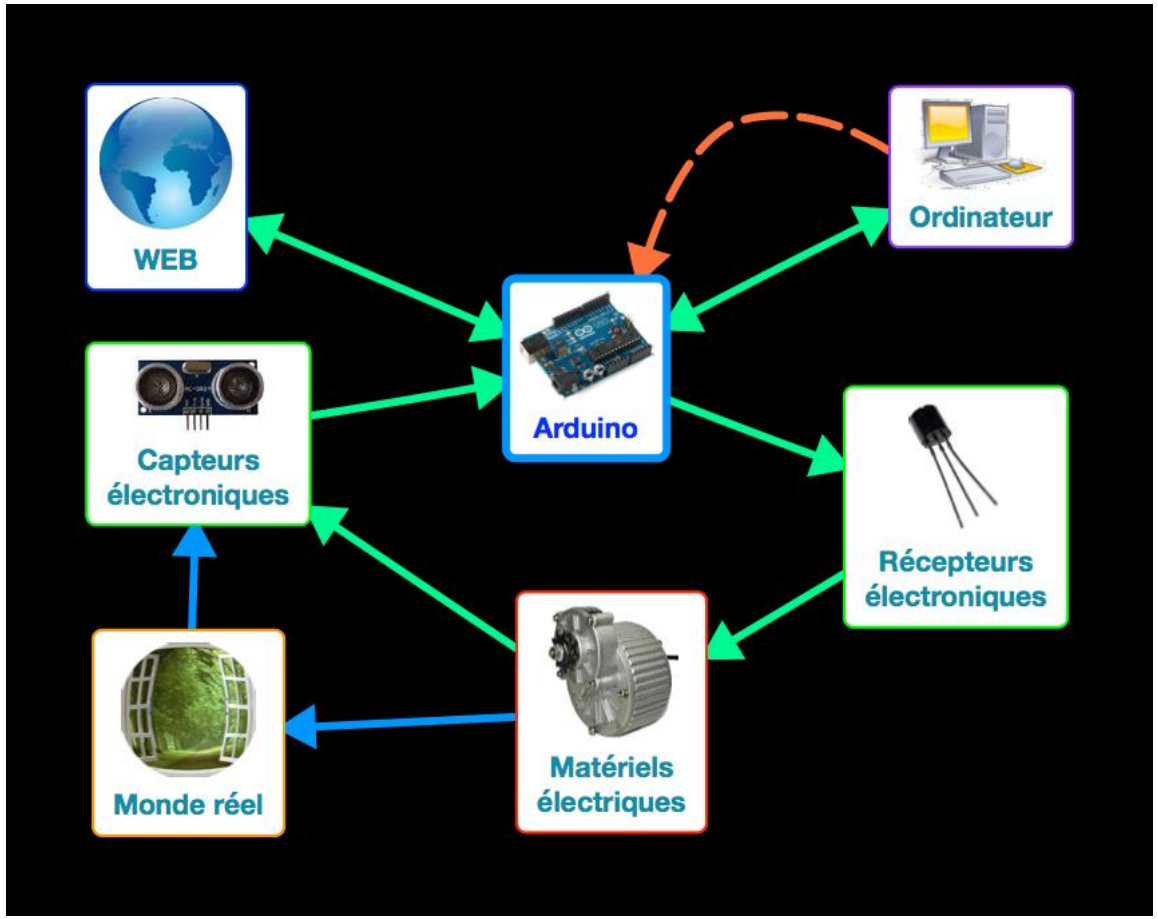
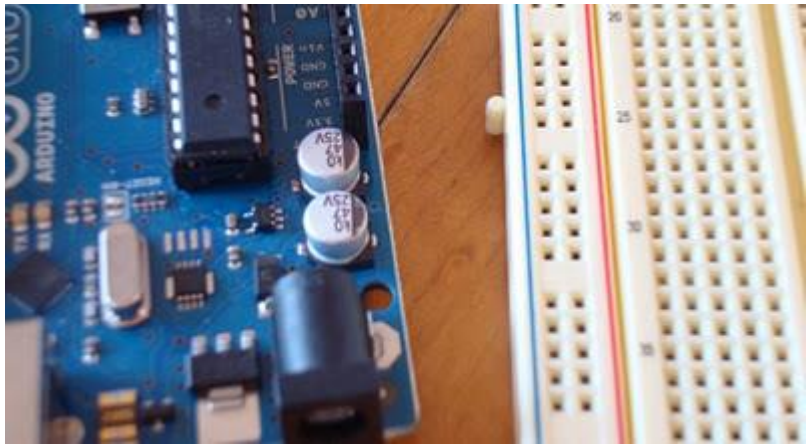
TP1-Arduino


