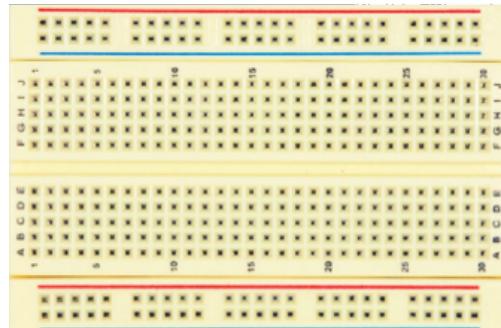
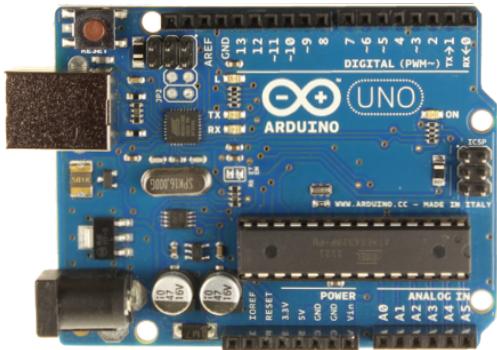
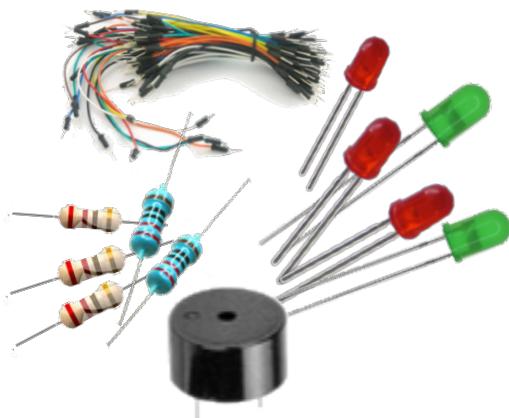


Arduino

*Transmettre aux
enfants le goût de la
programmation, de la
robotique et de
l'ingénierie*

Atelier Arduino Présentations



Si cet atelier vous a plu, vous pouvez refaire cet atelier chez vous avec un budget de moins de 100 €.

Il requiert :

- un Arduino Uno ou un clone (environ 15 €)
- une planche d'essai (breadboard)
- quelques LEDs
- un buzzer
- quelques résistances 220 Ω
- des fils
- pour les ateliers plus avancés une photo-résistance et une résistance 330 Ω

Il existe un starter kit officiel Arduino mais vous pouvez trouver ces composants dans presque tous les starter kits (nous avons des starter kits Farnell et Funduino).

Les supports sont disponibles sur le dépôt de documents de Devoxx4kids France

<http://devoxx4kidsfr.github.io/materials/ateliers/arduino/index.html>

Vous trouverez des informations plus générales sur nos activités et les différents ateliers sur le site Web de Devoxx4Kids France

<http://www.devoxx4kids.org/france/ateliers/arduino/>



Le micro-contrôleur

Le **micro-contrôleur** est une sorte de cerveau qui pilote les appareils électroniques. On en trouve dans les robots, certains jouets, des machines à laver le linge, des fours électriques ...

L'ordinateur ou le smartphone a aussi une carte mais beaucoup plus puissante que l'Arduino.

Il sait faire des opérations simples comme compter et il peut envoyer et recevoir des informations.

Pour faire du son, de la lumière, écrire des textes, nous permettre de contrôler l'appareil on va utiliser des **composants** tels que des LEDs, des buzzers, des écrans LCD, des boutons poussoirs.

Les LED sont des petites lampes. Il y en a de plusieurs couleurs.



Des LEDs



Un buzzer

Celui ci est un buzzer. Il joue des sons.

Pour que l'Arduino sache quoi faire faire aux composants nous allons écrire un **programme** qui indique à l'Arduino ce qu'il doit faire.

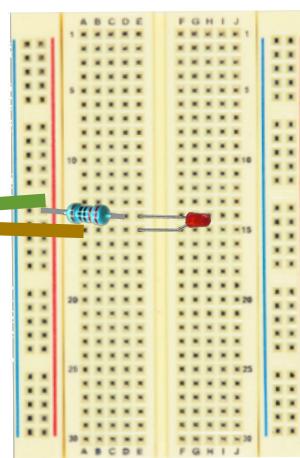


On aura besoin de l'ordinateur

- pour écrire les programmes et les envoyer à l'Arduino
- pour fournir du courant à l'Arduino



Le micro-contrôleur



La "planche à pain"

L'Arduino ne sait pas faire grand chose, mais il sait envoyer du courant aux composants par les **connecteurs** (ou **pins**) qui sont sur les côtés. Dans programme pour Arduino, on va indiquer sur quel pin on envoie du courant et quelle quantité.

Pour que tous les composants reçoivent du courant, on va utiliser la **planche à pain**, des **composants** et des **fils** pour faire un **circuit** qui part de l'Arduino, passe dans tous les composants et revient dans l'Arduino.



