles

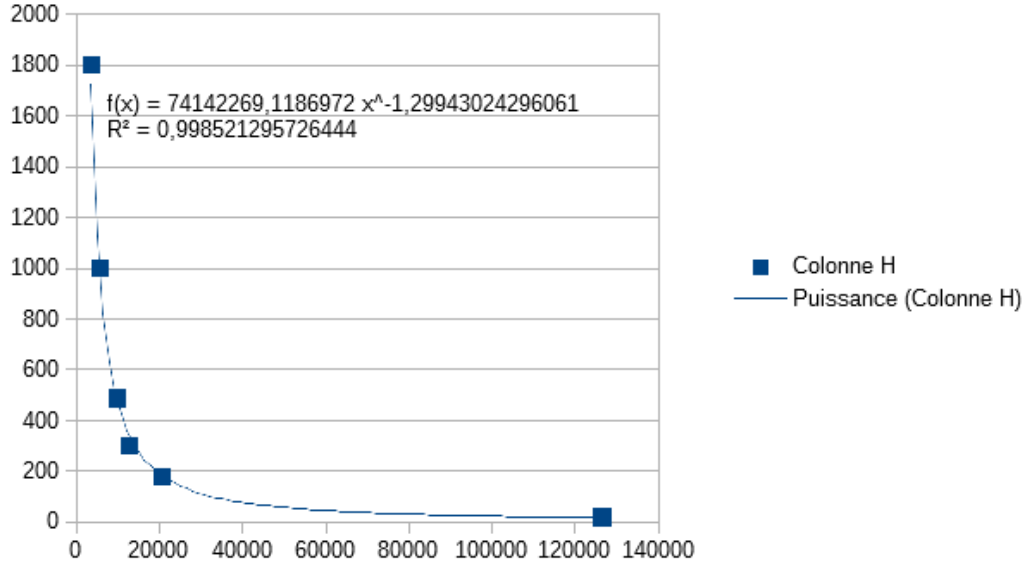
LDR04 - PHOTORESISTOR (LDR) 2-20K

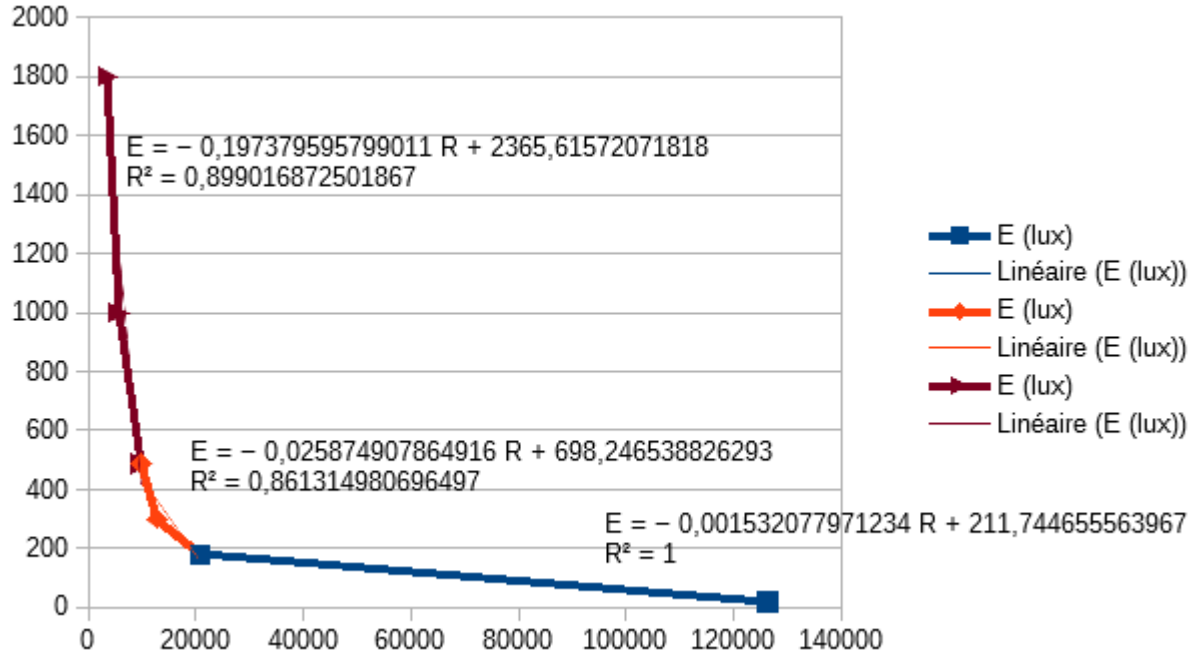


Specifications

- photoresistance (min - max): 2-20 kohm
- dark resistance (after 10 sec.): >2 Mohm
- gamma value at 10-100 Lux: 0.7
- max. power dissipation: 100mW
- max. breakdown voltage: 150Vdc
- peak spectral response: 540nm
- rise response time: 20ms
- fall response time: 30ms
- ambient temperature: -35°C to +70°C
- dimensions:
- D: 4.0 ± 0.2mm
- d: 3.5 ± 0.2mm
- H: 35.5 ± 2mm
- T: 1.5mm
- t: 0.40 ± 0.01mm
- W: 2.5 ± 0.2mm

Modèles


f(x) = 74142269,1186972 x^-1,29943024296061
R² = 0,998521295726444
Colonne H
Puissance (Colonne H)