WEB
Ordinateur
Arduino
Capteurs électroniques
Récepteurs électroniques
Monde réel
Matériels électriques

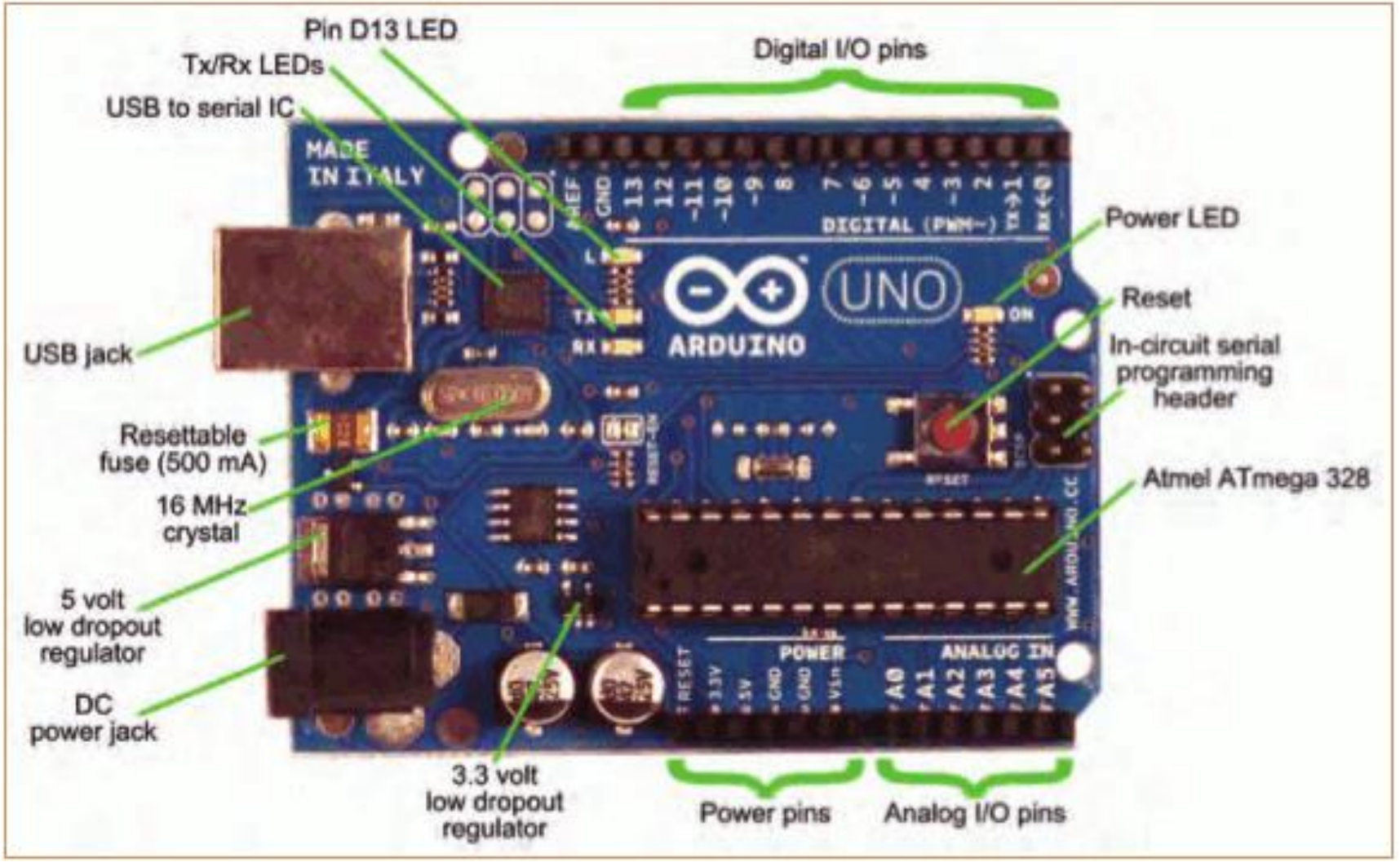
What is a Development Board



- A printed circuit board designed to facilitate work with a particular microcontroller.