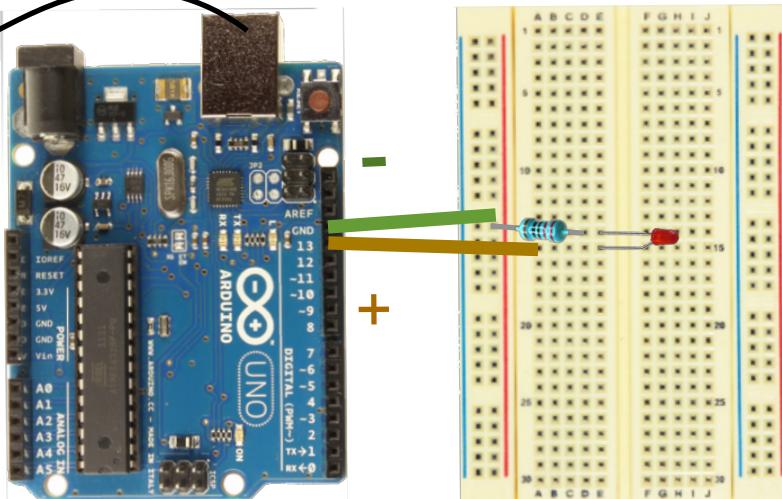
Plein de fils

Dans l'exemple, le programme va envoyer du courant sur le pin 13 du côté du + et la LED va recevoir du courant et s'allumer. Le courant va traverser la LED et le composant en bleu et reviendra dans l'Arduino du côté -.

Le **micro-contrôleur** a besoin de courant pour fonctionner. On ne peut pas le brancher directement sur la prise de courant qui fourni beaucoup trop de courant et pas sous la bonne forme.



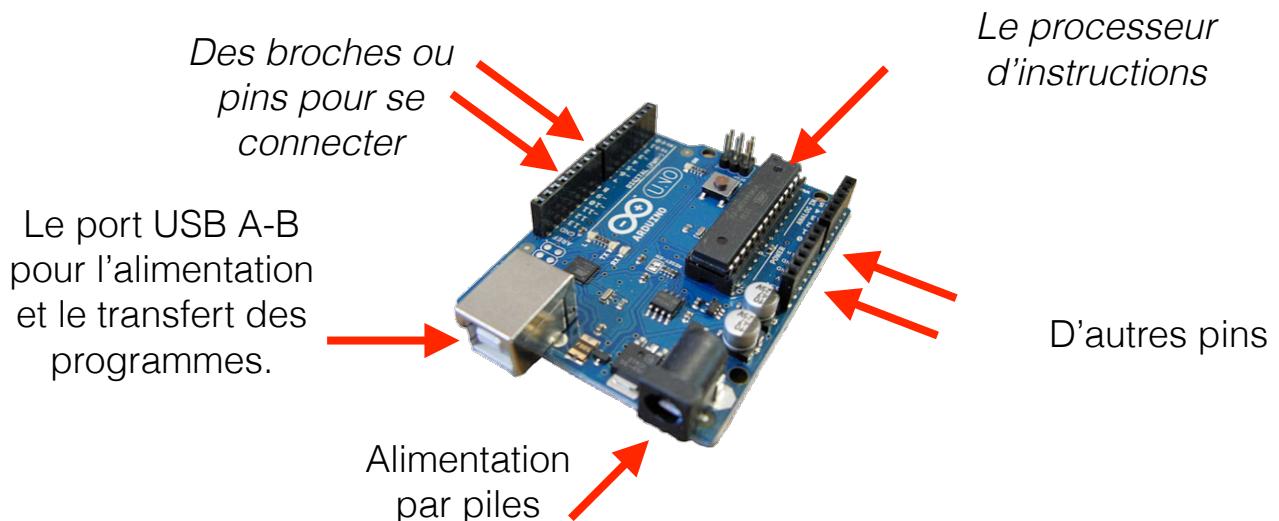
On va lui fournir le courant par le **câble USB** de l'ordinateur. On pourrait aussi utiliser des piles.



