

# Initiation au hardware et au langage propriétaire Arduino

---

THOMAS BOYER

PROGRAMMEUR BIOINFORMATICIEN

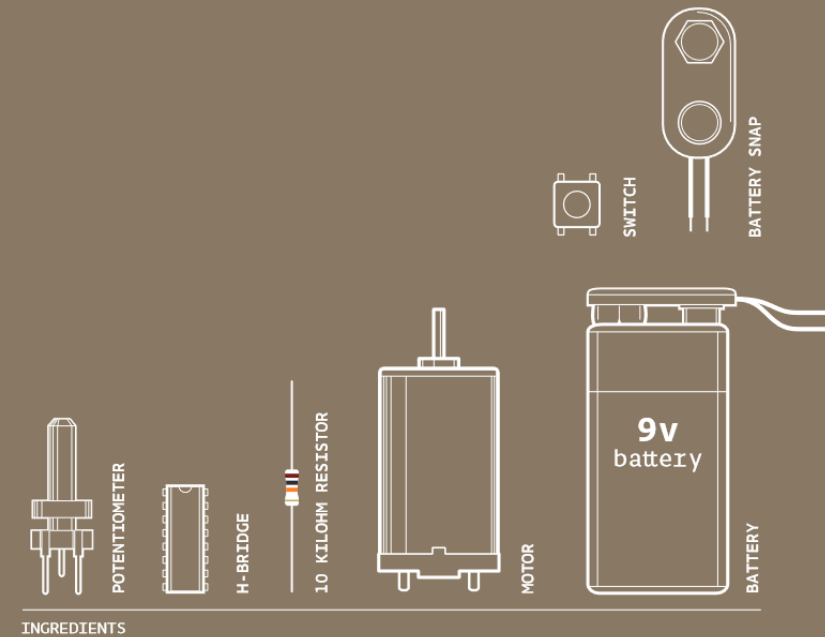
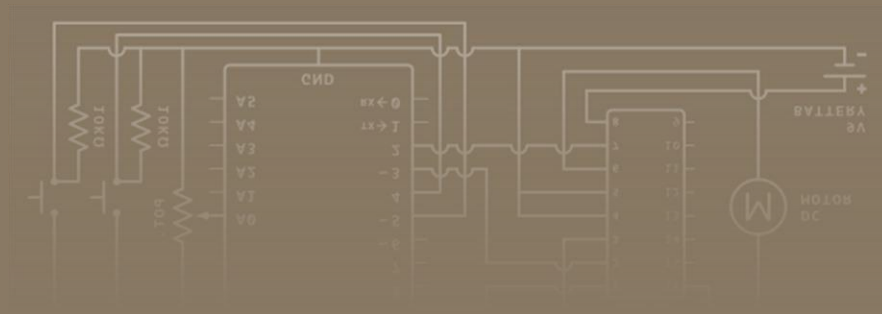
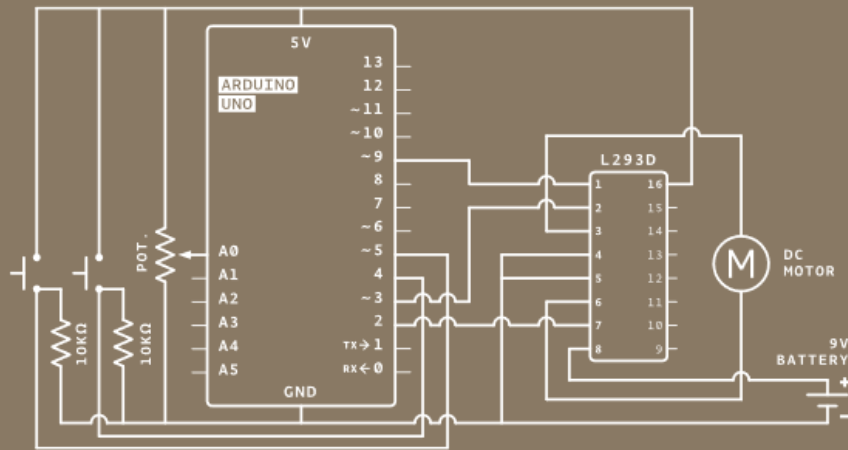


# Plan de séance

---

- Qu'est ce qu'arduino ? Comment ça fonctionne ?
- Installation du logiciel et de l'esp32
- Deux premiers projets
- Présentation des composants
- Développons vos projets

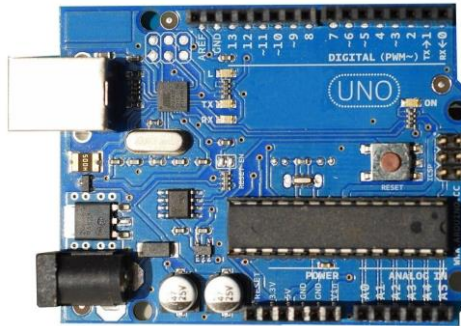
# Arduino : but et principe



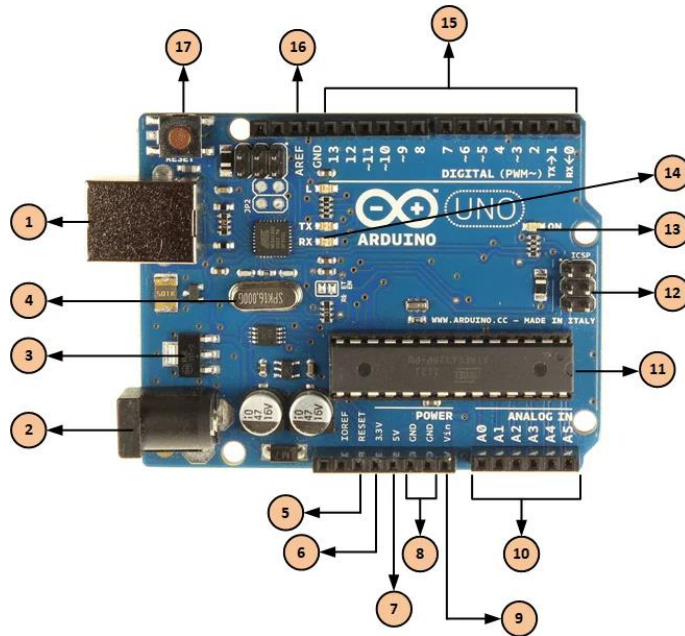
# Arduino

---

- Arduino : prototypage électronique basée sur du matériel et des logiciels flexibles et faciles à utiliser.
  - C'est donc un outil idéal pour créer un projets électroniques sans être un pro
- Le cœur d'un projet arduino, c'est un circuit imprimé qui utilise un microcontrôleur qui peut être programmé pour faire ce que l'on souhaite faire via un langage de programmation Arduino.
  - Pourquoi c'est le cœur ? Parce que ces cartes ont des broches d'entrée/sortie (E/S) qui permettent de connecter différents composants électroniques (capteurs de températures, caméras, etc.)
  - A partir donc d'un simple circuit imprimé, vous pouvez donner vie au projet que vous souhaitez.
- Notre microcontrôleur :
  - Arduino UNO ESP32



# Arduino



1

## Alimentation/Programmation par USB

La carte est alimentée avec un câble USB relié au PC.

2

## Alimentation Jack DC

Autre type d'alimentation. Pas de transfert de programme.

3

## Régulateur de tension

Contrôle la tension d'alimentation de l'Arduino (5V de tension de stabilisation).

4

## Oscillateur à quartz

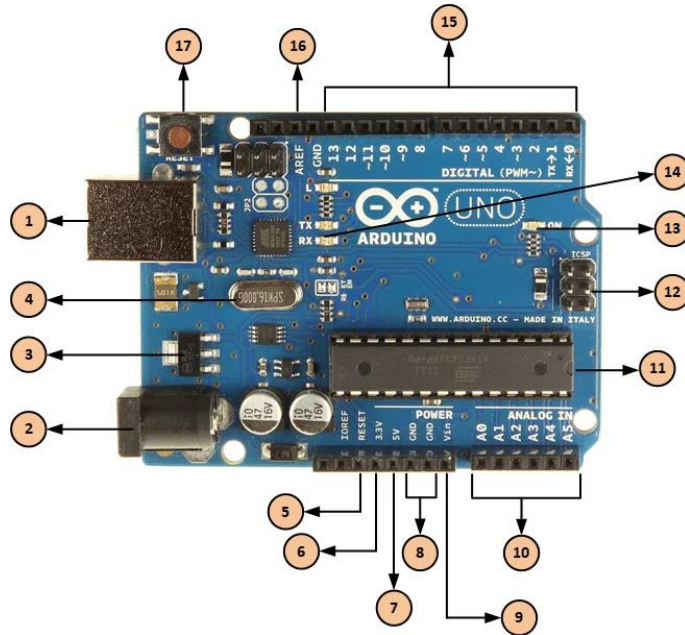
Élément aidant Arduino à calculer les données de temps.

5,17

## Arduino Reset

Bouton de redémarrage du programme : bouton « Reset » (17) ou bouton externe sur la broche de la carte « RESET » (5)

# Arduino



6,7,8,9

## Broches (3.3, 5, GND, Vin)

(6) : Broche d'alimentation 3.3V

(7) : Broche d'alimentation 5V

(8) : Terre/Masse

(9) : Vin (source de tension extérieure)

10

## Broches analogiques

Broches permettant de lire un signal analogique d'un capteur (capteur d'humidité, de température, etc.)

11

## Microcontrôleur principal

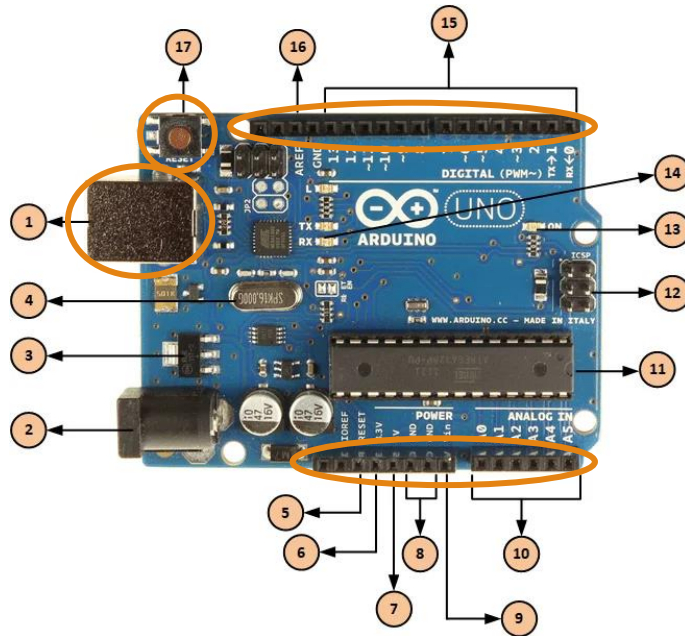
« Cerveau » de la puce

12

## Connecteur ICSP

Connecteurs de programmation

# Arduino



13

## Indicateur LED d'alimentation

Voyant indiquant si l'Arduino est alimentée.

14

## LEDs TX et RX

LEDs émission (TX) et réception (RX) de signal.

15

## Entrées/Sorties numériques

16

## Broche AREF

Broche utilisée pour définir une tension de référence externe

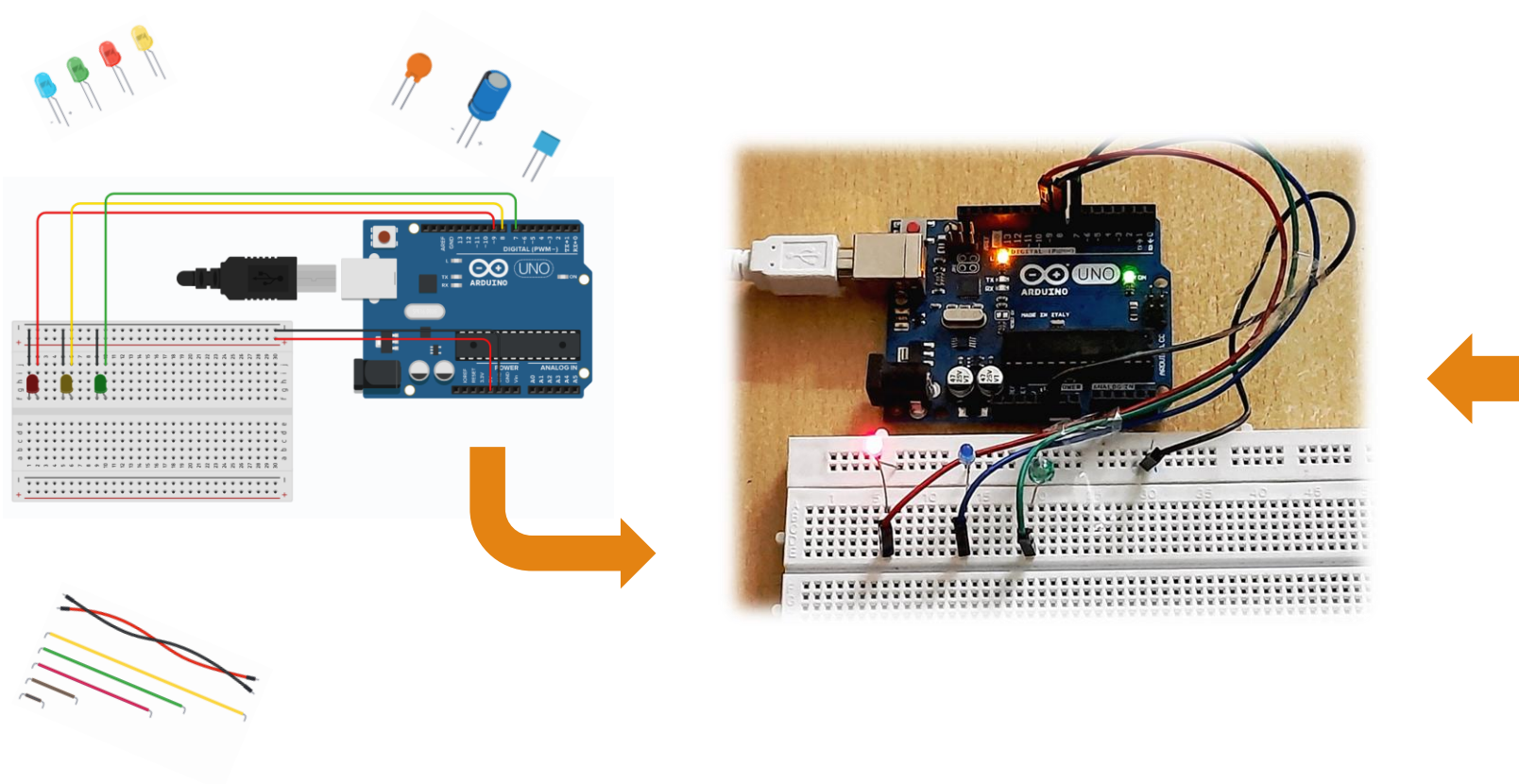
## L'important pour nous :

- L'alimentation
- Arduino Reset
- Broches et entrées/sorties numériques

OK, et maintenant ?