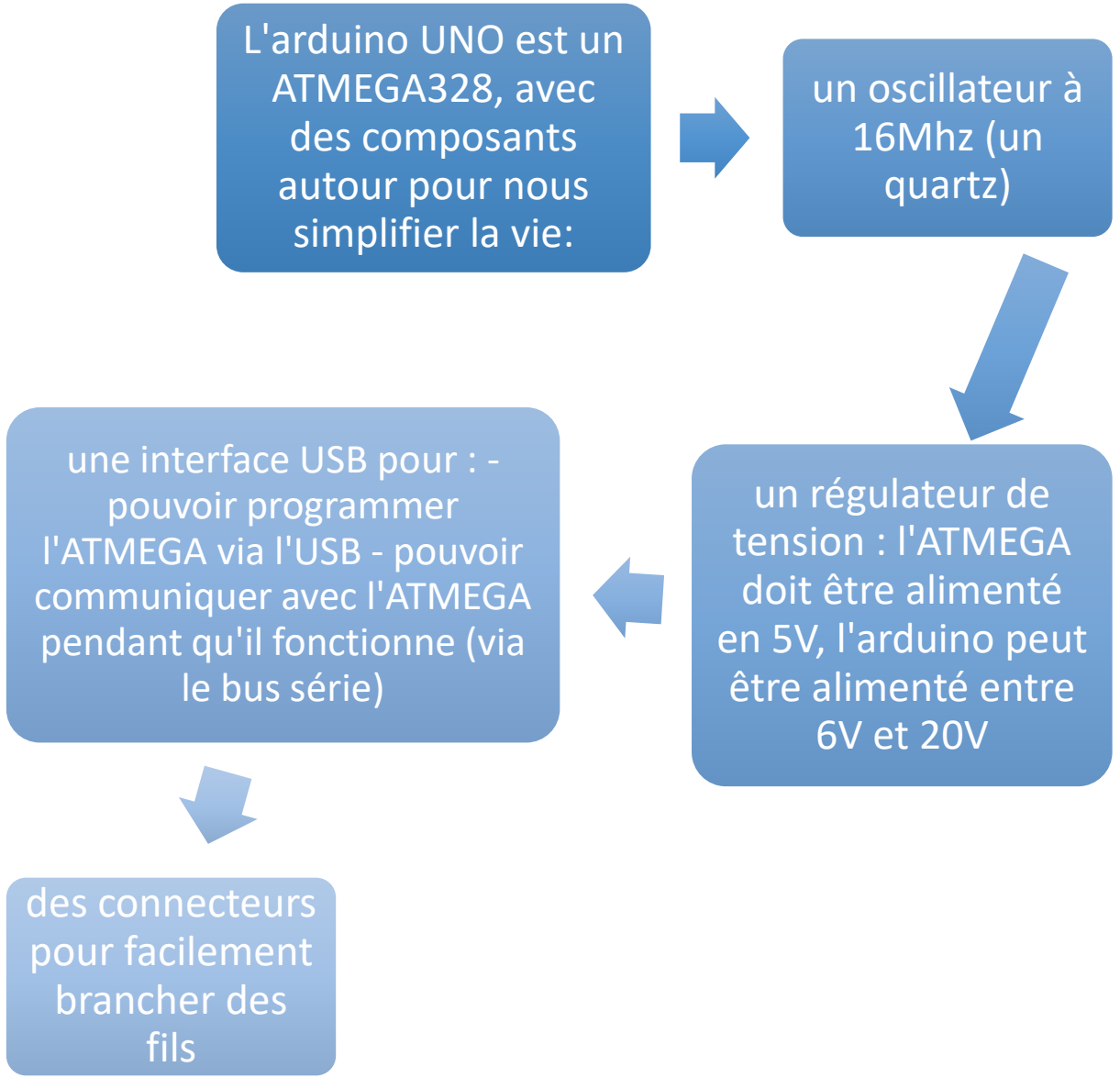
- Typical components include:
  - power circuit
  - programming interface
  - basic input; usually buttons and LEDs
  - I/O pins