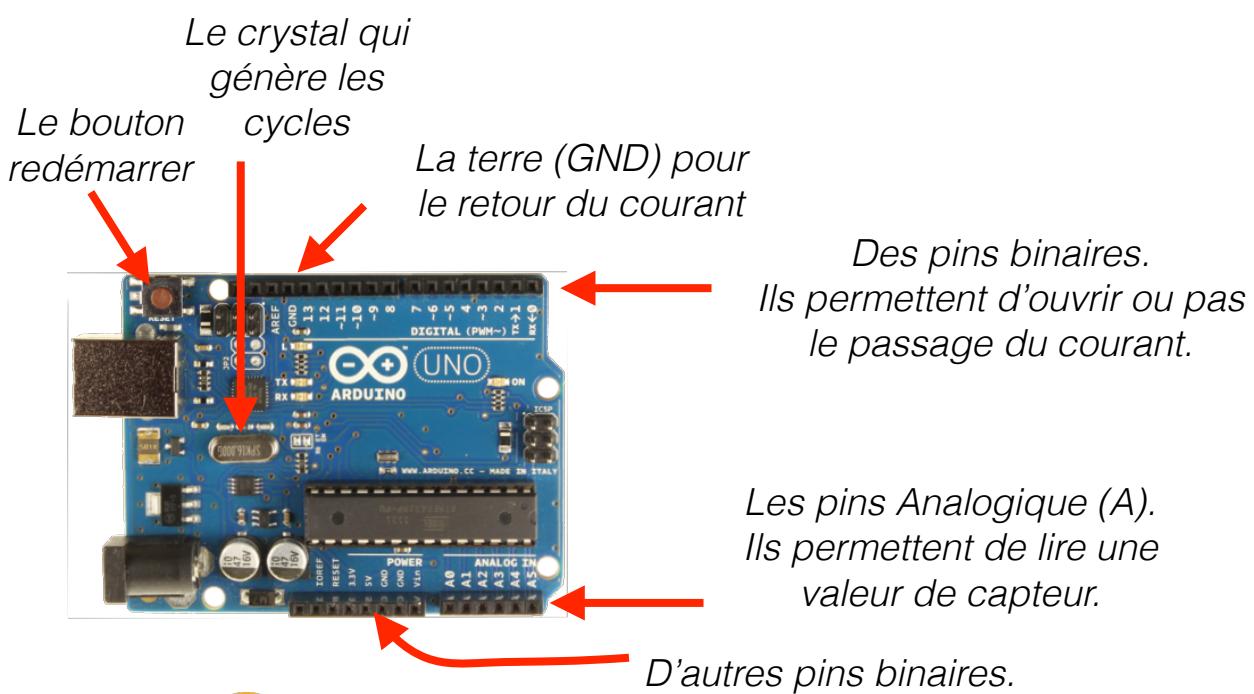
Pour que tous les composants reçoivent du courant, on va faire un **circuit** qui part de l'Arduino par les **connecteurs** (ou pins) qui se trouvent sur les côtés, passe dans tous les composants et revient dans l'Arduino.

On va tout connecter en utilisant la **planque d'essai** (la plaque avec plein de trous) qu'on appelle aussi **planche à pain**. Dans les appareils électroniques, le circuit est soudé pour être plus solide.

Les **connecteurs** sur les côtés sont connectées à l'alimentation électrique et au processeur d'instructions. Il va utiliser le programme pour déterminer quel courant est envoyé à quelle broche.



Le **crystal** bat à un rythme régulier et à chaque battement le **processeur** exécute une **instruction** (un petit bout du programme). Il va lire la quantité de courant sur un connecteur ou permettre/interdire le passage du courant sur le connecteur. Les connecteurs binaires sont 0 ou 1 (fermé ou ouvert). Certains connecteurs sont **analogiques**. Ils permettent de gérer plusieurs valeurs.



Nous allons commencer par vérifier que le matériel fonctionne et que l'on peut le programmer depuis l'ordinateur. Nous allons utiliser la LED de contrôle qui se trouve sur l'Arduino et charger un programme fourni avec l'appareil qui la fait clignoter.

Nous aurons besoin de



L'Arduino Uno



Le cable USB



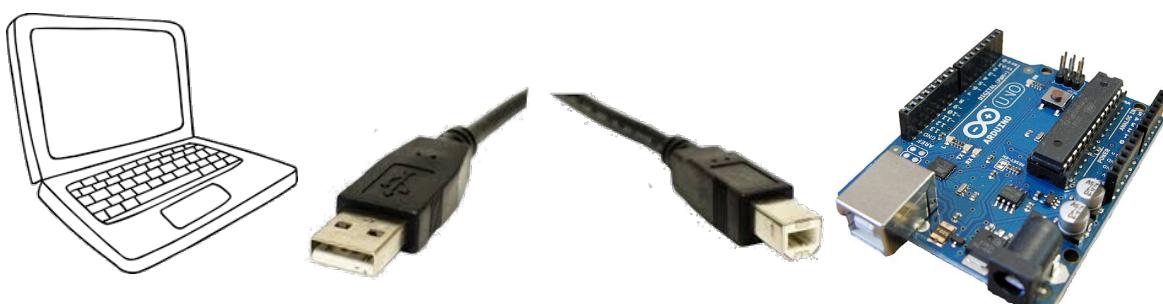
L'ordinateur



L'éditeur de programme

Connecte le cable à l'ordinateur d'un côté et l'Arduino de l'autre. Le cable a 2 fonctions :

- il fournit du **courant** à l'Arduino
- il **transmet le programme** à l'Arduino