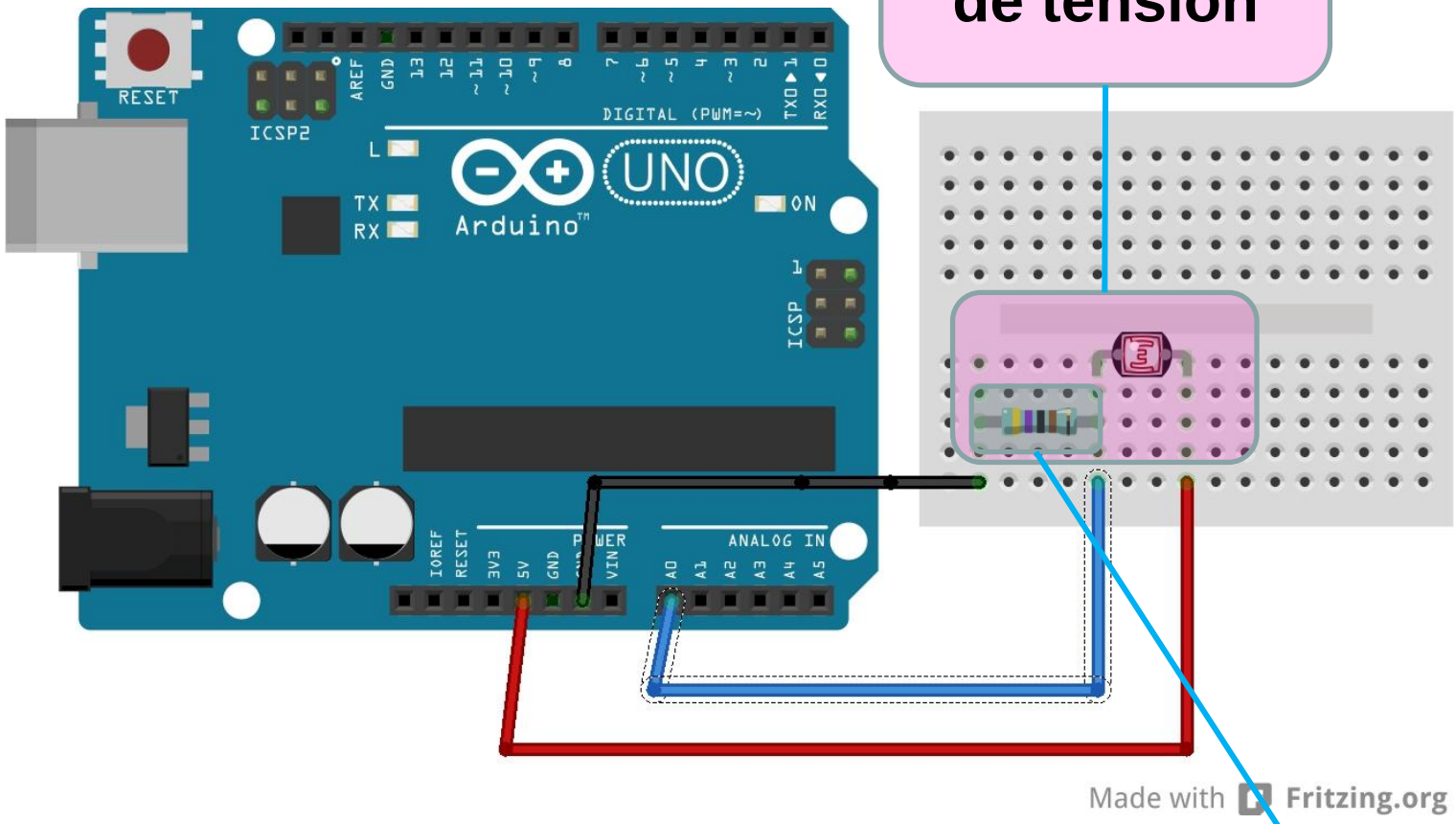

E = − 0,197379595799011 R + 2365,61572071818
R² = 0,899016872501867
E = − 0,025874907864916 R + 698,246538826293
R² = 0,861314980696497
E = − 0,001532077971234 R + 211,744655563967
R² = 1
E (lux)
Linéaire (E (lux))
E (lux)
Linéaire (E (lux))
E (lux)
Linéaire (E (lux))

Eclairement moyen (wikipaedia):

Activité ou lieu concerné	Éclairement moyen
Sensibilité d'une caméra de bas niveau	0,001 lux
Nuit de pleine lune	0,5 lux
Rue de nuit bien éclairée	20 à 70 lux
Local de vie	100 à 200 lux
Appartement bien éclairé	200 à 400 lux
Local de travail	200 à 3 000 lux
Stade de nuit (suivant les différentes catégories (E1,E2,E3,E4,E5))	150 à 1 500 lux
Extérieur par ciel couvert	500 à 25 000 lux
Extérieur en plein soleil	50 000 à 100 000 lux

Montage:

Pont diviseur de tension



Résistance R1 de petite valeur (R1>>R2)

Ici: $U_{A0} = \frac{R1}{(R1+R2)} E \approx \frac{R1}{R2} E$

Algorithme

- **Déclaration des variables globales**
- **Initialisation des variables et des paramètres de contrôles - setup():**
  - **Définir A0 comme une entrée analogique**
- **Boucle de contrôle - loop():**
  - **Lire la valeur de la tension à la pate A0 et mémoriser cette valeur dans la variable « sensorValue ».**
  - **Convertir sensorValue en sa tension correspondante**
  - **Calculer la résistance corresondante**
  - **Calculer la valeur de l'intensité (discuter le modèle)**
  - **Envoyer la valeur de l'angle à la console SSI celui-ci a changé**

Programme

ldr.ino (télécharger)

```
/*****************************************************************
 *  Modèle pour mesure et modélisation de photorésistance
 *    v1 : Olivier Boesch (c) 2018
 */
 #include <math.h>
// Broche de l'entrée analogique utilisée
int capteurPin = A0;
// Broche qui alimente le montage
int alimPin = 2;
// Broche de la led
int ledPin = 13;
// valeur de la résistance du pont en Ohms
float R = 10000.0;

void setup() {
  // démarrer la communication série à 9600 bauds (valeur par défaut)
  Serial.begin(9600);