The Arduino Development Board



Making-robots-with-arduino.pdf

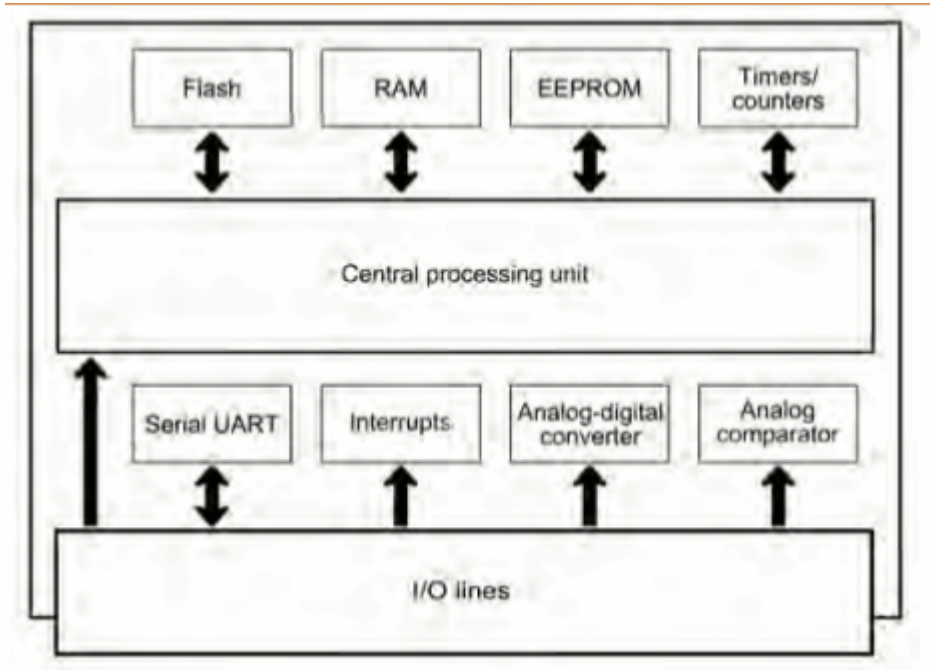
Arduino


L'arduino UNO est un ATMEGA328, avec des composants autour pour nous simplifier la vie:
un oscillateur à 16Mhz (un quartz)
un régulateur de tension : l'ATMEGA doit être alimenté en 5V, l'arduino peut être alimenté entre 6V et 20V
une interface USB pour : - pouvoir programmer l'ATMEGA via l'USB - pouvoir communiquer avec l'ATMEGA pendant qu'il fonctionne (via le bus série)
des connecteurs pour facilement brancher des fils

ATMEGA328-ATMEL

Ses caractéristiques: 8 bits	20 Mhz = 20 million d'instructions à la seconde	32Ko de mémoire Flash, pour le programme
2Ko de RAM, pour les données	1Ko de EEPROM, pour les données qui doivent être conservées	Ses entrées-sorties: 23 entrées/sorties (binaires) programmables
6 entrés analogiques (ADC)	6 sorties PWM (Pulse Width Modulation)	Bus: SPI, I2C, série... Parenthèse: les mémoire