# Arduino

- Ajout des composants nécessaires au projet et programmation



```
sketch_nov26a | Arduino IDE 2.0.1
File Edit Sketch Tools Help
Select Board
sketch_nov26a.ino
1 int red = 9;
2 int yellow = 8;
3 int green = 7;
4
5 void setup(){
6
7   pinMode(red, OUTPUT);
8   pinMode(yellow, OUTPUT);
9   pinMode(green, OUTPUT);
10
11 }
12 void loop(){
13   digitalWrite(red, HIGH);
14   delay(15000);
15   digitalWrite(red, LOW);
16
17   digitalWrite(yellow, HIGH);
18   delay(1000);
19   digitalWrite(yellow, LOW);
20   delay(500);
21
22   digitalWrite(yellow, HIGH);
23   delay(1000);
24   digitalWrite(yellow, LOW);
25   delay(500);
26
27 }
Output
Installing Firmata@2.5.9
Installed Firmata@2.5.9
Téléchargement LiquidCrystal@1.0.7
LiquidCrystal@1.0.7
Installing LiquidCrystal@1.0.7
Installed LiquidCrystal@1.0.7
Téléchargement SD@1.2.4
SD@1.2.4
```



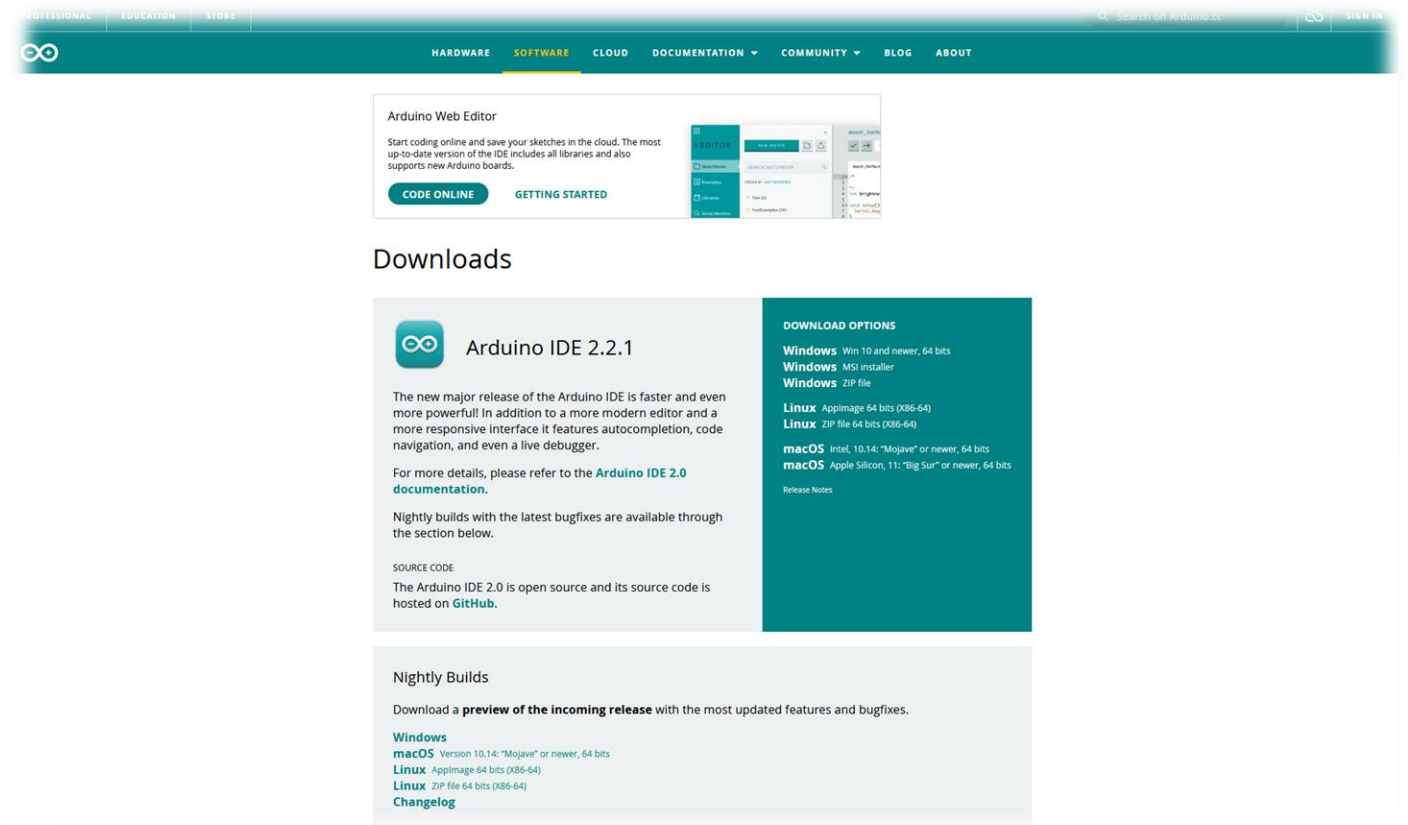
# Installation de l'IDE et du microcontrôleur

---

# Installation de l'IDE

- Avant de bidouiller, il faut installer l'environnement de développement intégré (IDE) pour interagir avec le microcontrôleur :

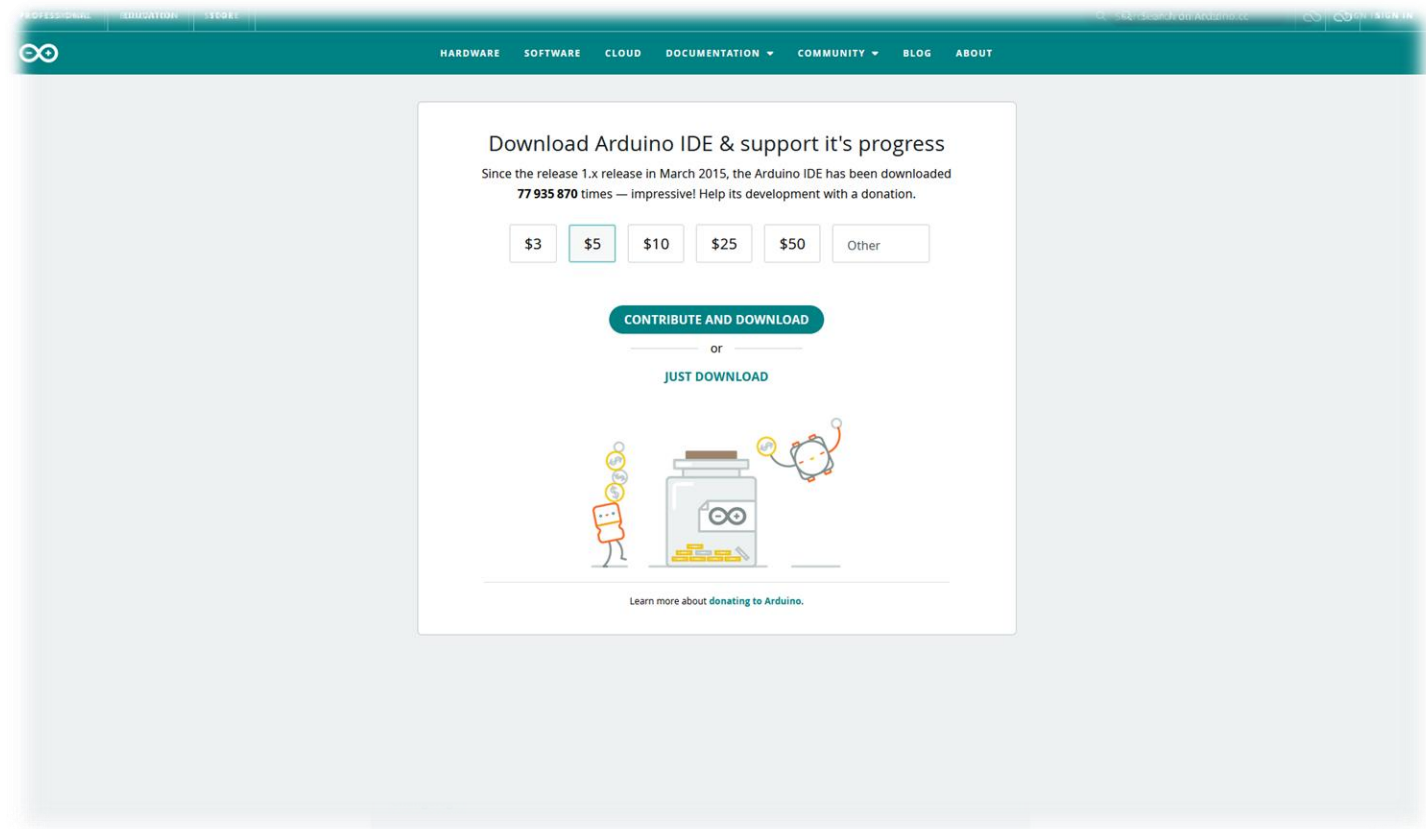
- <https://www.arduino.cc/download>



# Installation de l'IDE

- Avant de bidouiller, il faut installer l'environnement de développement intégré (IDE) pour interagir avec le microcontrôleur :

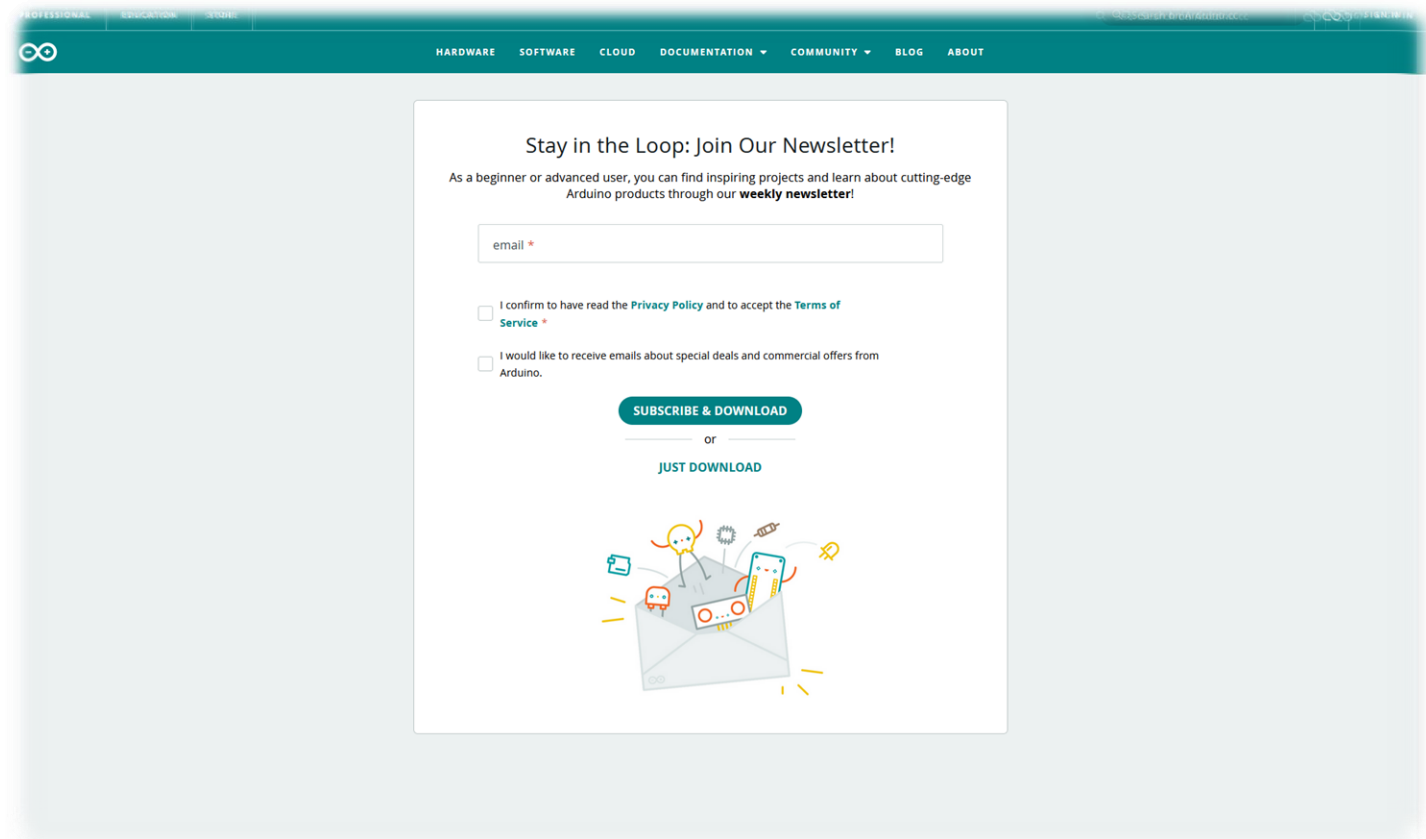
- <https://www.arduino.cc/download>



# Installation de l'IDE

- Avant de bidouiller, il faut installer l'environnement de développement intégré (IDE) pour interagir avec le microcontrôleur :