  // pin d'alimentations et de led à confirgurer en sortie
  pinMode (ledPin, OUTPUT);
  pinMode (alimPin,OUTPUT);
  while(!Serial){
  }
}
```

```
void loop() {
  //valeur du capteur de 0 à 1023
  int valeurCapteur;
  // tension calculée en V (de 0 à 5V)
  float tension;
  //résistance calculée en Ohm
  float resistance;
  //luminosite en lux calculée
  float luminosite;


  // allumer la led 13 -> montrer qu'on fait une mesure
  digitalWrite (ledPin, HIGH);
  // alimenter le montage
  digitalWrite (alimPin, HIGH);
  // attendre un peu (pour stabiliser la mesure)
  delay (100);
  // lecture capteur
  valeurCapteur = analogRead (capteurPin);
  // conversion en tension
  tension = valeurCapteur/1023.0*5.0; //valeur de 0 à 1023, tension de 0 à 5V
  //valeur vers résistance
  resistance = R*(5.0/tension-1.0);
```

```
// résistance vers température
  /************************ Modele ********************************/
  /* 4 segments de droite
     changer les valeurs de resistance et inscrire le modele correspondant */
  if (resistance<9902){
    luminosite = resistance*-0.197379595799011 + 2365.61572071818;
  }
  if ((resistance>=9902) && (resistance<20720)){
    luminosite = resistance*-0.025874907864916+698.246538826293;
  }
  if (resistance>=20720){
    luminosite = resistance*-0.001532077971234 + 211.744655563967;
  }
  /*********************************************************************/
  /******* Modele puissance ************/
  float lum2 = 74142269.1186972*pow(resistance,-1.29943024296061);
  /************************************/
  // arrêter d'alimenter le montage
  digitalWrite (alimPin, LOW);
  // eteindre la led 13 -> mesure finie
  digitalWrite (ledPin, LOW);
  //envoi de la luminosite avec son unité (lux)
  Serial.print (tension);
  Serial.print (" V\t");
  Serial.print (resistance);
  Serial.print (" Ohms\t");
  Serial.print (luminosite);
  Serial.print (" lx\t");
  Serial.print (lum2);
  Serial.println (" lx (p)");
  //attendre 2s avant de refaire une mesure
  delay (2000);
}
```