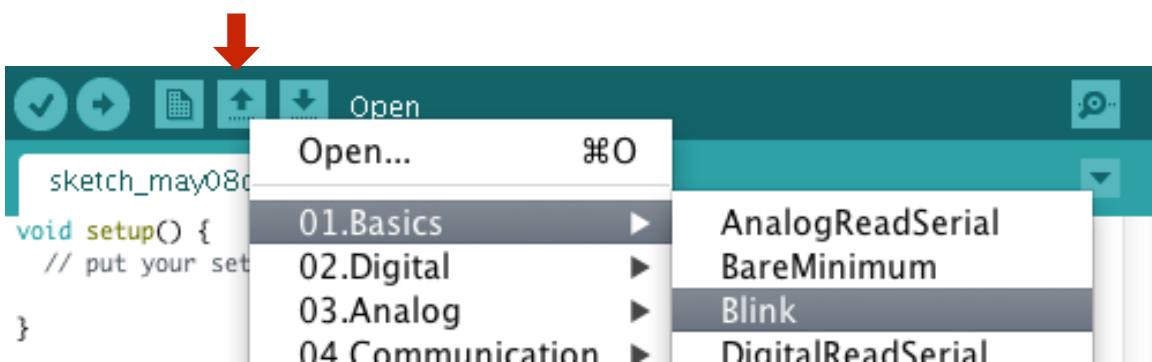
Lorsqu'il reçoit du courant l'Arduino démarre et fait clignoter la LED intégrée pour le faire savoir. Elle se trouve juste à côté du pin 13.

Pour le moment l'Arduino ne sait rien faire. Il attend un programme. Pour commencer, on va utiliser un des programmes fourni avec l'Arduino.

Lance l'outil qui a cette icône.



Ouvre le menu Fichier et utilise l'option Ouvre. Ouvre le dossier 01.Basics et choisi Blink.



Tu peux envoyer le programme sur l'Arduino en utilisant le bouton →..



Attend qu'il affiche
“Téléversement terminé” dans
la zone du bas.

La LED doit se remettre à clignoter sans interruption.

L'Arduino a besoin d'un **programme**. C'est une suite d'indications, un peu comme une route à suivre pour aller quelque part ou une recette de cuisine.

On va faire le programme sur l'ordinateur et on le transmettra à l'Arduino.



L'outil qui permet d'écrire le programme s'appelle un **éditeur de programme**. Cherche sur l'ordinateur le logiciel qui a cette icône et lance le.

L'éditeur est fourni avec des exemples de programmes.

Ouvre le menu Fichier et utilise l'option Ouvre.
Ouvre le dossier 01.Basics et choisi Blink.



Le programme Blink est maintenant chargé dans l'éditeur.

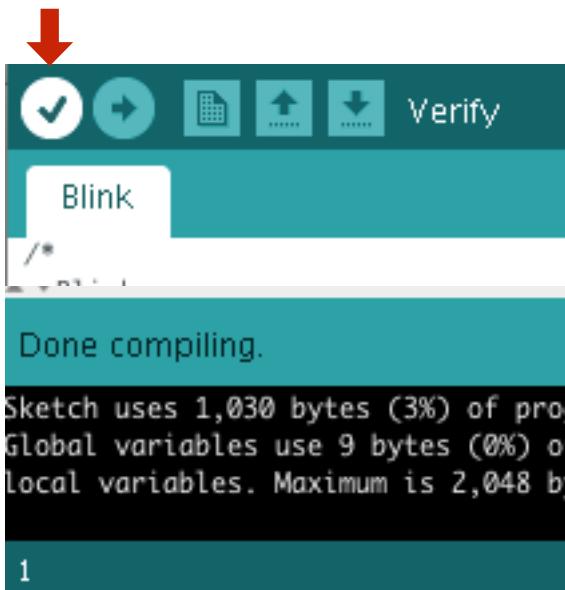
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

Ce programme fait clignoter une LED.

On verra dans la fiche suivante comment ça marche. Pour le moment on va l'utiliser.

Utilise le bouton ✓ pour vérifier que le programme est correct.



Cette action s'appelle **compiler un programme**. Le compilateur lit le programme et le transforme en une suite d'instructions très très simples que l'Arduino comprend.

Le résultat s'affiche dans la zone du bas

Si tout est correct, tu peux maintenant envoyer le programme sur l'Arduino (le téléverser) en utilisant le bouton ➔.



Attend qu'il affiche “Téléversement terminé” dans la zone du bas.

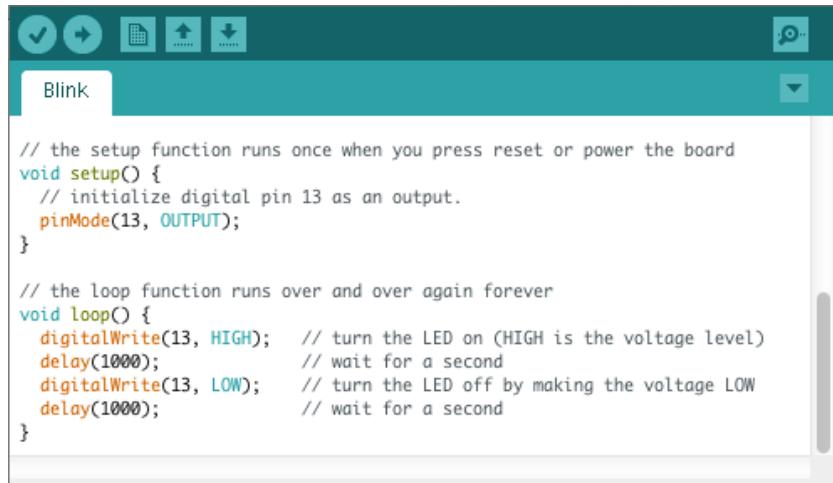
Observe la LED. Elle doit clignoter toutes les secondes.