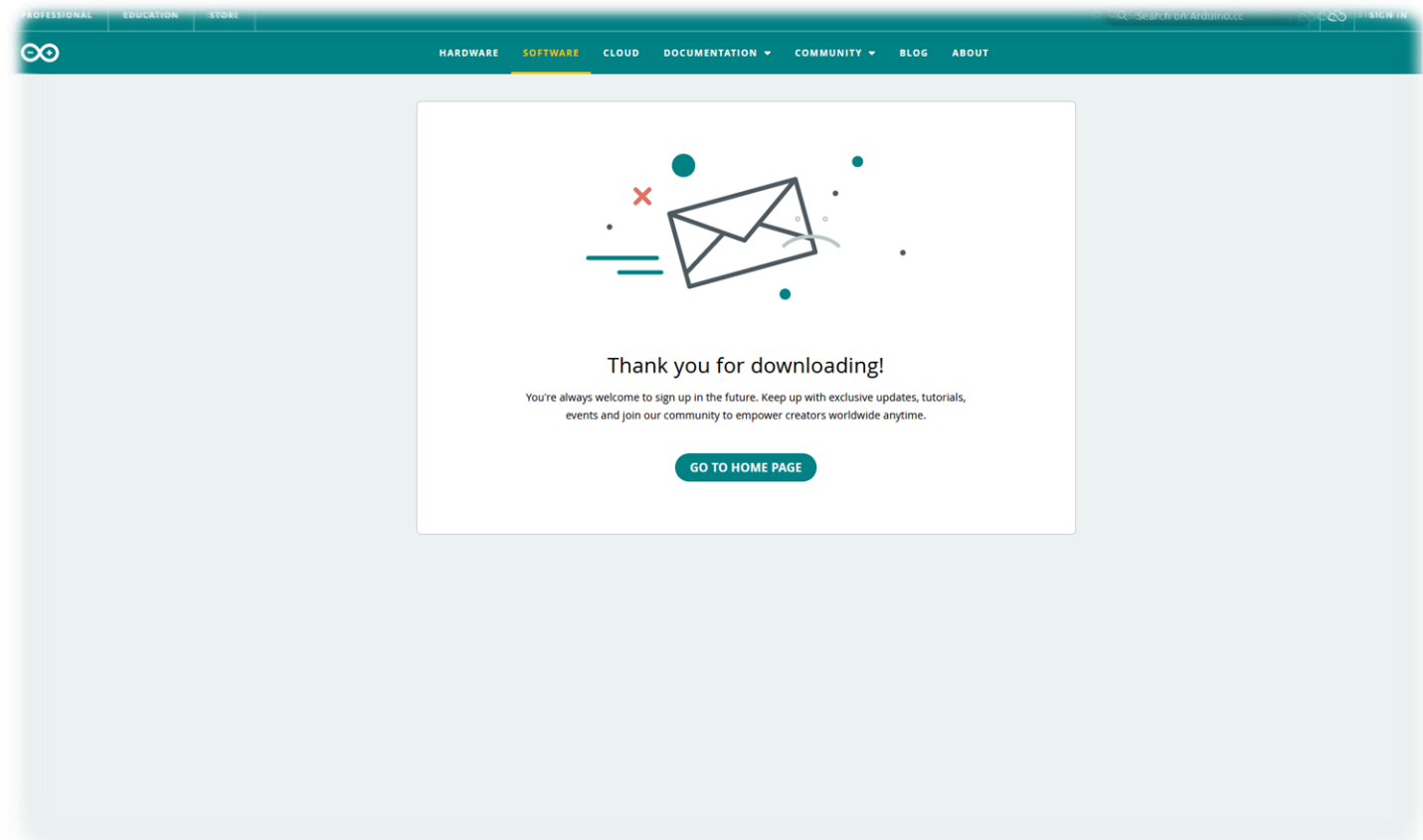
- <https://www.arduino.cc/download>



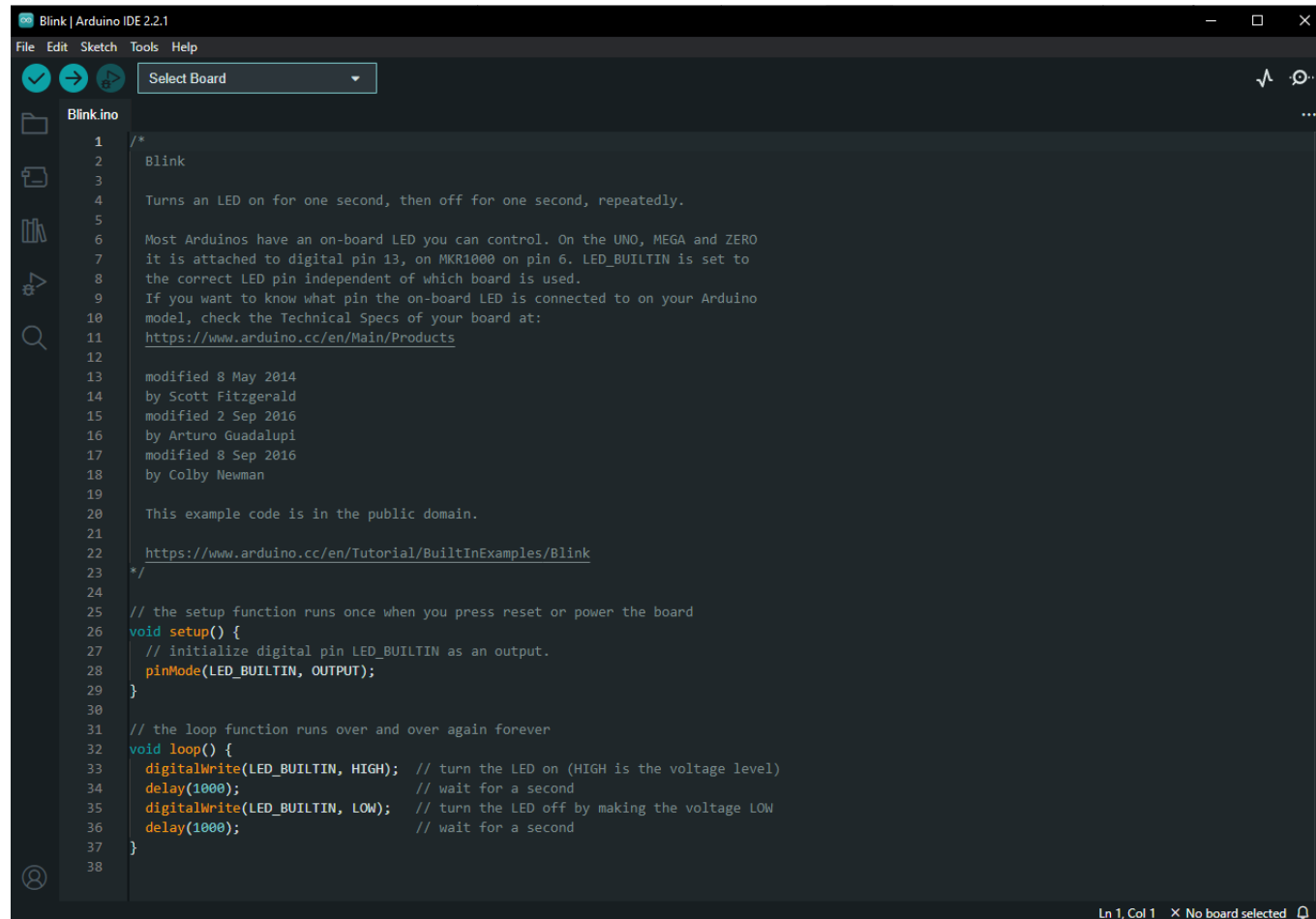
# Installation de l'IDE

- Avant de bidouiller, il faut installer l'environnement de développement intégré (IDE) pour interagir avec le microcontrôleur :