Annexes

- Sites de références des composants
- Documents de présentation du programme de seconde (M Loos)

Liens de références des composants

- https://www.mouser.co.uk/
- https://www.digikey.fr/
- https://uk.farnell.com/
- https://www.gotronic.fr/

Capteur de température à CTN

https://commons.wikimedia.org/wiki/File:Wireless_th

Les parties du programme traitées :

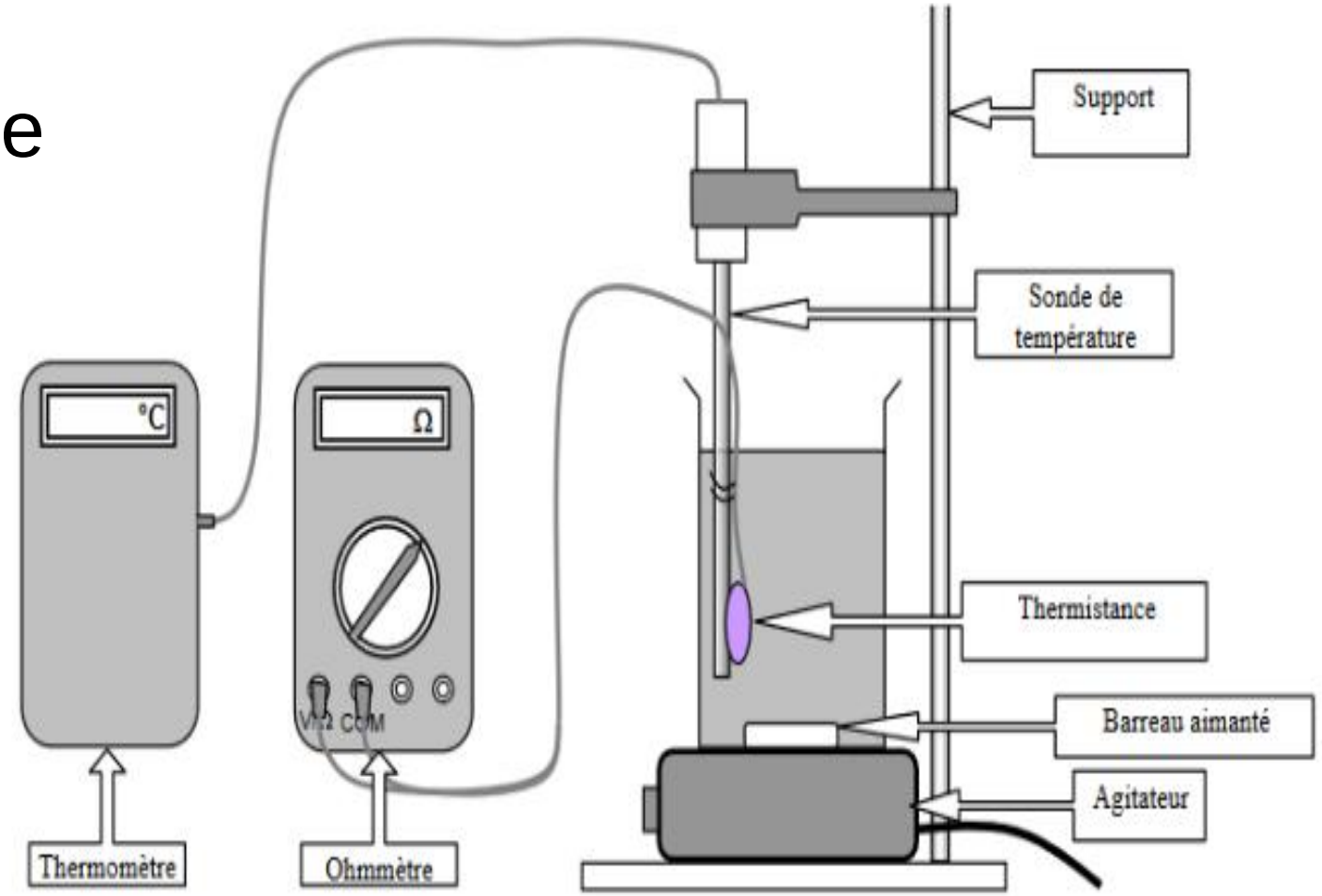
- *Représenter un nuage de points associé à la caractéristique d'un dipôle et modéliser la caractéristique de ce dipôle à l'aide d'un langage de programmation.*
- *Mesurer une grandeur physique à l'aide d'un capteur électrique résistif.*
- *Produire et utiliser une courbe d'étalonnage reliant la résistance d'un système avec une grandeur d'intérêt (température, etc.).*
- *Utiliser un dispositif avec microcontrôleur et capteur.*