Cherche la ligne où est `delay`, remplace 1000 par 5000, recompile le programme, téléverse le sur l'Arduino et regarde ce que fait la LED.

La LED doit clignoter plus lentement. `delay` indiquait que l'on doit attendre 1000 milli-secondes (1 seconde). On attend maintenant 5 secondes.

Le **programme Blink** est un des programmes exemple. Il fait clignoter une LED connectée au pin 13.

Ouvre le programme Blink.
Si besoin voit dans la fiche 05 comment recharger le programme Blink.



The screenshot shows the Arduino IDE interface with the title bar "Blink". The code editor contains the following code:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Le comporte 2 blocs principaux :

- `setup` qui dit ce qu'il faut faire 1 seule fois au démarrage
- `loop` qui dit ce qu'il faut faire de manière répétée à chaque cycle

Le programme commence par plusieurs ligne entre /* et */.

Ce sont des **commentaires**, c'est à dire des informations pour le programmeur.

La ligne qui commence par // est aussi un commentaire. Celui ci est sur une seule ligne.

La première ligne utilisable par l'Arduino est

```
void setup() { ... }
```

C'est une **fonction**, on y reviendra plus en détail plus tard.

Ce que fait la fonction est décrit entre les signes { }. Ces signes s'appellent des **accolades**.

`void loop() { ... }` est aussi une fonction.

Ces fonctions contiennent les instructions que l'on donne à l'Arduino. Ce sont aussi des fonctions mais celles-ci sont décrites dans du code qui sera envoyé avec notre programme. On n'a pas à les écrire. On appelle ce code fourni avec l'Arduino une **bibliothèque de fonctions**.

La LED s'éclaire quand il y a du courant et s'éteint quand il n'y en a plus. Pour la faire clignoter on va envoyer un courant, attendre un peu, puis ne plus envoyer le courant, attendre un peu. Et ensuite on recommence, c'est `loop` qui gérera ça.

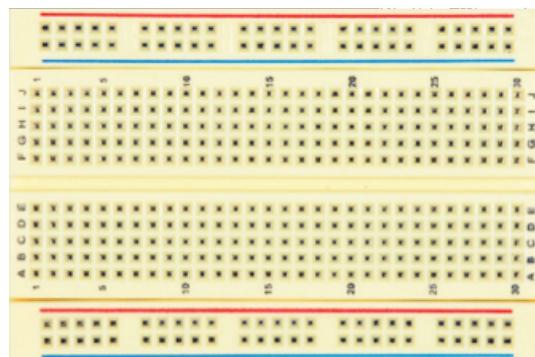
- `pinMode(13, OUTPUT)` indique que l'alimentation est faite par le pin13
- `digitalWrite(13, HIGH)` envoi la quantité maximale de courant sur le pin 13. La LED s'éclaire.
- `delay(1000)` attend 1s (1000 ms)
- `digitalWrite(13, LOW)` envoi la quantité minimale de courant sur le pin 13. La lampe s'éteint.

Les valeurs qui sont entre parenthèses après les fonctions s'appellent des **paramètres**. Par exemple, un paramètre permet d'utiliser `delay` pour attendre plus ou moins longtemps selon la valeur.

- `delay(1000)` attend 1s (1 000 ms)
- `delay(10000)` attend 10s (10 000 ms)

Dans un appareil, les circuits sont fabriqués en usine et soudés pour qu'ils soient solides.

Nous allons utiliser une **plaqué d'essai** qui permet de monter et démonter les circuits. On l'appelle aussi **planche à pain** qui est la traduction exacte du mot anglais breadboard.



La plaque ne fait rien. C'est seulement un support pour les composants.

On va ajouter des **composants** pour faire de la lumière ou du son, observer l'environnement (les capteurs de lumière, de température, ...). Les composants ont des pattes que l'on va piquer dans la planche pour construire le circuit.

Ils ont en général 2 pattes mais parfois plus.



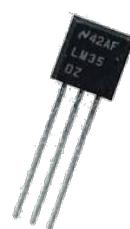
Une résistance 220 Ω



Un buzzer



Un capteur de luminosité

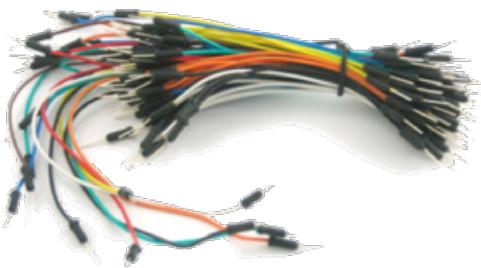


Un capteur de température

Les LEDs ont deux pattes mais elles n'ont pas la même longueur car la LED se place dans un sens particulier.