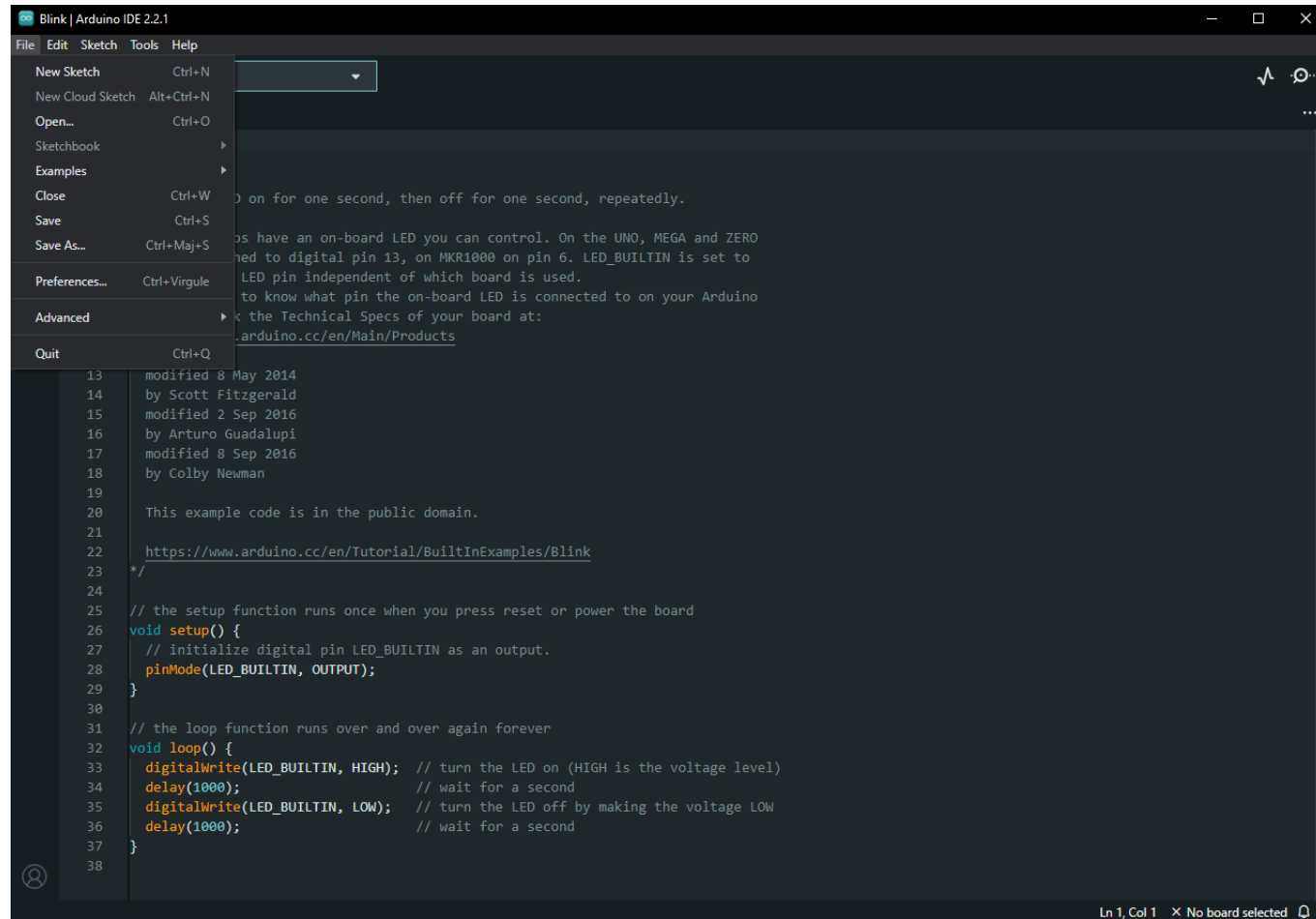
- <https://www.arduino.cc/download>



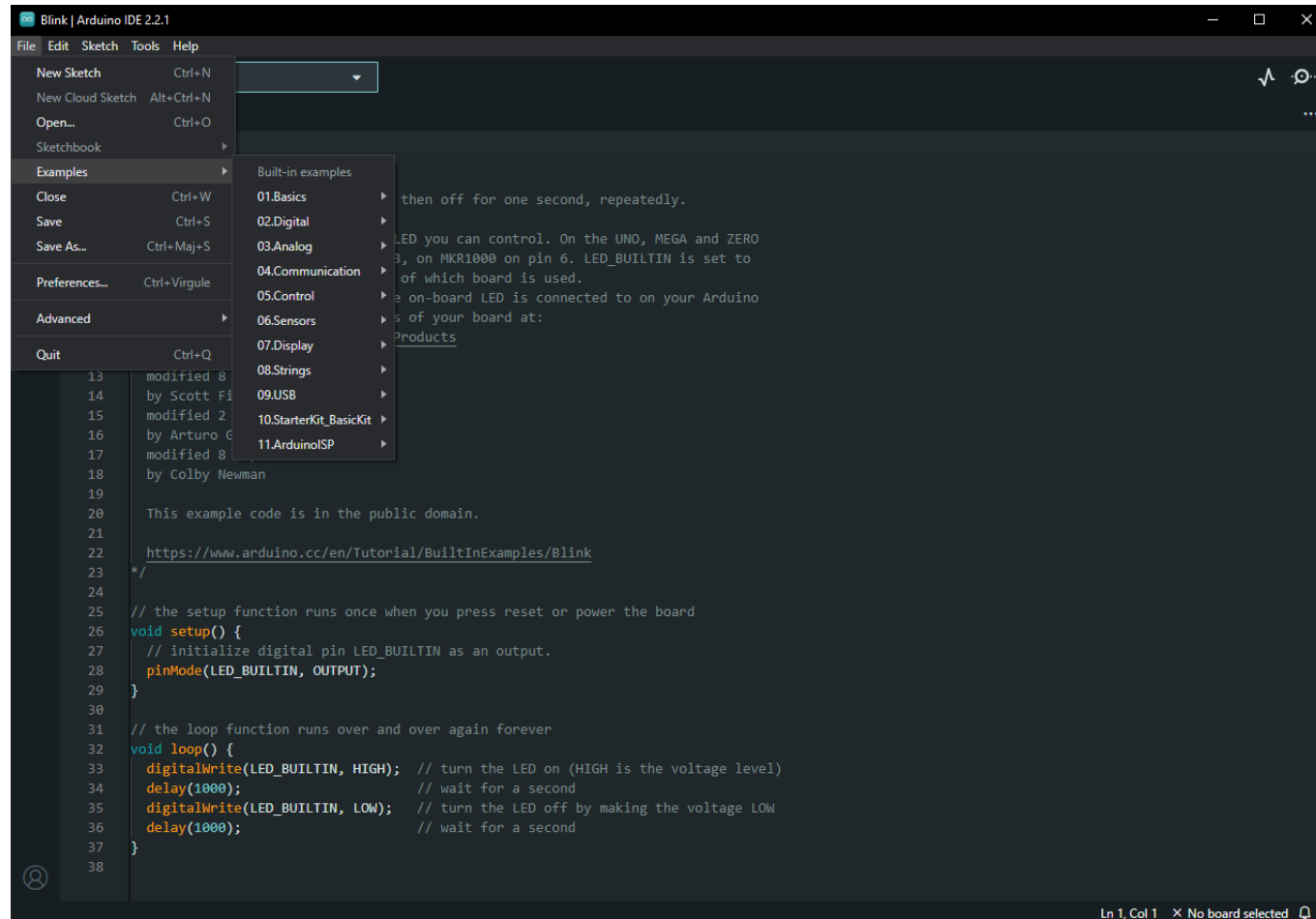
# Installation de l'IDE



# Installation de l'IDE

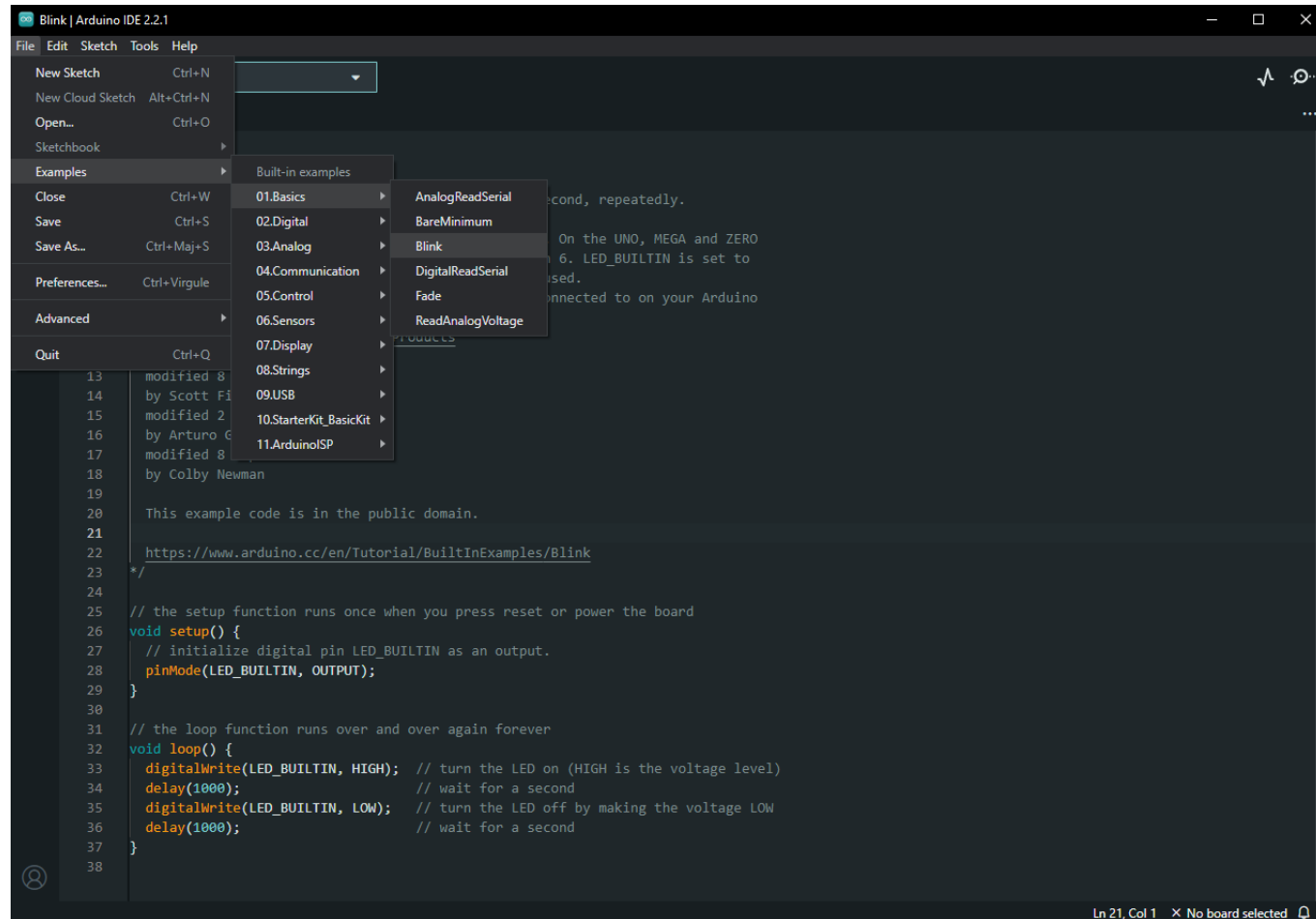


# Installation de l'IDE

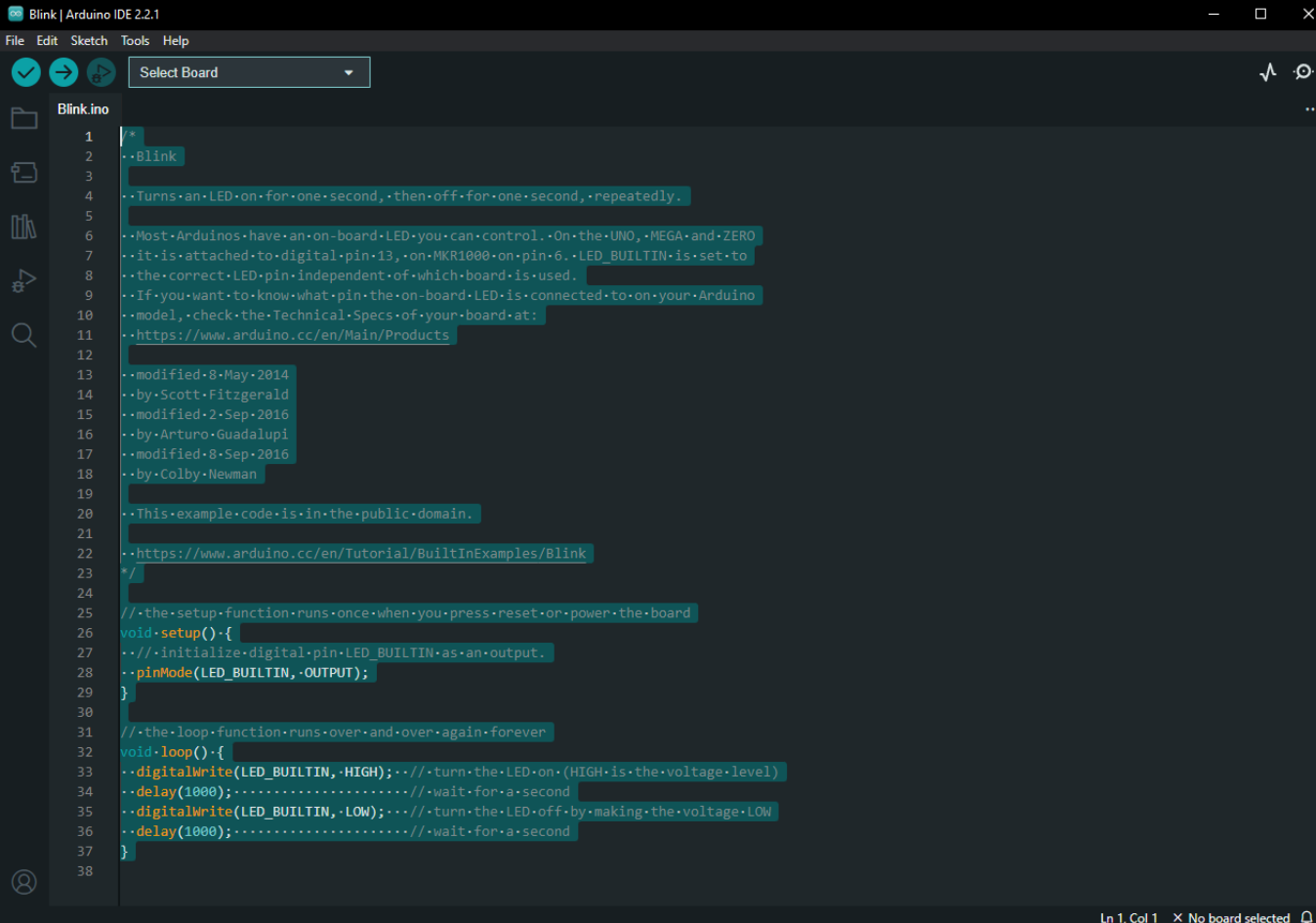




# Installation de l'IDE



# Installation de l'IDE

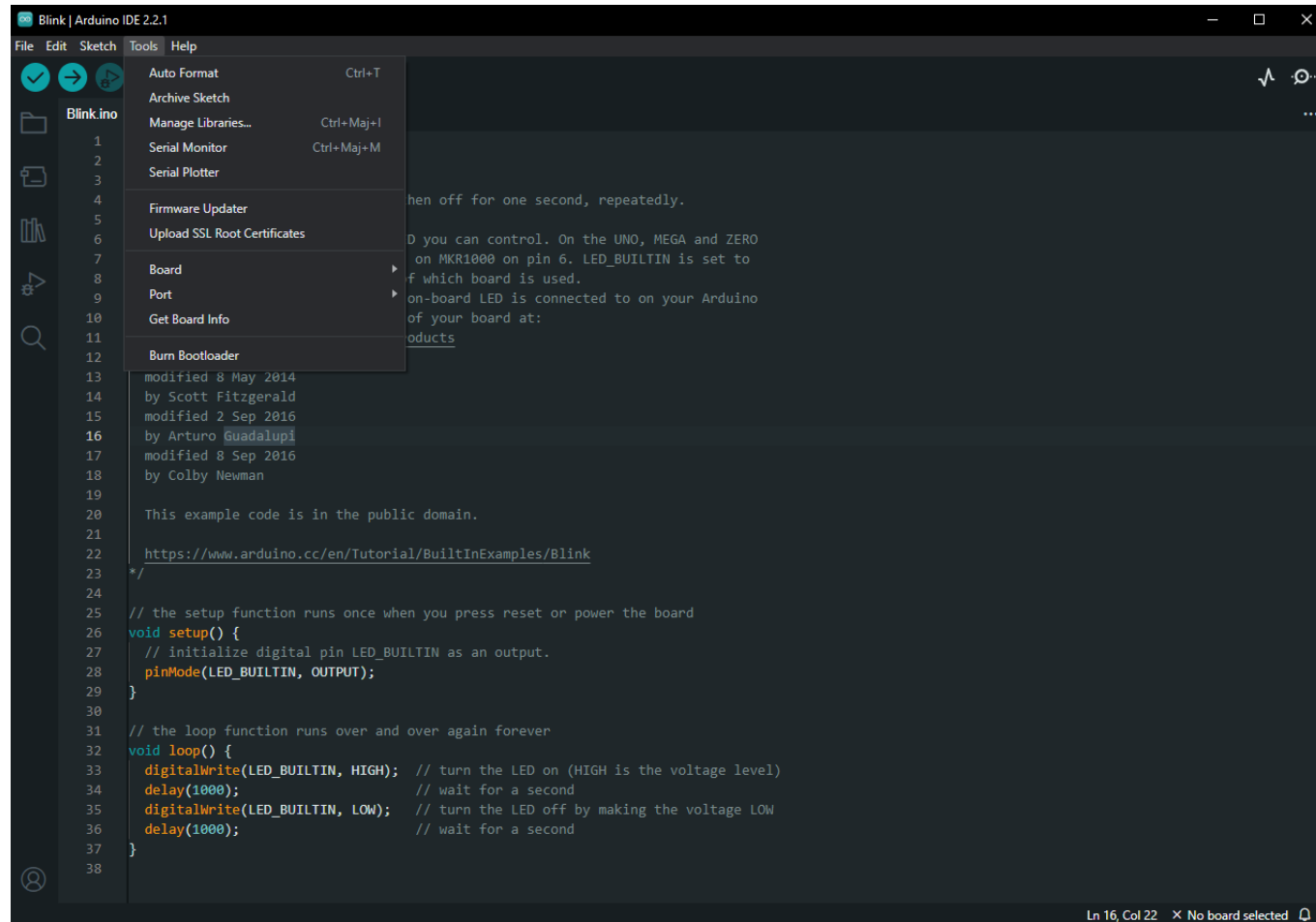


The screenshot displays the Arduino IDE 2.2.1 interface. The title bar reads "Blink | Arduino IDE 2.2.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, and uploading, along with a "Select Board" dropdown menu. The left sidebar contains icons for file explorer, search, and a user profile. The main editor area shows the "Blink.ino" file with the following code:

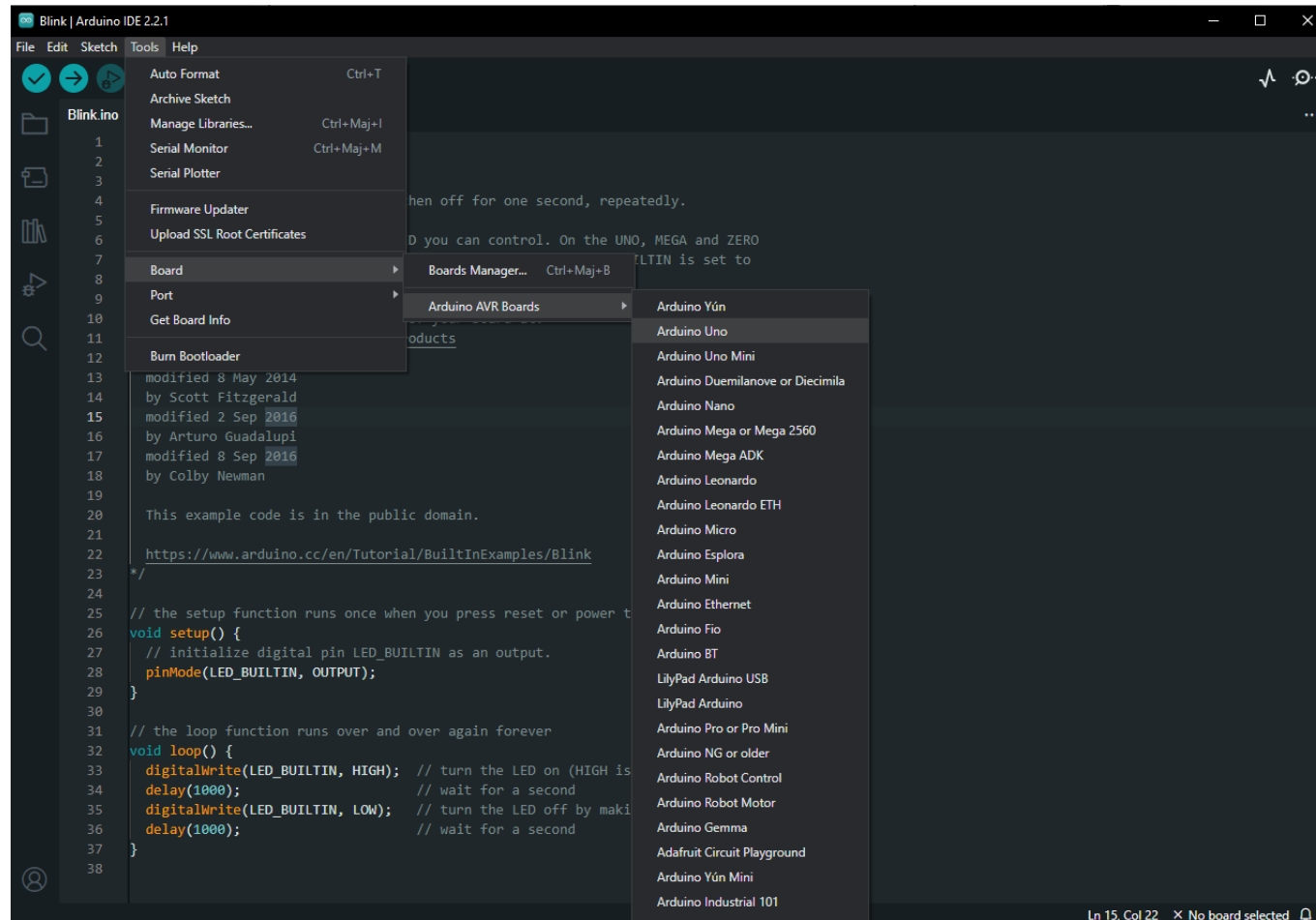
```
1  *
2  ..Blink
3
4  ..Turns an LED on for one second, then off for one second, repeatedly.
5
6  ..Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  ..it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  ..the correct LED pin independent of which board is used.
9  ..If you want to know what pin the on-board LED is connected to on your Arduino
10 ..model, check the Technical Specs of your board at:
11 ..https://www.arduino.cc/en/Main/Products
12
13 ..modified 8-May-2014
14 ..by Scott Fitzgerald
15 ..modified 2-Sep-2016
16 ..by Arturo Guadalupi
17 ..modified 8-Sep-2016
18 ..by Colby Newman
19
20 ..This example code is in the public domain.
21
22 ..https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38
```

The status bar at the bottom right indicates "Ln 1, Col 1" and "No board selected".