Intérêts de cette partie

- Modéliser par une équation le comportement d'un dipôle « non linéaire » : la CTN.
- Utiliser ce modèle mathématique pour remonter à la grandeur à mesurer : la température.
- Applications : thermomètre, thermostat, ...
- Technique transposable à d'autres capteurs.

Caractéristique $R = f(\theta)$ de la CTN

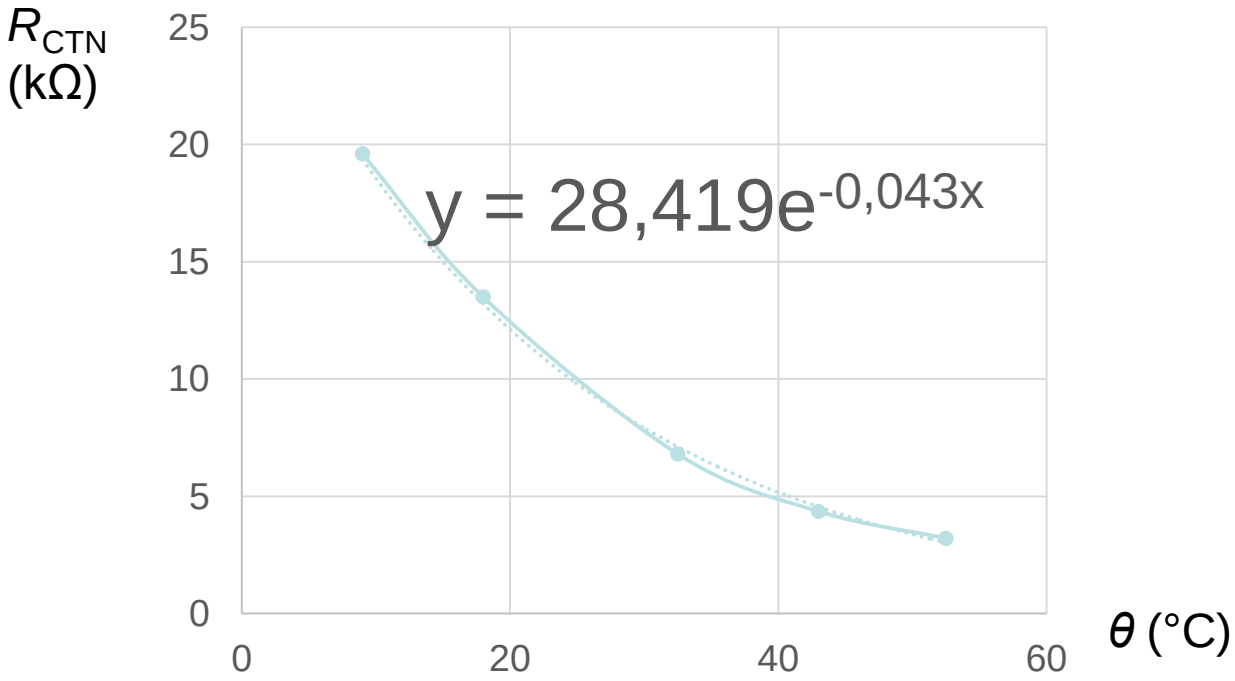
Montage



https://mathsciences.ac-versailles.fr/IMG/pdf/ctn.pdf

Capteur de température à CTN

- *Représenter un nuage de points associé à la caractéristique d'un dipôle et modéliser la caractéristique de ce dipôle à l'aide d'un langage de programmation.*



Modélisation de la caractéristique de la CTN

Le modèle qui permet de relier de manière suffisante la valeur de la résistance d'une CTN à sa température est la suivante :

$$R(T) = R(T_0) \times \exp\left[B\left(\frac{1}{T} - \frac{1}{T_0}\right)\right]$$

avec T_0 la température normalisée 298 K (NF C93-271) et B l'indice de sensibilité thermique caractéristique du matériau qui constitue la CTN.