The Arduino Microcontroller: Atmel ARV Atmega 328



Specification

Making-robots-with-arduino.pdf

What is the Arduino

The word "Arduino" can mean 3 things

A physical piece of hardware

A programming environment

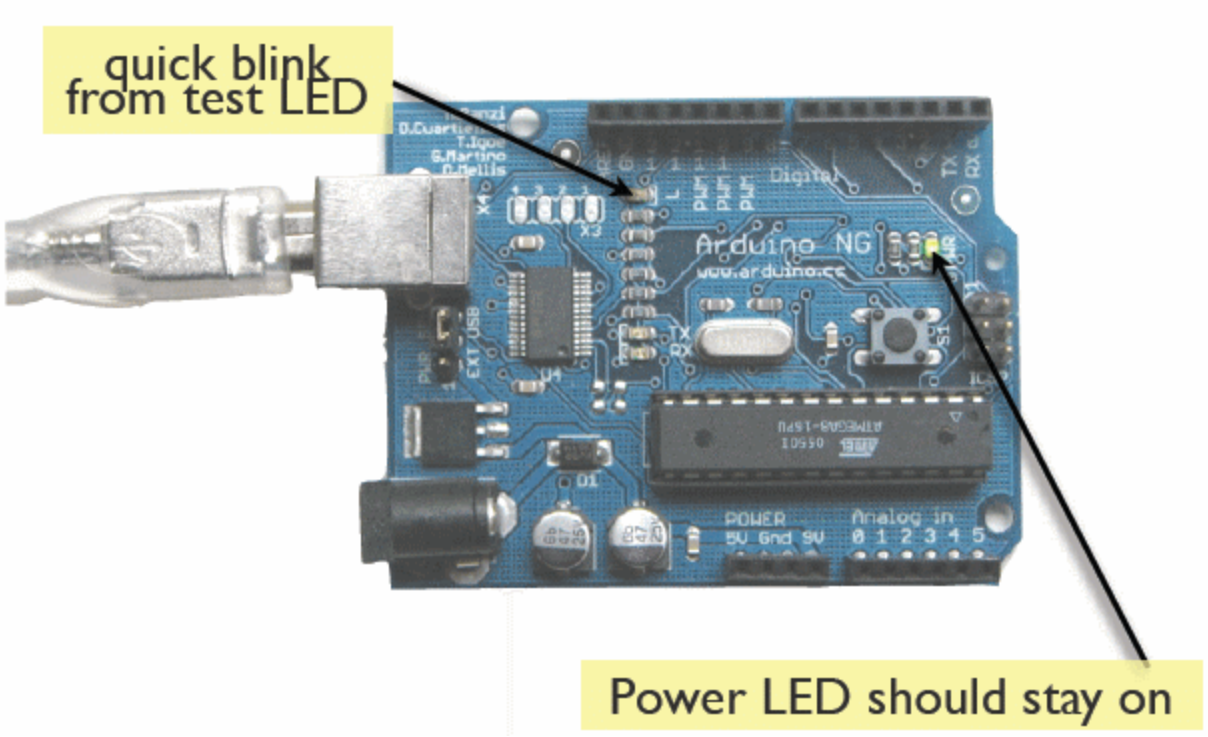
A community & philosophy

todbot.com/blog/bionicarduino

Getting Started

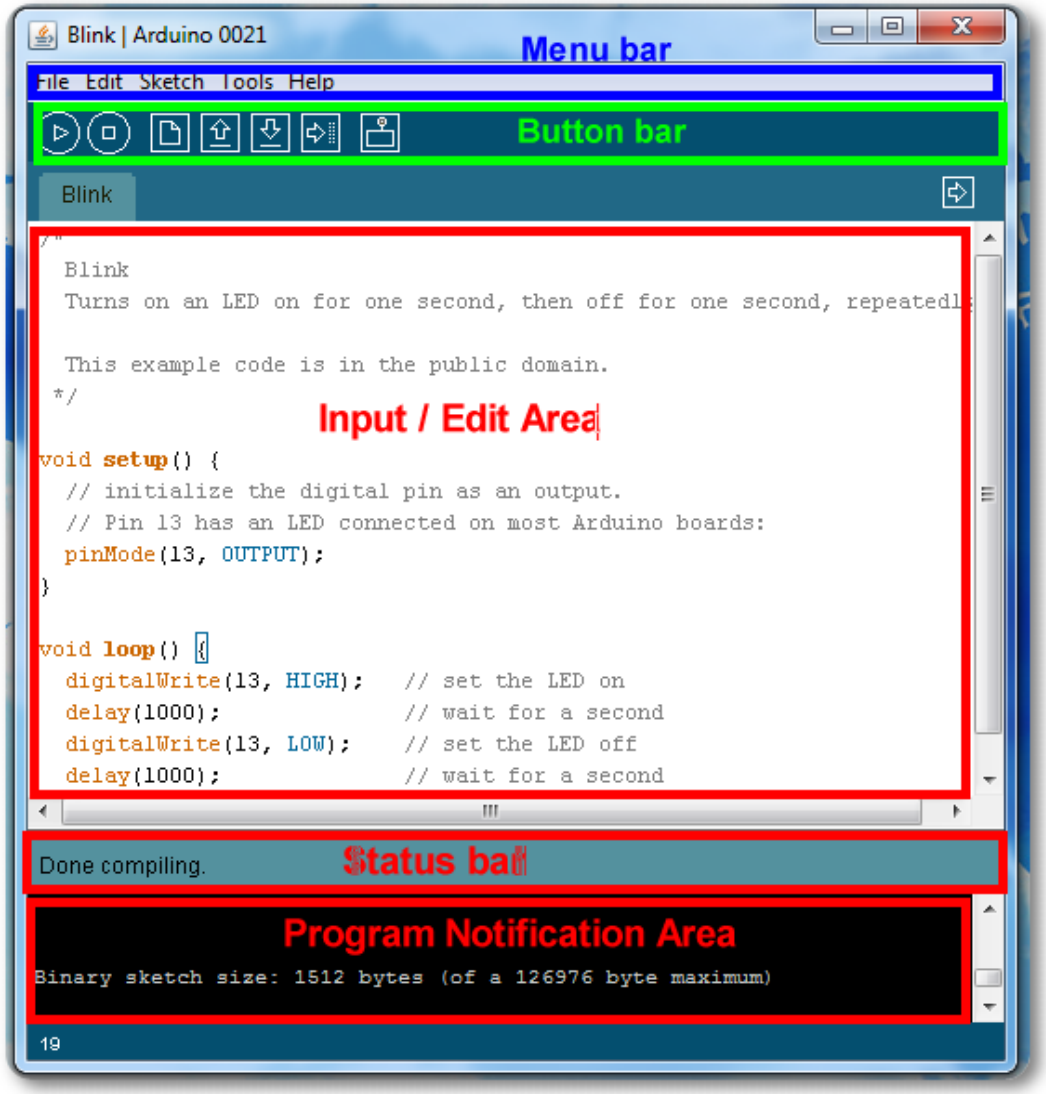
- Check out: http://arduino.cc/en/Guide/HomePage