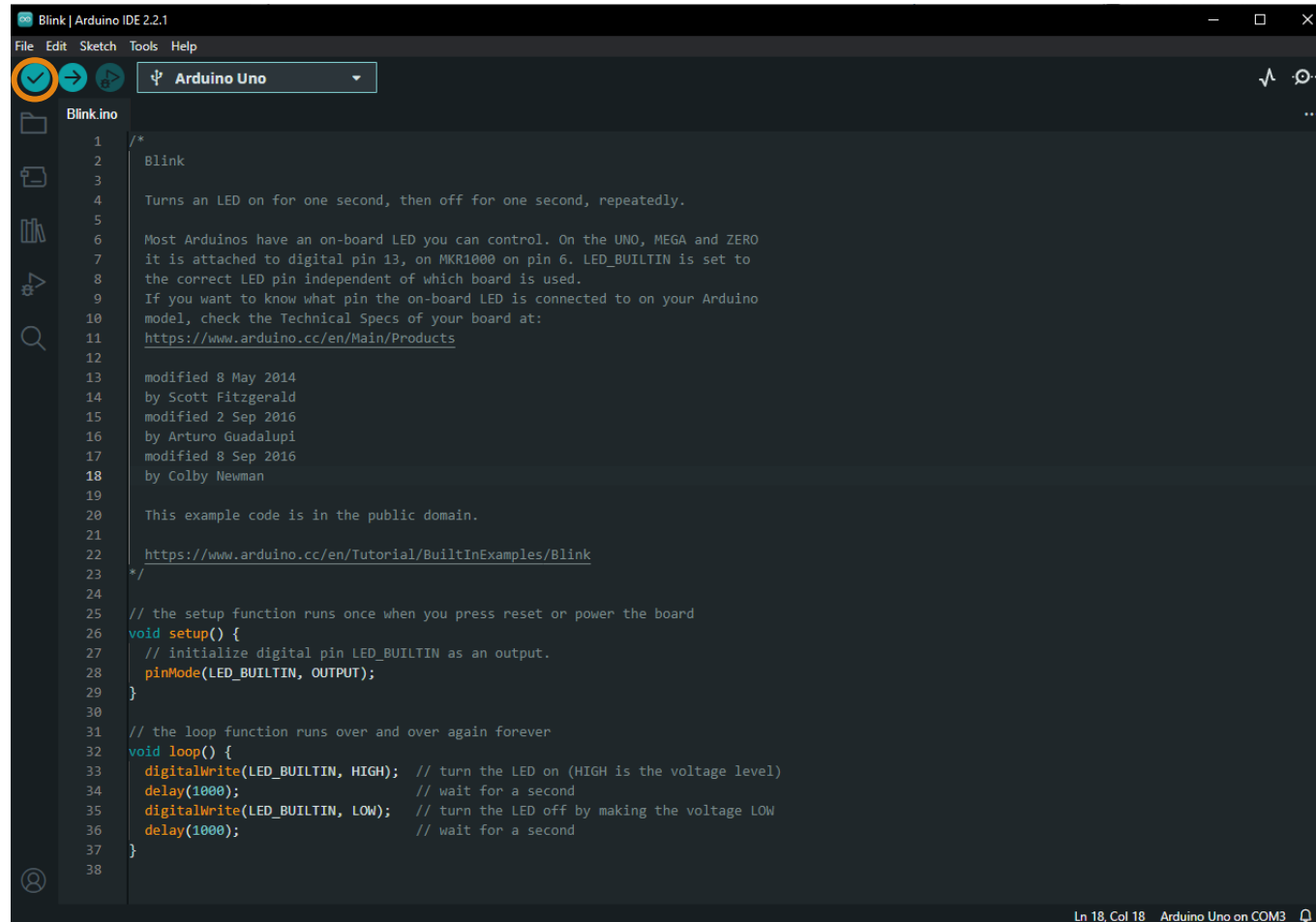
# Connexion au microcontrôleur



# Connexion au microcontrôleur

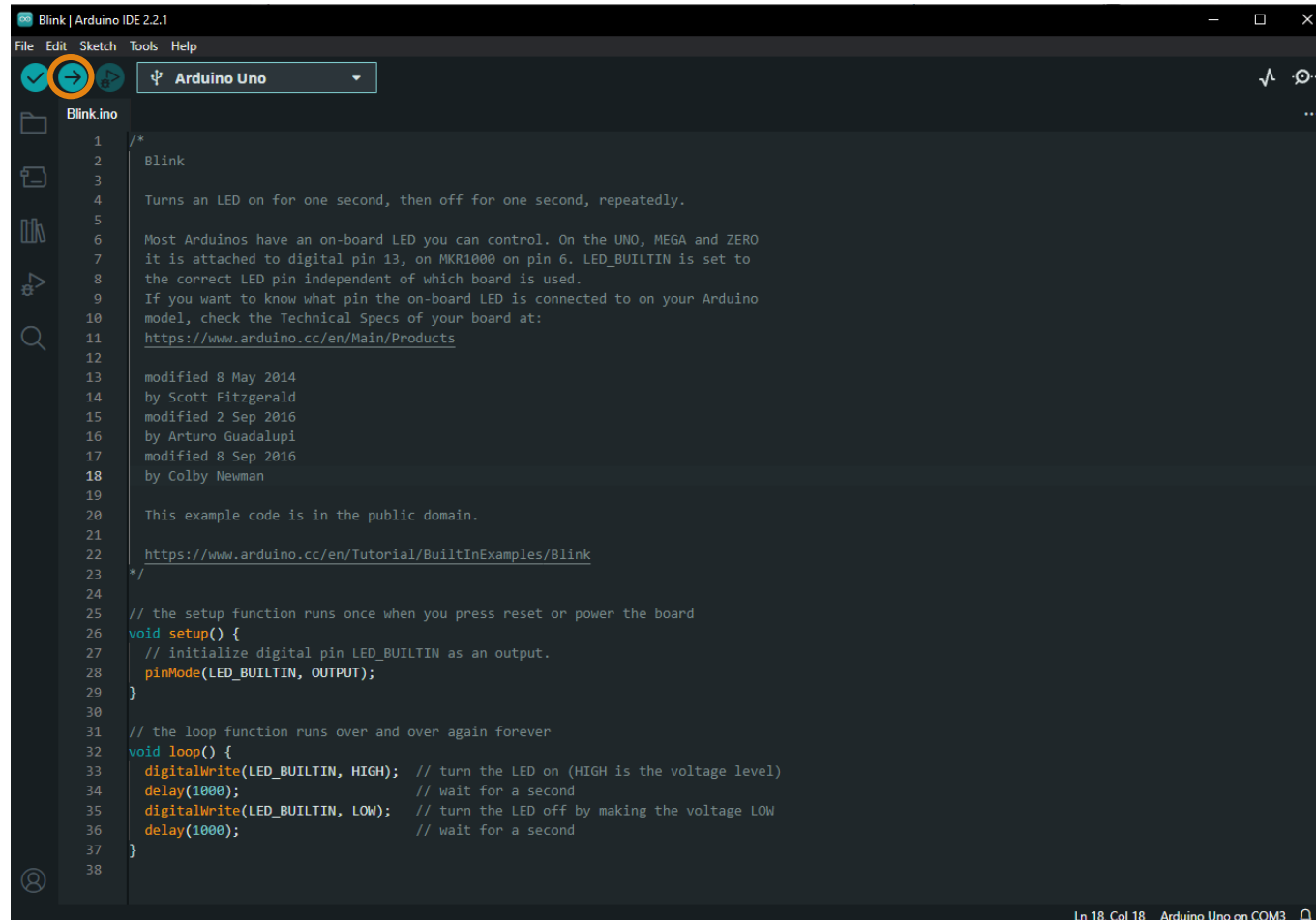


# Connexion au microcontrôleur



```
Blink | Arduino IDE 2.2.1
File Edit Sketch Tools Help
[Verify] [Run] [Serial] Arduino Uno
Blink.ino
1  /*
2  Blink
3
4  Turns an LED on for one second, then off for one second, repeatedly.
5
6  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  the correct LED pin independent of which board is used.
9  If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38
Ln 18, Col 18  Arduino Uno on COM3
```

# Connexion au microcontrôleur



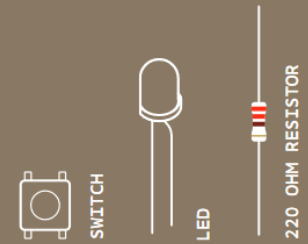
```
1  /*
2  Blink
3
4  Turns an LED on for one second, then off for one second, repeatedly.
5
6  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  the correct LED pin independent of which board is used.
9  If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38
```



Ca clignote ?

# Premier projet

---



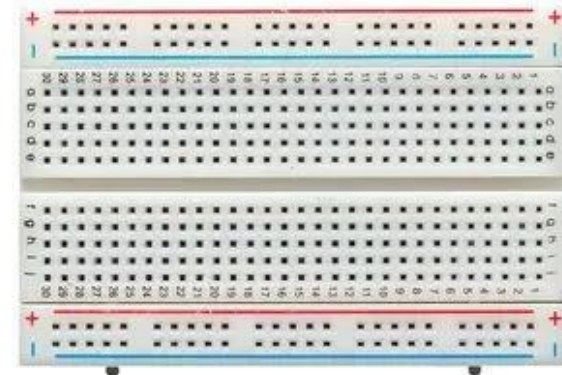
---

INGREDIENTS

# Premier projet

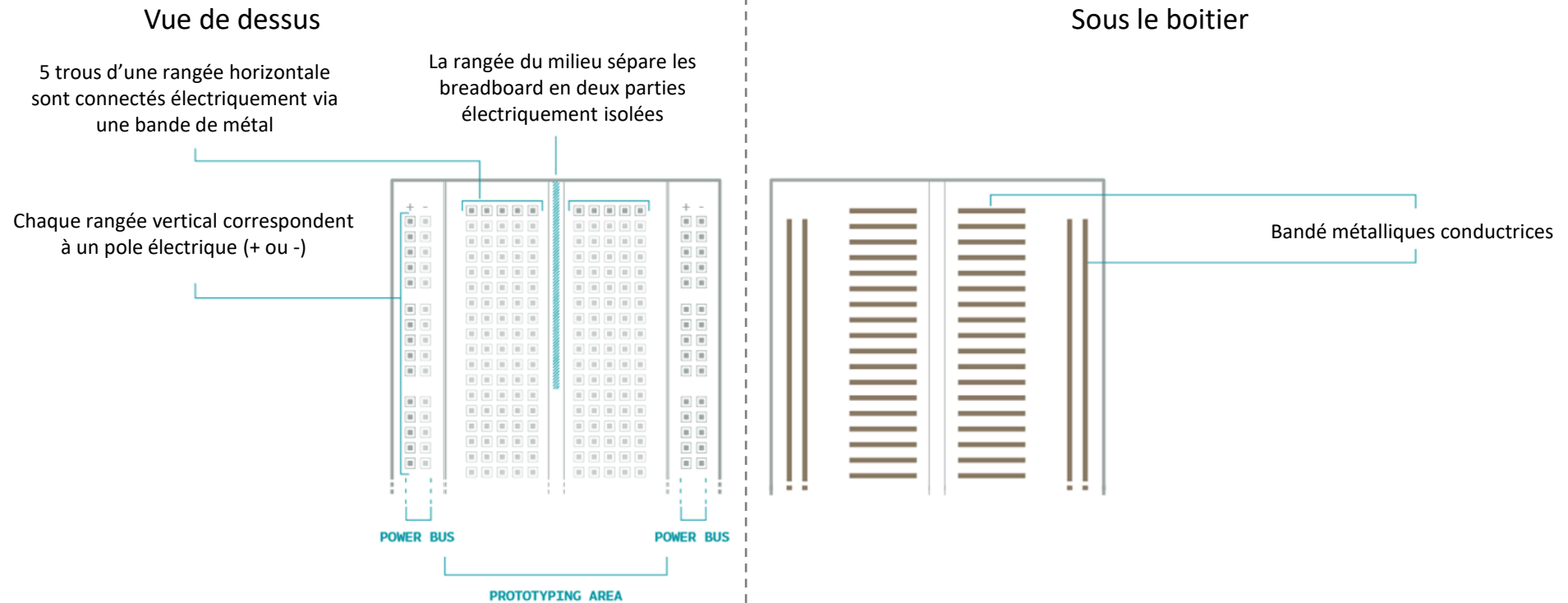
---

- L'électricité est une forme d'énergie, comme l'est la chaleur, la lumière ou la gravité. On peut transformer cette énergie en une autre forme d'énergie (lumière d'une ampoule, son dans une enceinte, etc.).
  - Pour faire ça, on utilise ce qu'on appelle des transducteurs : un élément qui transforme un signal physique en un autre (électricité en chaleur, électricité en lumière, etc.).
  - Pour faire marcher tout ces éléments, on doit les relier à un générateur d'électricité : on crée alors ce qu'on appelle un circuit, normalement via des fil électriques et des soudures.
    - On va s'affranchir des soudures avec une breadboard



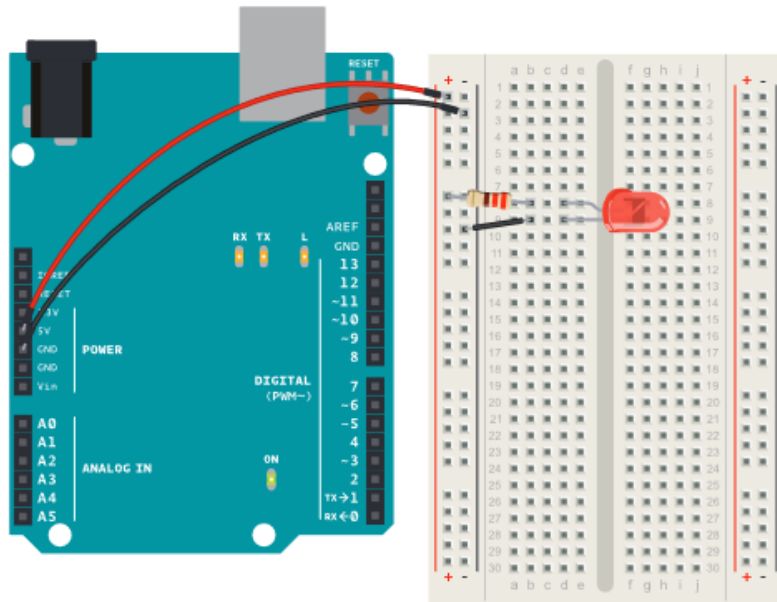


# Premier projet : breadboard

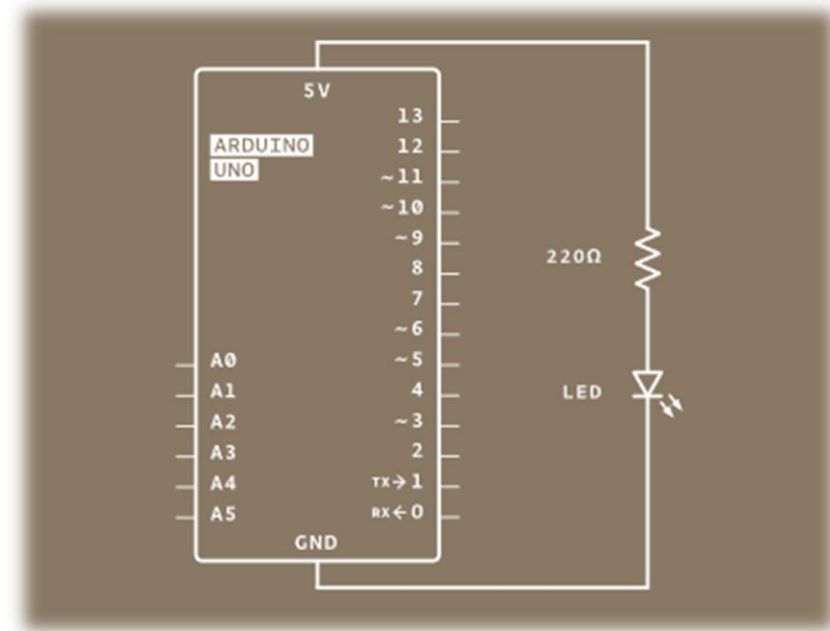


# Premier projet : vues du circuit

A travers cette initiation, vous aurez deux visualisations du circuit électrique à construire :



Une vue du dessus de la breadboard



Une vue schématique du circuit

(voir diapo suivante)

# Premier projet : Symboles

---



UNCONNECTED WIRES



MOTOR



RESISTOR



MOSFET



GROUND



PUSHBUTTON



DIODE



POTENTIOMETER



TILT SWITCH



CONNECTED WIRES



TRANSISTOR



POLARIZED CAPACITOR



CAPACITOR



PHOTO RESISTOR



PIEZO

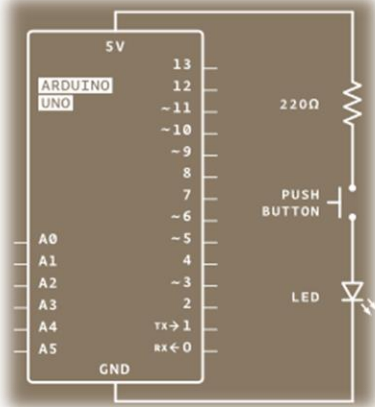
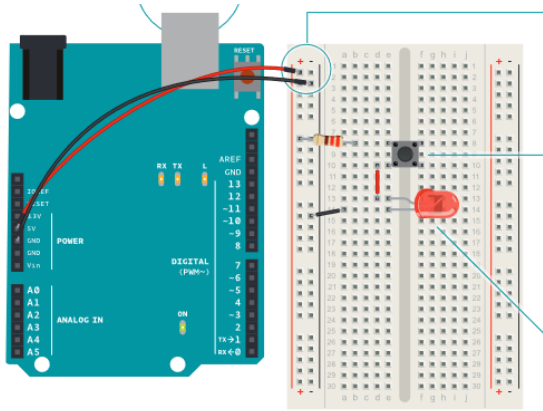


BATTERY



LED

# Premier projet : composants

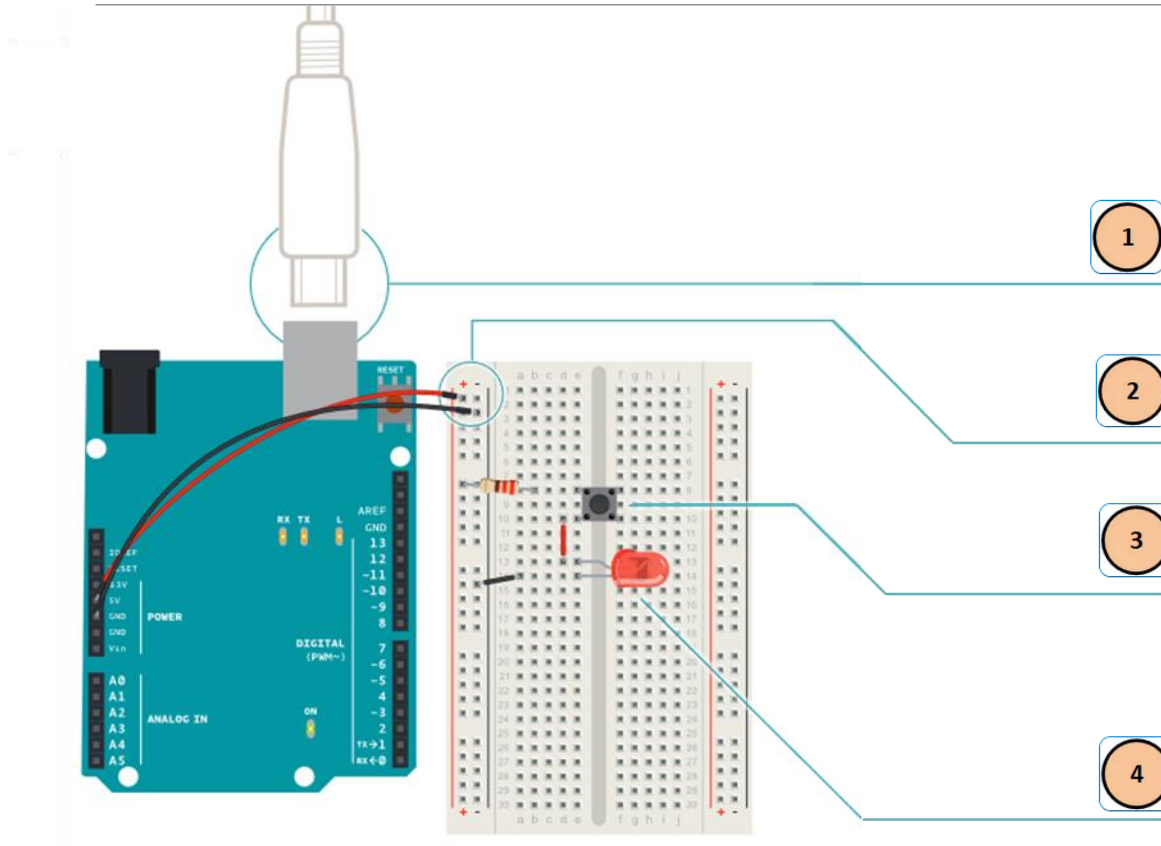


Pour faire ce premier schema, il vous faut :

- Une LED (light-emitting diode) qui transforme l'énergie électrique en énergie lumineuse.
- Une résistance qui résiste au courant électrique, et réduit donc l'énergie électrique circulant après lui.
- Un interrupteur bouton qui permet au courant de passé quand fermé/enfoncé.