Modélisation de la caractéristique de la CTN

La relation théorique pour un semi-conducteur est celle de Steinhart-Hart dont tous les éléments sont accessibles sur wikipédia.

$$\frac{1}{T} = C_1 + C_2 \times \ln(R) + C_3 \times [ln(R)]^3$$

L'article donne le code python qui permet de modéliser par cette relation un semi conducteur pour lequel on a fait trois mesures.
https://fr.wikipedia.org/wiki/Relation_de_Steinhart-Hart .

Capteur de température à CTN

- *Mesurer une grandeur physique à l'aide d'un capteur électrique résistif.*
- *Produire et utiliser une courbe d'étalonnage reliant la résistance d'un système avec une grandeur d'intérêt (température …)*
- *Utiliser un dispositif avec microcontrôleur et capteur.*

Programmation du microcontrôleur

Exemple de programme (les coefficients C1 à C3 dépendent de la CTN utilisée et sont à déterminer)

```
int ThermistorPin = 0;
int Vo;
float R1 = 10000;
float logR2, R2, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 =
2.019202697e-07;
```

Programmation du microcontrôleur

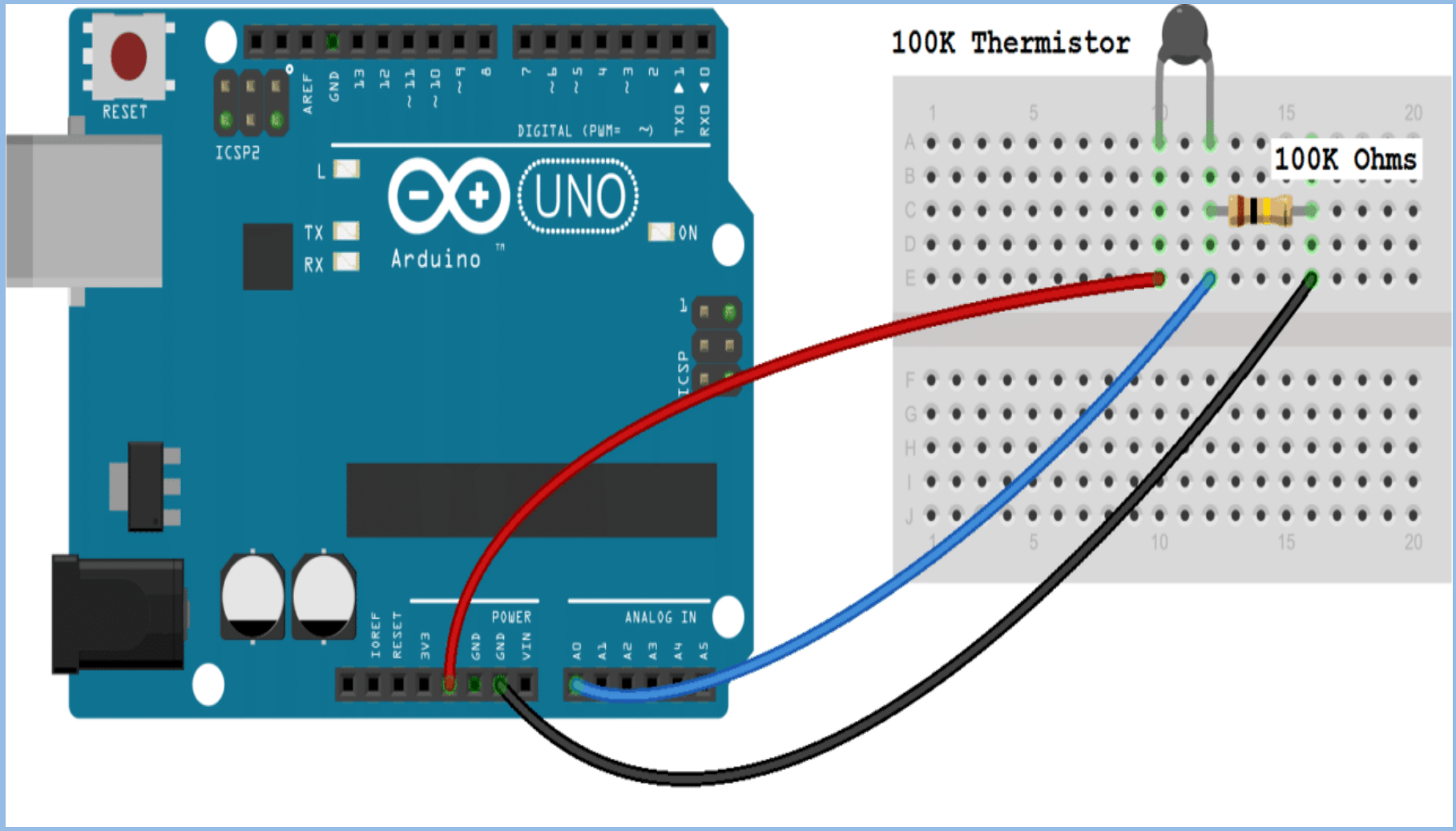
```
void setup()
{
Serial.begin(9600);
}
void loop()
{
  Vo = analogRead(ThermistorPin);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));
  T = T - 273.15;

  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.println(" °C"); // En physique, un résultat sans unité, c’est
zéro !

  delay(2000); // une mesure toutes les deux secondes
}
```