Une LED

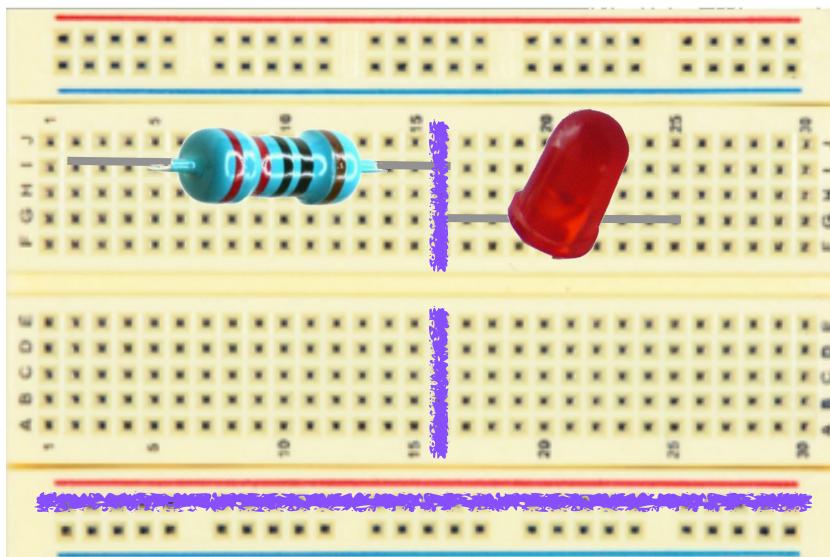


On utilise aussi des **fils** pour connecter la plaque à l'Arduino par exemple ou faire des connections qui ne sont pas faites par la plaque.

A l'intérieur de la plaque certains trous sont reliés entre eux.

Au centre les **rangées** sont connectées verticalement.

La zone vide au milieu sépare les rangées. Les rangées haut et bas ne sont pas connectées.



La résistance est connectée à la LED par rangée 16.

Sur les bords, les **rails** connectent les trous horizontalement. Les rails sont en général utilisé pour se connecter à l'Arduino ou pour connecter différentes parties du circuit.

Nous allons utiliser le programme Blink avec une LED rouge.
Nous aurons besoin de créer un circuit sur la plaque d'essai.

Nous aurons besoin de



L'Arduino Uno



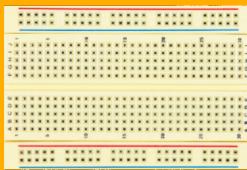
1 LED



Le cable USB



L'ordinateur



La planche
d'essai



1 résistance
 $220\ \Omega$



Le programme
Blink



L'éditeur de
programme

La **LED** émet de la lumière lorsque le courant passe.

Elle a une patte plus longue que l'autre car c'est un composant qui ne fonctionne que dans un sens.

Il y a des LEDs de plusieurs couleurs.



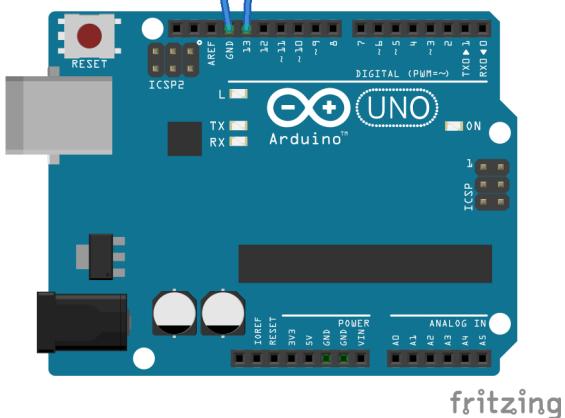
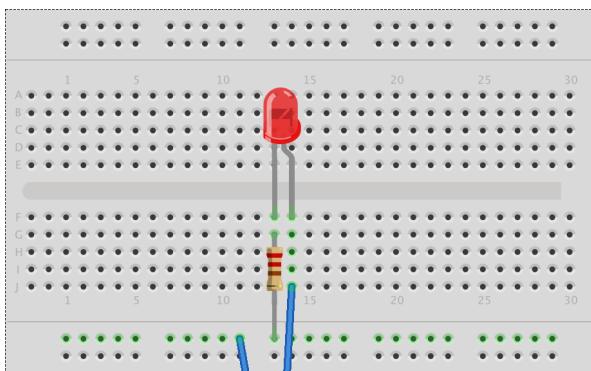


On va aussi avoir besoin d'une **résistance**.

La LED consomme une partie du courant envoyé par l'Arduino (environ 2V sur 5V). La résistance est là pour absorber le reste du courant.

Nous allons voir par la suite comment ça marche mais pour le moment on va essayer.

Attention c'est du matériel électrique. Débranche l'Arduino de l'ordinateur pour modifier le circuit.



- Le fil d'alimentation est sur le pin 13.
- Il doit être sur la même rangée verticale que la patte longue de la LED
- La résistance doit être sur la même rangée que l'autre patte de la LED
- L'autre bout de la résistance doit être sur un des rails horizontaux
- Il faut connecter ce rail au connecteur GND (la terre)

Lorsque le circuit est prêt rebranche l'Arduino sur l'ordinateur si besoin redémarre le.

Un **courant électrique** est un déplacement d'ensemble de porteurs de charges électriques, généralement des **électrons**, au sein d'un matériau conducteur tel que le fil électrique.

C'est un peu similaire au déplacement de l'eau :

La quantité d'eau à un instant donné et dans une longueur du fleuve donnée est quantifiable (par exemple 60 m³ d'eau).