# Premier projet : montage



1

Si l'arduino est connecté à une batterie ou à un PC, déconnecter avant de faire le montage

2

Connecter un fil rouge au pin 5V du circuit imprimé Arduino, puis le relier à la rangée du pole + sur le breadboard. Connecter un fil noir au pin GND (terre) du circuit imprimé Arduino, puis le relier à la rangée du pole - sur le breadboard. Le courant circule dans les deux colonnes de gauche.

3

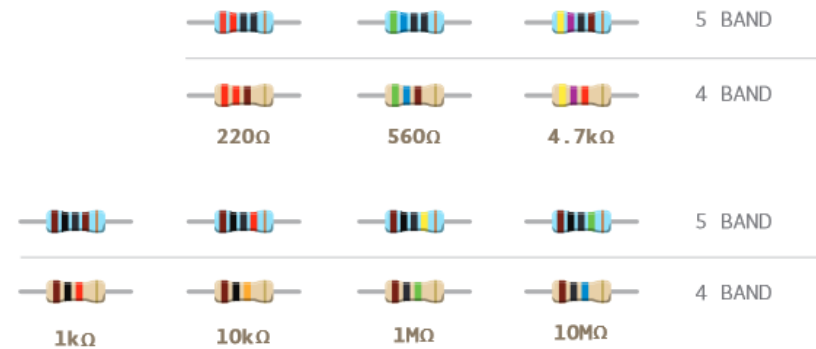
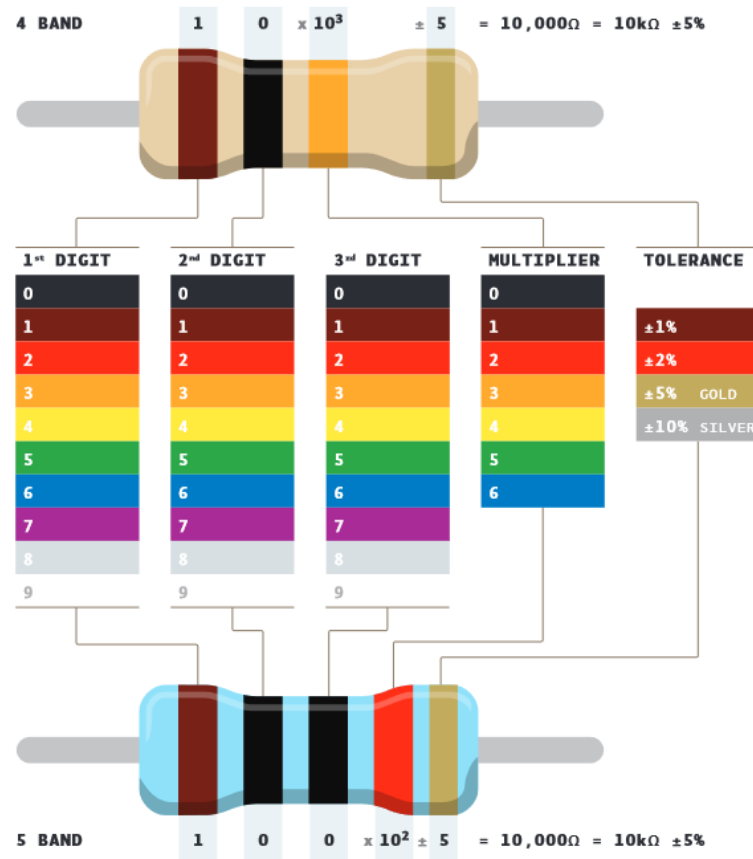
Connecter l'interrupteur a cheval entre les deux parties du breadboard, deux pieds dans la partie de gauche et deux pieds dans la partie de droite.

4

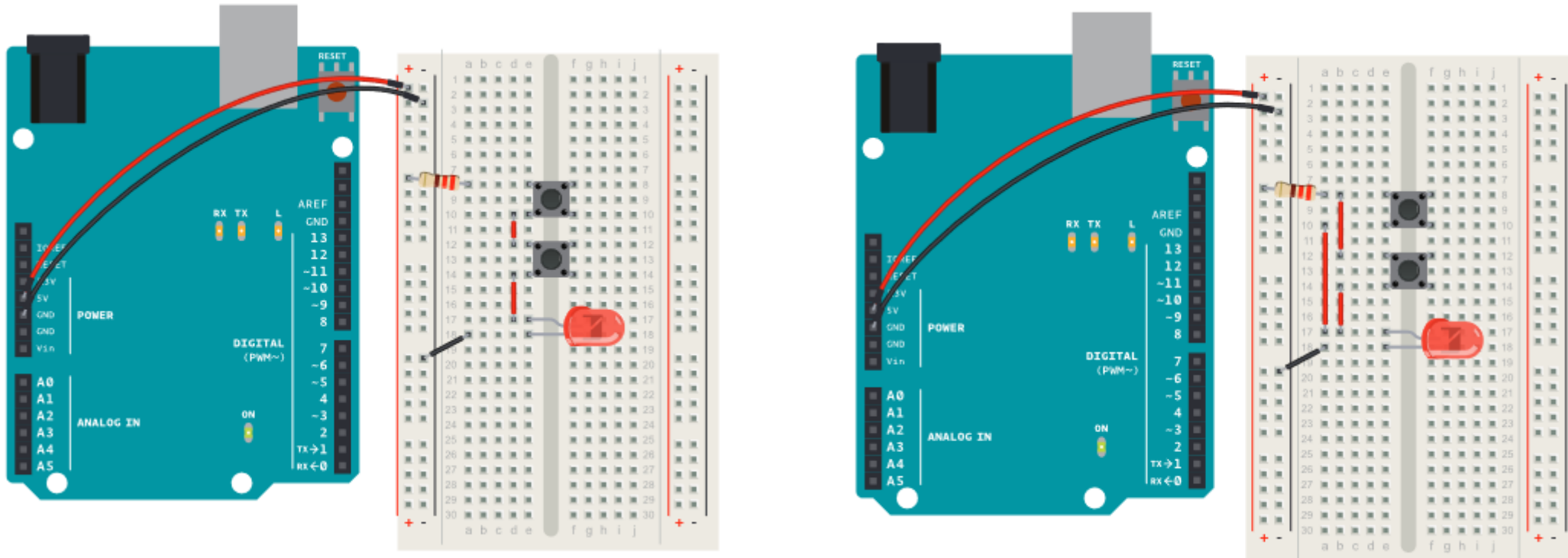
Utiliser une resistance de 220 Ohms pour connecter un coté de l'interrupteur avec le pole +. De l'autre coté de l'interrupteur, connecter l'anode de la LED (la plus grande tige de métal). Enfin, connecter la cathode (la plus petit tige de métal) au pole - (ajouter un fil si nécessaire). Maintenant, connecter l'Arduino en USB pour avoir du courant, et tester.



# Lire une résistance



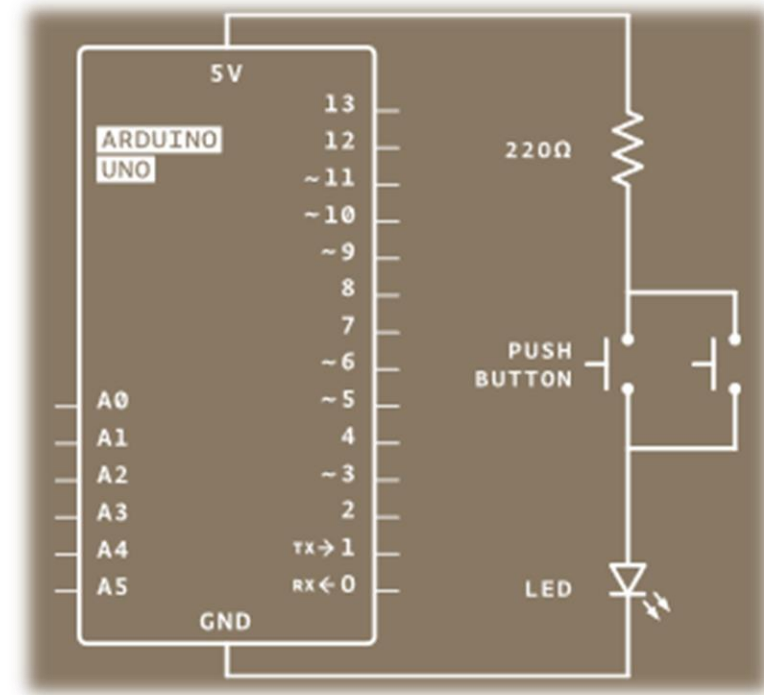
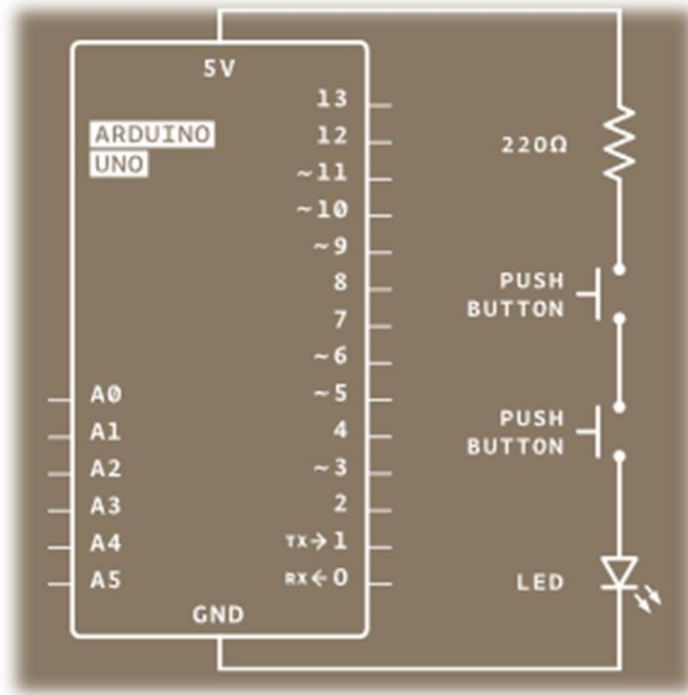
# Premier projet : montage en série vs montage en parallèle



Quelle différence entre les deux ?

# Premier projet : série vs parallèle

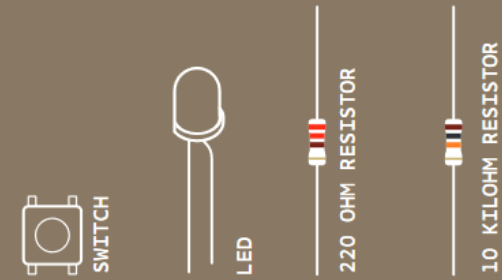
---





# Deuxième projet

---

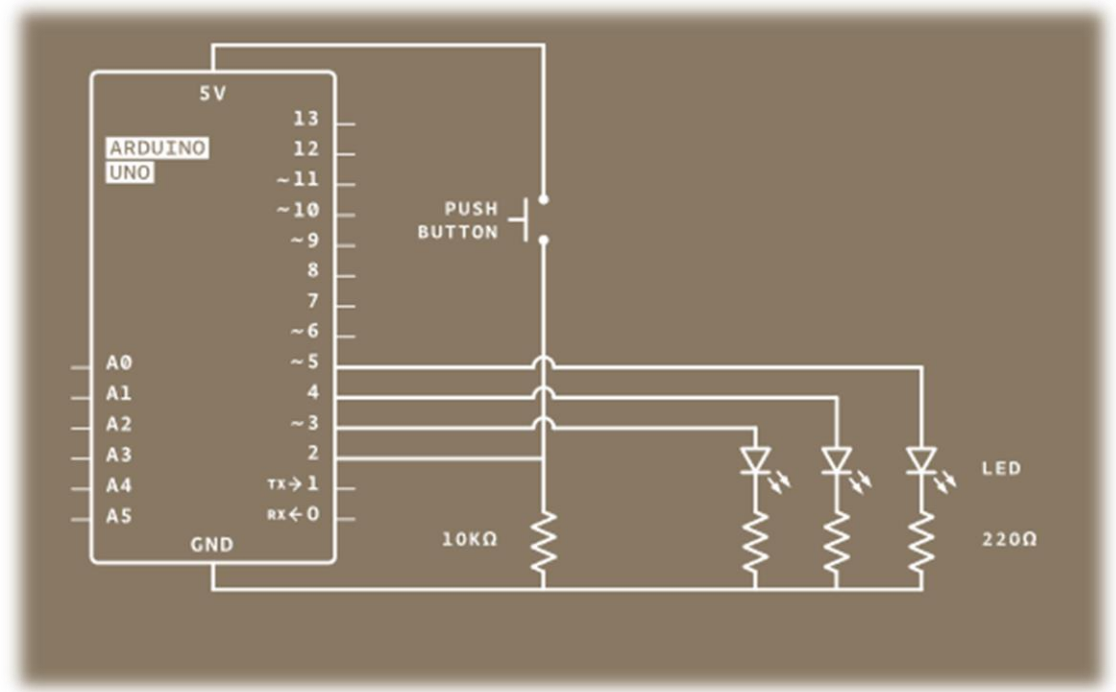
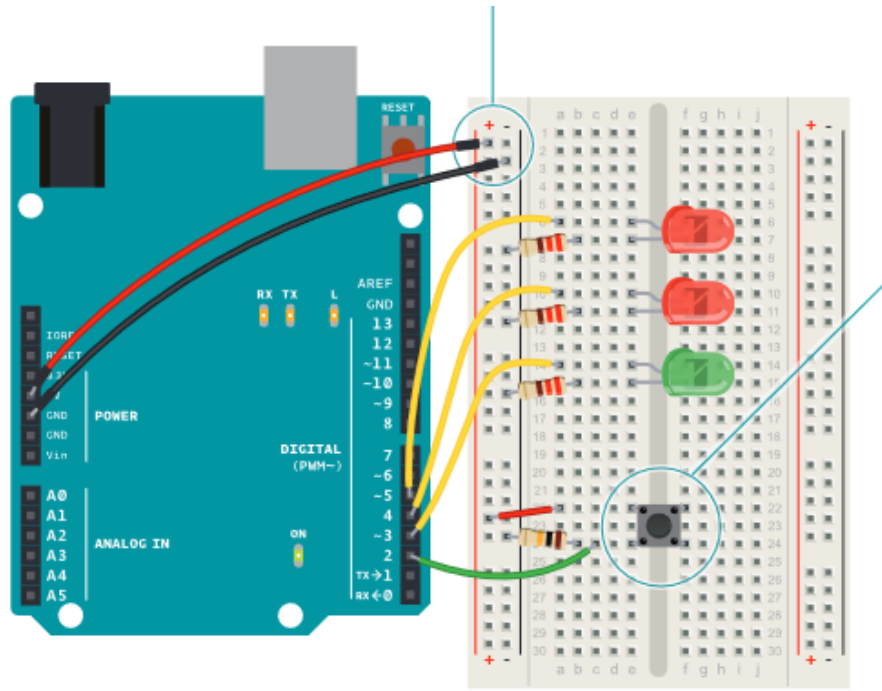