Programmation du microcontrôleur

Nécessite un ordinateur et l'installation du logiciel (libre) qui pilote le microcontrôleur.

Nécessité de programmer ou de trouver un code tout fait sur internet. Celui qui précède est une adaptation de celui qui se trouve sur le site :
http://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/

Câblage du microcontrôleur



http://www.circuitbasics.com/arduino-thermistor-

Câblage du microcontrôleur

Analyse du montage :
La CTN et la résistance sont montées en série avec une source de tension constante de 5 V. Le potentiel entre les deux résistances permet de remonter à la valeur de la résistance de la CTN

$R_{CTN} = Rs\ (U_{RT} / U_{Rs}) = Rs / [(E/U_{RT}) - 1]$

On voit, par la même occasion, que la résistance ne sert pas à limiter l'intensité dans un circuit mais à créer des potentiels déterminés.

Câblage du microcontrôleur

Ce montage peut être réalisé rapidement. Un domino triple et 3 fils sont suffisants.

On peut proposer aux élèves qui maitrisent la syntaxe de refaire le code pour le cas où on inverse la résistance et la CTN (Il faudra leur donner la nouvelle relation entre les grandeurs) ou bien d’afficher le résultat en plusieurs unités (°C, F et K).

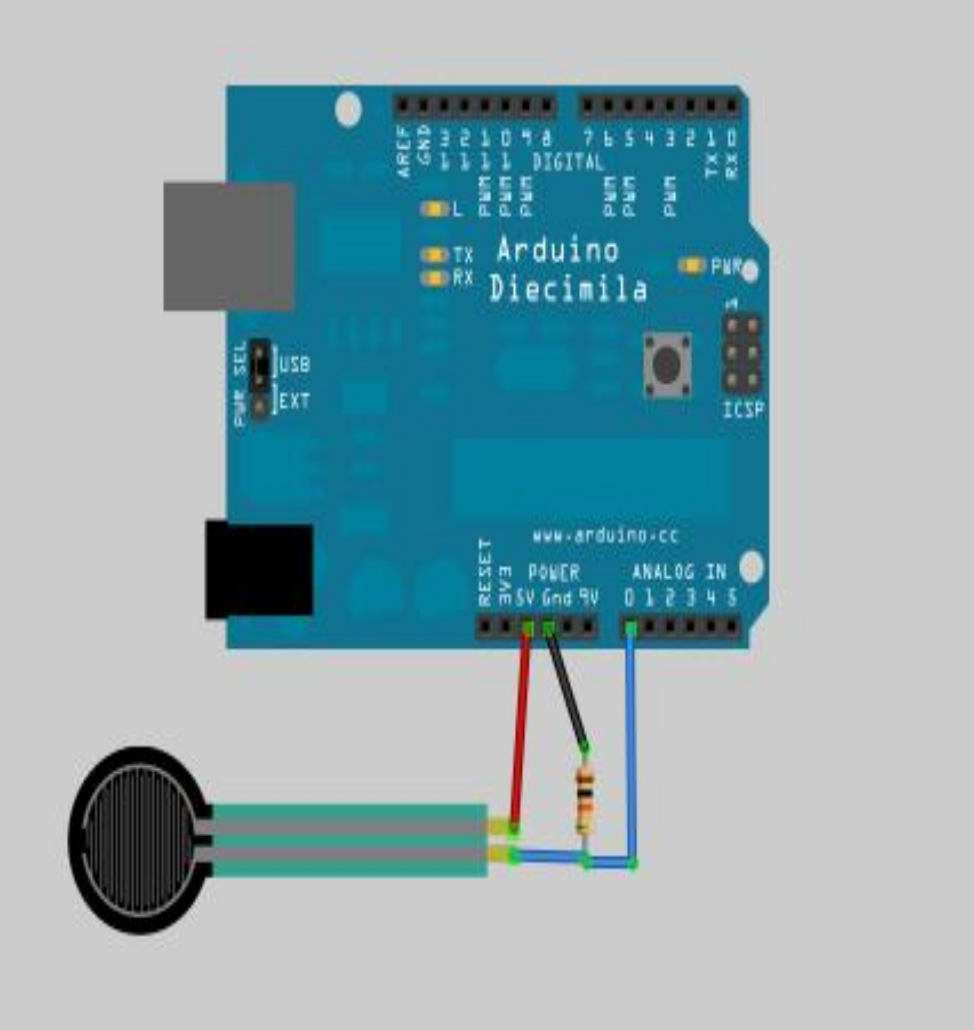
Critique du montage

Remarque : l'essai réalisé avec une CTN et une résistance de 10 KΩ met en évidence que la CTN chauffe (la puissance dissipée par le montage est de l'ordre de 25 mW partagée entre la CTN et la résistance).

On peut améliorer le montage:

- En utilisant une résistance et une CTN de 100 KΩ.
- En l'alimentant seulement le temps de la mesure avec une sortie commandée (la meilleure solution).

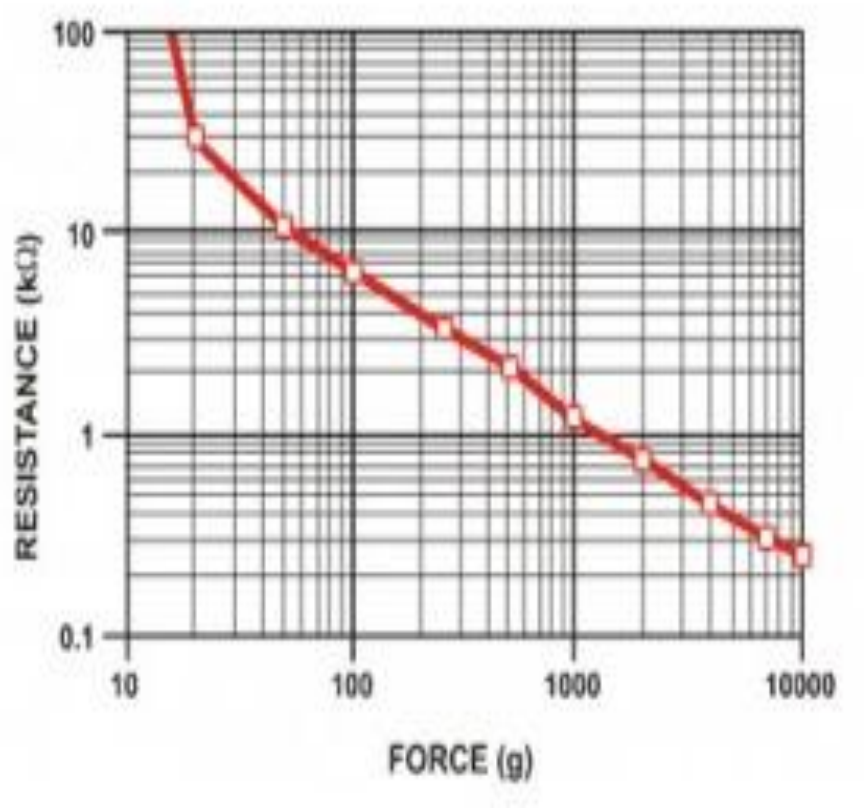
Capteur de force à FSR



http://www.ladyada.net/wiki/tutorials/learn/sensors/fsr.html

Caractéristique de la FSR

Courbe exploitable pour des forces supérieures à 0,5 N



http://www.ladyada.net/wiki/tutorials/learn/sensors/fsr.html

Modélisation de la caractéristique $R = f(F)$

Sur une plage assez importante on peut modéliser la variation de la résistance par une relation de la forme :

$$\log R = A \log F + B$$

D'où la possibilité de déterminer une valeur approximative de la force avec un microcontrôleur.

Modélisation de la caractéristique $R = f(F)$

Remarques :

- C'est simple à mettre en œuvre
- Le capteur est plus coûteux qu'une CTN
- Ce n'est pas très précis.
- Il ne faut pas prendre en compte des mesures réalisées avec une force trop faible.