*C'est le **débit**.*



*Le dénivelé, c'est à dire la différence d'altitude entre le point haut et le point bas, peut être assimilé à la **différence de potentiel**.*

C'est le dénivelé qui met l'eau en mouvement. De la même façon, c'est la différence de potentiel qui met les électrons en mouvement.

Le courant est défini par son intensité et sa tension.

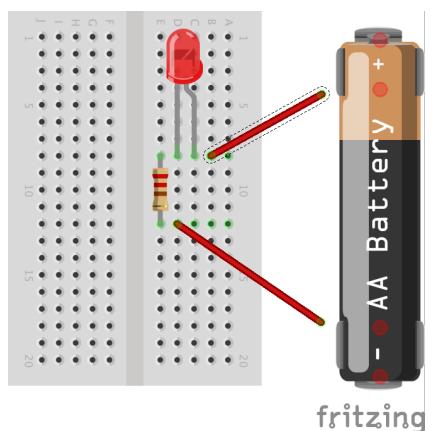
- L'**intensité** du courant électrique est un nombre décrivant le **débit** de charge électrique à travers une surface donnée, par exemple la section d'un fil électrique.
- La **tension** électrique est la circulation du champ électrique le long d'un circuit. Dans la pratique, c'est la **différence de potentiel** électrique entre deux points d'un circuit électrique.

*Elle se mesure en **ampères**, (symbole **A**).*

*Elle se mesure en **volts**, (symbole **V**).*

Il faut un **générateur** qui permet aux charges de se mettre en mouvement.

Pour simplifier, le courant électrique sort du générateur par la borne positive (+), traverse le circuit électrique et revient au générateur par sa borne négative (-).



fritzing

Il y a deux façons de transporter du **courant électrique** :

- sous la forme de **courant continu** comme les piles
- sous la forme de **courant alternatif** comme le 200V qui est fourni par les prises de courant



L'Arduino doit être alimenté en courant continu 5V, par des piles, une alimentation externe, par l'ordinateur qui transforme le courant alternatif en courant continu.

Chacun des connecteurs de l'Arduino Uno reçoit une tension de 5V et une intensité de 40mA (limité à 200mA pour le total)

Les composants du circuit absorbent une partie du potentiel. Par exemple, la LED consomme environ 2V.

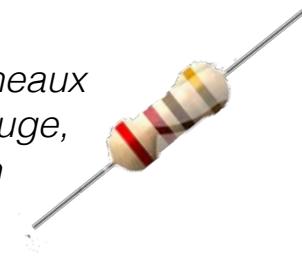
On utilise aussi des **résistances** dont le but est seulement de modifier le potentiel et réduire le courant de retour.

La résistance se mesure en **Ohms** dont le symbole est **Ω** (c'est une lettre grecque). Les anneaux indiquent la valeur de la résistance.

220 Ω 5 anneaux
Rouge, Rouge,
Noir, Noir



220 Ω 4 anneaux
Rouge, Rouge,
Marron



L'**électronique** utilise l'électricité pour traiter, transmettre et recevoir des informations.

Les micro-contrôleurs comme l'Arduino sont des composants matériels qui font des traitements électroniques.

L'électronique traite des **signaux électroniques**.

Un **signal** est une grandeur qui est considérée comme représentant de manière suffisamment satisfaisante une grandeur physique donnée et qui porte l'information à traiter.

Par exemple, un son naturel est un signal analogique.



Un **signal analogique** évolue d'une façon continue dans le temps.

La plupart des grandeurs physiques (par exemple, la température, la luminosité) évoluent d'une façon continue et sont analogiques.

Le signal est transformé en variation du courant électrique. L'objectif du circuit électronique sera de manipuler les caractéristiques du courant pour transformer le signal.

Par exemple, on peut augmenter l'amplitude d'un son et le rendre plus fort.



circuit
→



Le traitement analogique est très rapide, mais il ne permet pas de tout faire.

De plus en plus, l'électronique utilise aussi du logiciel, des programmes. Pour cela on a besoin de données numériques.

On peut transformer un signal analogique en **signal numérique**, c'est à dire en une suite de nombres. On parle de signaux discrétisés ou numérisés pour lesquels on ne prend en compte qu'un nombre fini d'états.



Dans le cas le plus simple, un signal numérique n'a avoir que 2 états possible : 1 et 0.

L'électronique numérique est utilisée en particulier dans les systèmes contenant un microprocesseur ou un micro-contrôleur.

L'Arduino inclus un Convertisseur Analogique Numérique qui gère les 6 **connecteurs analogiques** appelés A0 à A5. Ces ports sont utiles lorsque l'on travaille avec des capteurs de température ou de lumière.



Les autres **connecteurs** de l'Arduino sont **numériques** (ou digital en anglais) et ne gèrent que les états **0** et **1**, le signal est à LOW ou HIGH.



On a vu que l'Arduino ne connaît que 2 états : 0 et 1. Le courant ne passe pas (LOW), ou il passe (HIGH).

Comment fait on pour manipuler des nombres ?