---

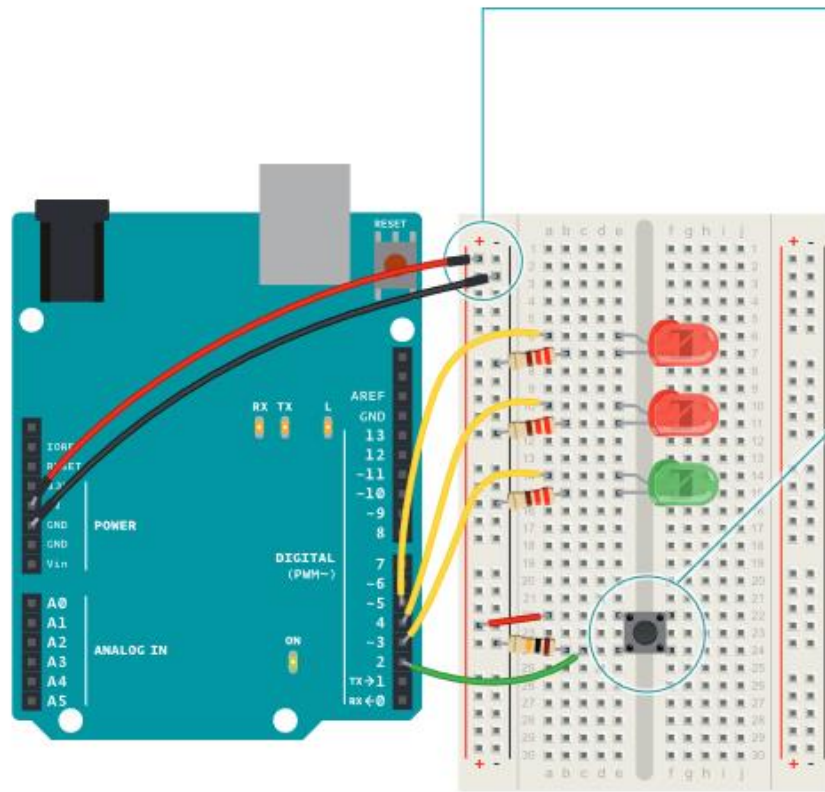
INGREDIENTS

# Deuxième projet : circuit

- Ici, je vous propose de faire un deuxième circuit simple mais avec du code :
  - Quand je n'appuie pas sur le bouton, la LED verte reste allumée. Sinon, les deux autres LEDs clignotent.
  - Le circuit :



# Deuxième projet : montage



Connecter un fil rouge au pin 5V du circuit imprimé Arduino, puis le relier à la rangée du pole + sur le breadboard.

Connecter un fil noir au pin GND (terre) du circuit imprimé Arduino, puis le relier à la rangée du pole - sur le breadboard.

Placer deux LEDs rouges et une LED verte sur le breadboard.

Attacher chaque cathode (tige courte) à une résistance de 220 Ohms et au pole -. Attacher l'anode (tige longue) de la LED verte au pin 3, et l'anode des LEDs rouge aux pin 4 et 5.

Remettre l'interrupteur à cheval sur les deux parties de la breadboard. Attacher une partie de l'interrupteur au pole +, l'autre à une résistance de 10 000 ohms puis au pole - ainsi qu'au pin 2.

Et maintenant, on code.

# Deuxième projet : code

---

- Chaque programme Arduino a deux fonctions principales appelées `setup()` et `loop()`.
  - `setup()` : fonction lancée une fois, au lancement du Arduino, pour configurer l'appareil (configuration des pin, des appareils, etc.).
  - `loop()` : fonction qui fonctionne continuellement après la configuration faite par la fonction `setup()`. C'est ici que les signaux sont lues et interprétés, et les actions à faire en fonction de ce signal sont aussi codées ici.

```
void setup(){  
}  
  
void loop(){  
}
```

- Une fonction est un bout de code que l'on peut écrire et exécuter pour effectuer certaines tâches.
- Nous allons aussi déclarer des variables : ce sont des noms auxquels on associe des valeurs.
  - Exemple : `int switchState = 0;` → Je stocke la valeur 0 dans la variable `switchState` de type « integer » (un nombre)

# Deuxième projet : code

---

- Je déclare (crée) la variable `switchState` en indiquant son type de données au début (`int` pour « integer ») puis je lui assigne sa valeur avec le signe « = ».
- Ici, je définie si mes pin reçoivent un signal (INPUT) ou émettent un signal (OUTPUT) avec la fonction `pinMode`. Les pin des LEDs (3, 4 et 5) sont des sorties tandis que le pin de l'interrupteur est une entrée.
- Ici, je vérifie le signal que m'envoie le pin 2 (l'interrupteur). J'utilise alors la fonction `digitalRead()` pour voir si le signal est HIGH (1) ou LOW (0), et je stock le signal dans la variable `switchState`.

```
1 int switchState = 0;
```

```
2 void setup(){  
3   pinMode(3,OUTPUT);  
4   pinMode(4,OUTPUT);  
5   pinMode(5,OUTPUT);  
6   pinMode(2,INPUT);  
7 }
```

```
8 void loop(){  
9   switchState = digitalRead(2);  
10  // this is a comment
```

# Deuxième projet : code

- J'utilise ensuite la structure conditionnelle if() pour définir quelles actions exécuter en fonction de la valeur de switchState.
  - Ce premier bloc m'indique que si switchState est égale à LOW (c'est-à-dire que l'interrupteur n'est pas appuyé), j'envoie un signal au pin 3 (LED verte) mais pas aux pin 4 et 5 (LEDs rouges).
- J'utilise ensuite le mot clef else{} qui me permet de définir quelles actions exécuter si la condition définie dans if() n'est pas respectée :
  - C'est-à-dire qu'ici, si switchState est égale à HIGH, j'éteins ma LED verte et j'allume mes LEDs rouges.
  - La fonction delay(250) demande à la fonction d'attendre 250 millisecondes (1/4 seconde) avant de continuer

```
11  if (switchState == LOW) {  
12      // the button is not pressed  
  
13      digitalWrite(3, HIGH); // green LED  
14      digitalWrite(4, LOW);  // red LED  
15      digitalWrite(5, LOW);  // red LED  
16  }
```

```
17  else { // the button is pressed  
18      digitalWrite(3, LOW);  
19      digitalWrite(4, LOW);  
20      digitalWrite(5, HIGH);  
  
21      delay(250); // wait for a quarter second  
22      // toggle the LEDs  
23      digitalWrite(4, HIGH);  
24      digitalWrite(5, LOW);  
25      delay(250); // wait for a quarter second  
  
26  }  
27 } // go back to the beginning of the loop
```

# Deuxième projet : code

---

```
1 int switchState = 0;
2 void setup(){
3   pinMode(3,OUTPUT);
4   pinMode(4,OUTPUT);
5   pinMode(5,OUTPUT);
6   pinMode(2,INPUT);
7 }
8 void loop(){
9   switchState = digitalRead(2);
10  // this is a comment
11  if (switchState == LOW) {
12    // the button is not pressed
13    digitalWrite(3, HIGH); // green LED
14    digitalWrite(4, LOW);  // red LED
15    digitalWrite(5, LOW);  // red LED
16  }
17  else { // the button is pressed
18    digitalWrite(3, LOW);
19    digitalWrite(4, LOW);
20    digitalWrite(5, HIGH);
21    delay(250); // wait for a quarter second
22    // toggle the LEDs
23    digitalWrite(4, HIGH);
24    digitalWrite(5, LOW);
25    delay(250); // wait for a quarter second
26  }
27 } // go back to the beginning of the loop
```

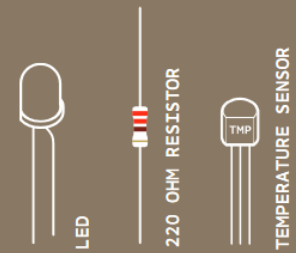
Ca clignote ?

Pour aller plus loin:

- Comment feriez vous pour faire en sorte que les LEDs rouges clignotent dès le lancement du programme ?
- Envisagez vous d'autres configurations ? Plus d'interrupteurs, de LEDs, etc. ?

# Troisième projet

---



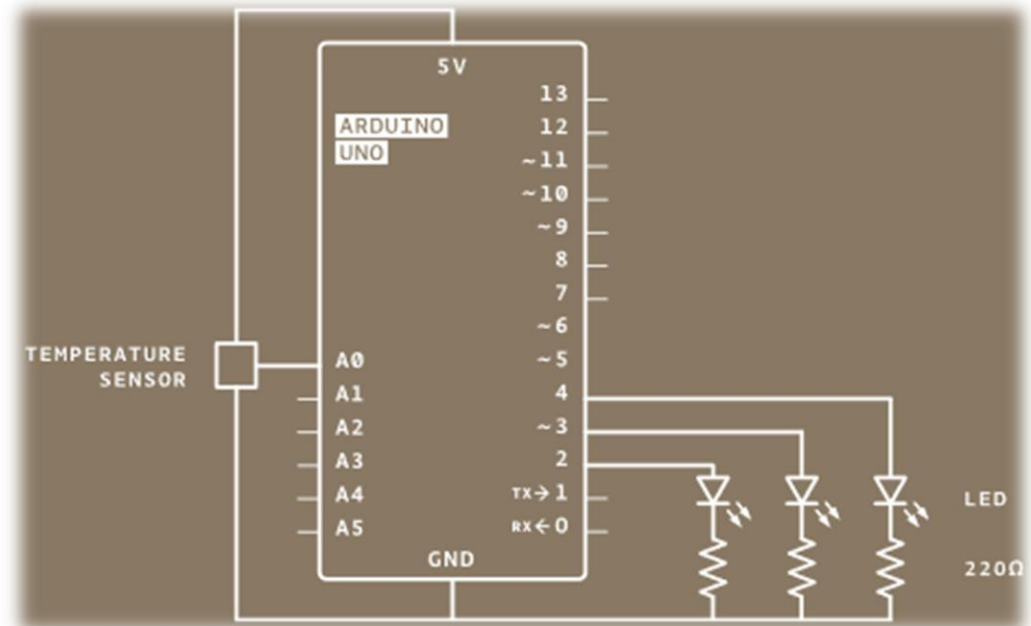
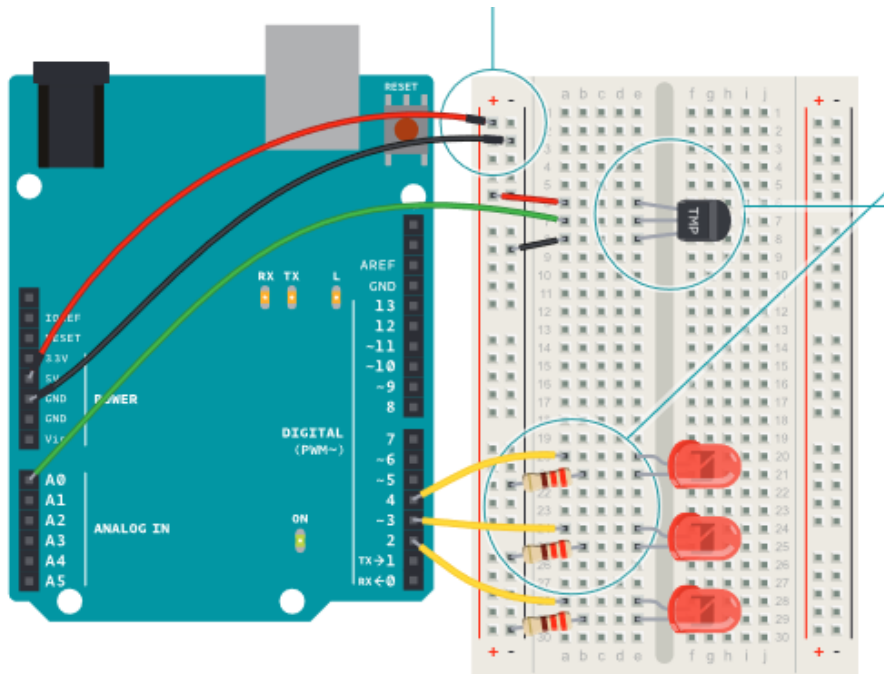
---

INGREDIENTS

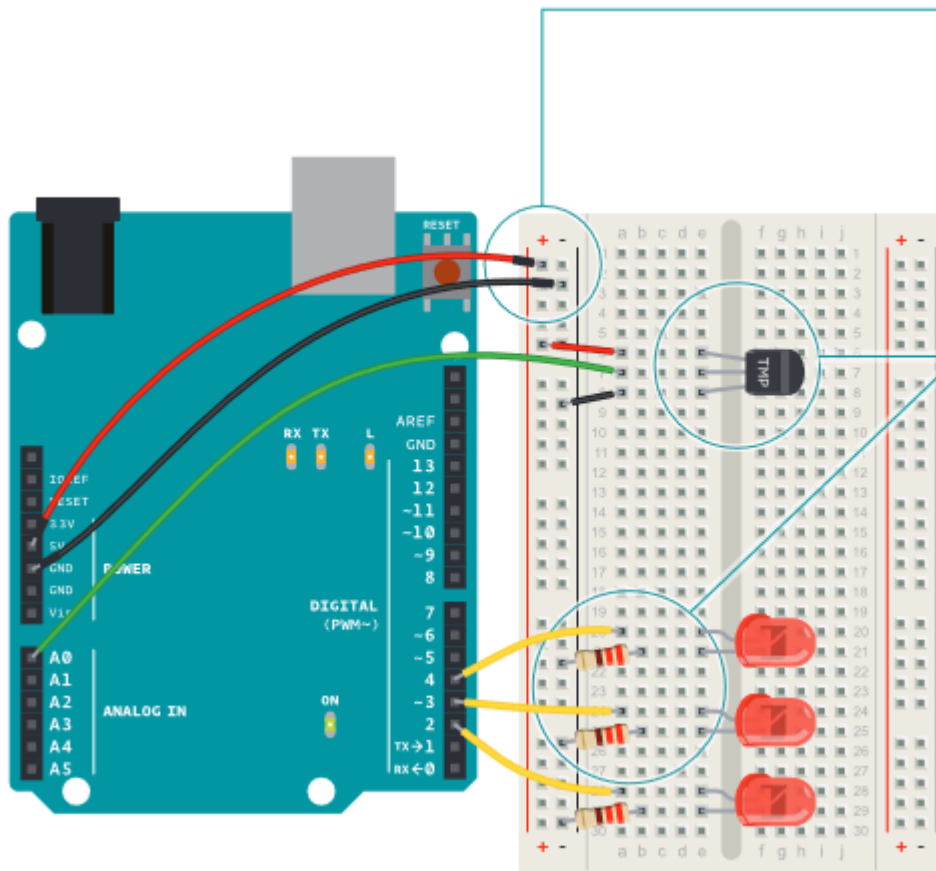


# Troisième projet : circuit

- Ici, je vous propose de faire un troisième et dernier circuit :
  - Cette fois ci, utilisons un capteur de temperature.
  - Le circuit :



# Troisième projet: montage



Comme pour avant, connecter les poles + et - à la breadbord.

Attacher la cathode (petite tige) de chaque LED à une résistance de 220 ohms puis au pole - puis attacher chaque anode (grande tige) aux pin 2 à 4.

Placer le capteur de température avec la partie ronde faisant dos à la breadbord.

En ayant la face plate du capteur en face de soi :

- Attacher le pin de gauche au pole + et le pin de droite au pole -.
- Attacher le pin du milieu au pin A0 du Arduino.

# Troisième projet : code

---

- Je déclare les constantes sensorPin et baselineTemp. Des constantes sont comme des variables, mais leur valeur ne peut changer.
- J'exécute la commande serial.begin(9600) au lancement du programme pour observer les valeurs relevées par le capteur directement sur l'ordinateur, 9600 étant le nombre de bit transmis par seconde.
- Ici, au lieu d'écrire trois fois pinMode et digitalWrite, j'utilise la déclaration for qui me permet de parcourir tout les pin du nombre 2 à 4 pour ensuite les paramétrer en pinMode « OUTPUT » et digitalWrite « LOW »

```
1 const int sensorPin = A0;  
2 const float baselineTemp = 20.0;
```

```
3 void setup(){  
4   Serial.begin(9600); // open a serial port
```

```
5   for(int pinNumber = 2; pinNumber<5; pinNumber++){  
6     pinMode(pinNumber,OUTPUT);  
7     digitalWrite(pinNumber, LOW);  
8   }  
9 }
```

# Troisième projet : code

---

- Je déclare la variable `sensorVal` qui prend la valeur mesurée par le capteur de température via la fonction `analogRead()`. C'est une valeur en 0 et 1023 qui représente le voltage sur le pin.
- J'utilise la fonction `Serial.print()` pour afficher dans la console la valeur mesurée par le capteur
- Je convertie la valeur mesurée en volts.
- Et j'affiche le voltage calculé.

```
10 void loop(){  
11   int sensorVal = analogRead(sensorPin);
```

```
12   Serial.print("Sensor Value: ");  
13   Serial.print(sensorVal);
```

```
14   // convert the ADC reading to voltage  
15   float voltage = (sensorVal/1024.0) * 5.0;
```

```
16   Serial.print(", Volts: ");  
17   Serial.print(voltage);
```

# Troisième projet : code

---

- On va pas aller dans le détail, mais un changement de 10 mV dans le capteur correspond a un changement de 1 °C. Donc on calcule la température réelle à partir de ça et on l'affiche avec `Serial.println()`.
- Maintenant que la réelle température est calculée, on peut programmer l'appareil. Si la température mesurée est inférieur à la température ambiante, rien ne s'allume.
- Si la température est supérieur de 2 à 4 °C au dessus de la temperature ambiante, une LED s'allume.

```
18  Serial.print(", degrees C: ");
19  // convert the voltage to temperature in degrees
20  float temperature = (voltage - .5) * 100;
21  Serial.println(temperature);
```

```
22  if(temperature < baselineTemp){
23      digitalWrite(2, LOW);
24      digitalWrite(3, LOW);
25      digitalWrite(4, LOW);
```

```
26  }else if(temperature >= baselineTemp+2 &&
        temperature < baselineTemp+4){
27      digitalWrite(2, HIGH);
28      digitalWrite(3, LOW);
29      digitalWrite(4, LOW);
```

# Troisième projet : code

---

- Si la température est supérieur de 4 à 6 °C au dessus de la temperature ambiante, deux LEDs s'allument.
- Si la température est supérieur de 6°C au dessus de la temperature ambiante, trois LEDs s'allument.
- On ajoute un delay() de 1 milliseconde pour ne pas obtenir de valeurs aberrantes.

```
30 }else if(temperature >= baselineTemp+4 &&  
    temperature < baselineTemp+6){  
31     digitalWrite(2, HIGH);  
32     digitalWrite(3, HIGH);  
33     digitalWrite(4, LOW);
```

```
34 }else if(temperature >= baselineTemp+6){  
35     digitalWrite(2, HIGH);  
36     digitalWrite(3, HIGH);  
37     digitalWrite(4, HIGH);
```

```
38 }  
39 delay(1);  
40 }
```

# Troisième projet : code

---

```
1 const int sensorPin = A0;
2 const float baselineTemp = 20.0;
3 void setup(){
4   Serial.begin(9600); // open a serial port
5   for(int pinNumber = 2; pinNumber<5; pinNumber++){
6     pinMode(pinNumber,OUTPUT);
7     digitalWrite(pinNumber, LOW);
8   }
9 }
10 void loop(){
11   int sensorVal = analogRead(sensorPin);
12   Serial.print("Sensor Value: ");
13   Serial.print(sensorVal);
14   // convert the ADC reading to voltage
15   float voltage = (sensorVal/1024.0) * 5.0;
16   Serial.print(", Volts: ");
17   Serial.print(voltage);
18   Serial.print(", degrees C: ");
19   // convert the voltage to temperature in degrees
20   float temperature = (voltage - .5) * 100;
21   Serial.println(temperature);
```

```
22   if(temperature < baselineTemp){
23     digitalWrite(2, LOW);
24     digitalWrite(3, LOW);
25     digitalWrite(4, LOW);
26   }else if(temperature >= baselineTemp+2 &&
27     temperature < baselineTemp+4){
28     digitalWrite(2, HIGH);
29     digitalWrite(3, LOW);
30     digitalWrite(4, LOW);
31   }else if(temperature >= baselineTemp+4 &&
32     temperature < baselineTemp+6){
33     digitalWrite(2, HIGH);
34     digitalWrite(3, HIGH);
35     digitalWrite(4, LOW);
36   }else if(temperature >= baselineTemp+6){
37     digitalWrite(2, HIGH);
38     digitalWrite(3, HIGH);
39     digitalWrite(4, HIGH);
40   }
```

Que voyez vous?

Pour aller plus loin:

- Afficher certains messages en fonction de chaque élévation de température ?

# Arduino : présentation des composants

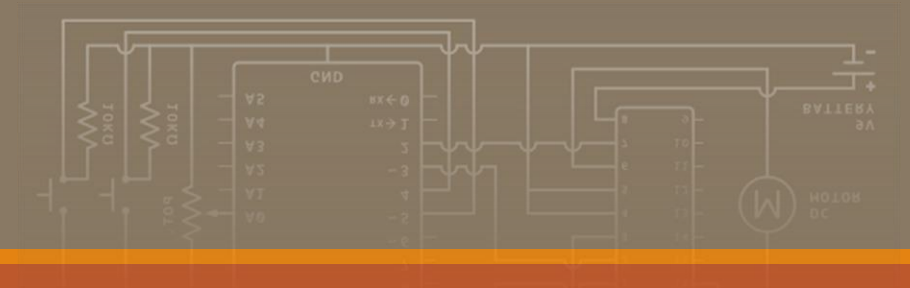
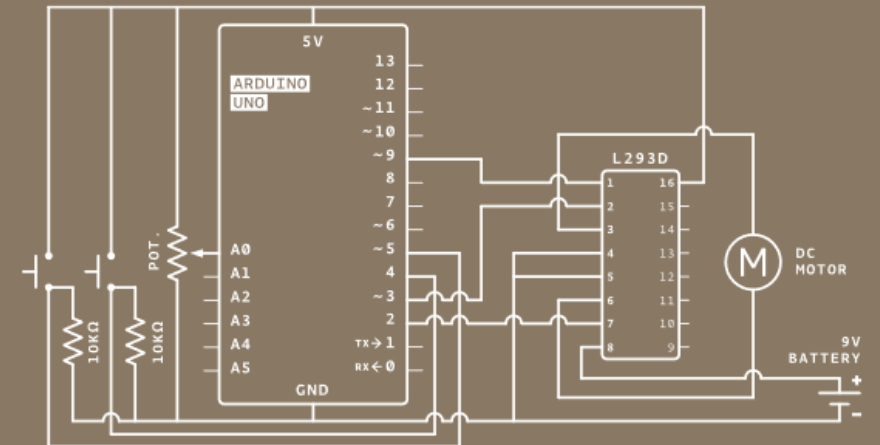
## Projets :

Scanneur de produits à code barre

Détecteur d'incendie en forêt

Pot de fleur connecté

Détecteur de niveau d'eau dans un réservoir





# Lecteur de code barre

---

[https://wiki.dfrobot.com/Barcode\\_Reader\\_Scanner\\_Module-CCD\\_Camera\\_SKU\\_DFR0314#target\\_6](https://wiki.dfrobot.com/Barcode_Reader_Scanner_Module-CCD_Camera_SKU_DFR0314#target_6)

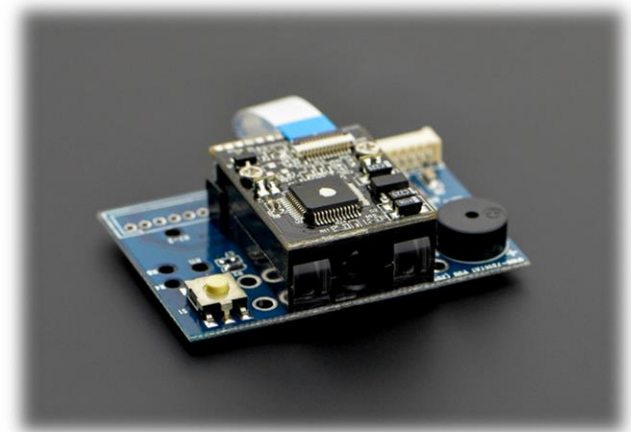
```
// Définition des broches RX et TX
#define RX_PIN 2
#define TX_PIN 3

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600); // Initialise la communication série avec le lecteur de code-barres
}

void loop() {
  if (Serial1.available() > 0) {
    // Lecture des données du lecteur de code-barres
    String codeBarre = Serial1.readStringUntil('\n');

    // Affichage du code-barres sur le moniteur série
    Serial.println("Code-barres : " + codeBarre);

    // Vous pouvez ajouter ici la logique pour traiter le code-barres (par exemple, envoyer à une base de
    données, allumer une LED, etc.)
  }
}
```



# Capteur MQ-2

[https://wiki.dfrobot.com/Analog\\_Gas\\_Sensor\\_SKU\\_SEN0127](https://wiki.dfrobot.com/Analog_Gas_Sensor_SKU_SEN0127)

```
int pinMQ2 = A0; // Broche analogique sur laquelle le capteur MQ-2 est connecté
int seuilFumee = 300; // Ajustez ce seuil en fonction de votre environnement

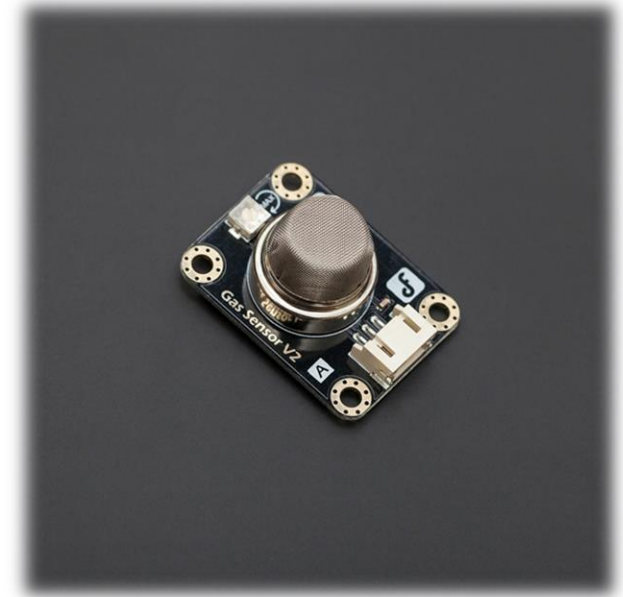
void setup() {
  Serial.begin(9600);
}

void loop() {
  int valeurCapteur = analogRead(pinMQ2);

  // Affiche la valeur du capteur sur le moniteur série
  Serial.println("Valeur du capteur : " + String(valeurCapteur));

  // Vérifie si la valeur dépasse le seuil de fumée
  if (valeurCapteur > seuilFumee) {
    Serial.println("Détection de fumée ! Alerte incendie !");
    // Vous pouvez ajouter ici la logique pour déclencher une alerte ou prendre d'autres mesures nécessaires
  }

  delay(1000); // Attend une seconde entre chaque lecture du capteur
}
```



# Capteur à humidité

<https://www.dfrobot.com/product-174.html>

```
// Broche analogique sur laquelle le capteur d'humidité est connecté
int pinCapteurHumidite = A0;

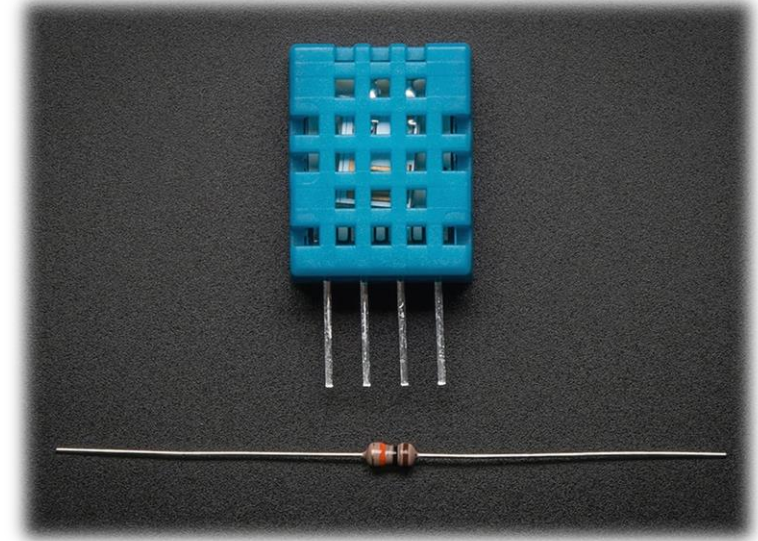
void setup() {
  Serial.begin(9600);
}

void loop() {
  // Lit la valeur analogique du capteur d'humidité
  int valeurHumidite = analogRead(pinCapteurHumidite);

  // Convertit la valeur analogique en pourcentage d'humidité
  int pourcentageHumidite = map(valeurHumidite, 0, 1023, 0, 100);

  // Affiche la valeur sur le moniteur série
  Serial.print("Humidité du sol : ");
  Serial.print(pourcentageHumidite);
  Serial.println("%");

  delay(1000); // Attend une seconde entre chaque lecture
}
```



# Module à ultrason

---

<https://www.dfrobot.com/product-1935.html>

```
// Broche analogique sur laquelle le capteur d'humidité est connecté
int pinCapteurHumidite = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Lit la valeur analogique du capteur d'humidité
  int valeurHumidite = analogRead(pinCapteurHumidite);

  // Convertit la valeur analogique en pourcentage d'humidité
  int pourcentageHumidite = map(valeurHumidite, 0, 1023, 0, 100);

  // Affiche la valeur sur le moniteur série
  Serial.print("Humidité du sol : ");
  Serial.print(pourcentageHumidite);
  Serial.println("%");

  delay(1000); // Attend une seconde entre chaque lecture
}
```