Les nombres vont être représentés sous forme de bits d'information. Le **bit** n'a que deux états **0 ou 1**. On dit aussi qu'ils sont représentés **en binaire** (bi- signifie 2 comme dans bicyclette).

Par exemple le nombre 42 s'écrit 101010 en binaire.

Si on n'a que 2 états il faut compter en **base 2**.

Tu comptes en **base 10** avec 10 chiffres : 0 1 2 3 4 5 6 7 8 9.

Tu as 10 états possibles mais tu peux compter au delà de 9. De 0 à 9 c'est facile, et pour les nombres suivants tu dois changer de rang. On ajoute le rang des dizaines et on recommence pour les unités 10 11 12 13 14 ...

L'ordinateur fait pareil en base 2 avec seulement 2 signes :

- 0 s'écrit 0
- 1 s'écrit 1
- 2 s'écrit 10 (on doit changer de rang car 2 n'existe pas)
- 3 s'écrit 11
- 4 s'écrit 100
- 5 s'écrit 101

C'est nécessaire pour le micro-contrôleur mais pas pratique pour nous. Il existe des recettes pour convertir la base 2 en base 10, et le contraire. En informatique on appelle les recettes des **algorithmes**.

Si tu lis le nombre 9534

Tu sais que c'est $9 \times 1000 + 5 \times 100 + 3 \times 10 + 4$

L'unité est 10 fois plus grande à chaque rang.

Pour **convertir la base 2 en base 10**, on fait un peu pareil mais avec 2, puis 4 (2×2), puis 8 (2×4) :

$$\begin{array}{r} 1010 \\ \swarrow \quad \searrow \\ 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 0 \\ \hline 10 \end{array}$$

Pour **convertir la base 10 en base 2**, on va diviser le nombre en base 10 par 2, garder le reste et rediviser le quotient jusqu'à ce que la division donne 0.

Par exemple pour 6 : $6/2$

$$\begin{array}{r} q=3 \ r=0 \\ \swarrow \quad \searrow \\ 3/2 \quad q=1 \ r=1 \\ \swarrow \quad \searrow \\ 1/2 \quad q=0 \ r=1 \\ \swarrow \quad \searrow \\ 0 \quad 110 \end{array}$$

6 s'écrit 110 en binaire

Et 9534 s'écrit 10010100111110 en binaire !

Pour simplifier l'écriture, les informaticiens regroupent les bits.

- Par **paquet de 4**, cela s'appelle un **nombre hexadécimal** car il y a 16 états sur 4 bits ($2 \times 2 \times 2 \times 2$). Les signes sont désignés par 0 1 2 3 4 5 6 7 8 9 A B C D E F.

Par exemple 9534 s'écrit 253E en hexadécimal ce qui est un plus simple à retenir.

- Par **paquet de 8**, cela s'appelle un **octet (byte en anglais)**. Il est représenté soit par 2 hexadécimaux (par exemple FF pour 255) ou par un nombre décimal équivalent.

Nous allons un peu transformer le programme Blink pur qu'il soit plus facile à utiliser pour nous. Nous allons écrire nos propres fonctions.

Une **fonction** en informatique c'est une suite d'instructions qui réalise un calcul ou une tâche.

Les fonctions permettent de regrouper des instructions que l'on utilise souvent et de les réutiliser plus facilement. Elles permettent aussi de découper un problème compliqué en éléments plus simples.

Nous avons déjà utilisé des fonctions :

- `pinMode(13, OUTPUT)`
- `digitalWrite(13, HIGH)`
- `delay(1000)` attend 1s (1000 ms)

La définition d'une fonction comporte des **paramètres d'entrée**, par exemple 13 ou HIGH. La fonction va les utiliser pour réaliser sa tâche.

Elle peut retourner une **valeur de sortie**.

Par exemple si j'écris

```
int produit = multiplie(2, 3)
```

produit aura la valeur 6 après l'appel de la fonction.

Le mot qui est devant `produit` s'appelle le **type** et `produit` s'appelle une **variable**. On ne connaît pas ça valeur. Ce sera le résultat de l'exécution de la fonction.

Il existe plusieurs types : `int` indique que c'est un nombre entier. Il existe d'autres type comme `float` pour les nombre décimaux, ou `char` pour les caractères.

Les types sont aussi nécessaires pour la définition de la fonction.

Par exemple, la fonction qui multiplie deux nombres entiers s'écrirait comme ça :

```
int multiplie(int a, int b) {  
    return a * b;  
}
```

Tu peux maintenant écrire des fonctions plus simples à utiliser comme `allumeLED()`, `eteintLED` et `attend` pour éviter de calculer les milli-secondes.

```
// the loop function runs over and over again forever  
void loop() {  
    allumeLED();  
    attend(1);           // wait for a second  
    eteintLED();  
    attend(1);           // wait for a second  
}  
  
void allumeLED() {  
    digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)  
}  
  
void eteintLED() {  
    digitalWrite(13, LOW); // turn the LED off by making the voltage LOW  
}  
  
void attend(int temps) {  
    delay(temps * 1000); // attends le nombre de secondes demandées  
}
```