  1. **Download & install the Arduino environment (IDE)**
  2. **Connect the board to your computer via the UBS cable**
  3. **If needed, install the drivers (not needed in lab)**
  4. **Launch the Arduino IDE**
  5. **Select your board**
  6. **Select your serial port**
  7. **Open the blink example**
  8. **Upload the program**

Try It: Connect the USB Cable



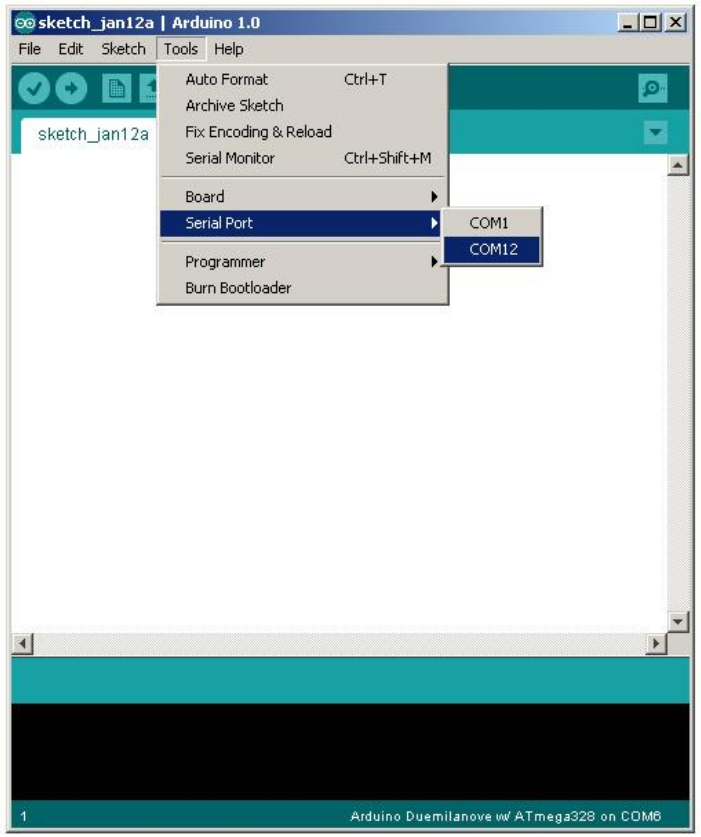
todbot.com/blog/bionicarduino

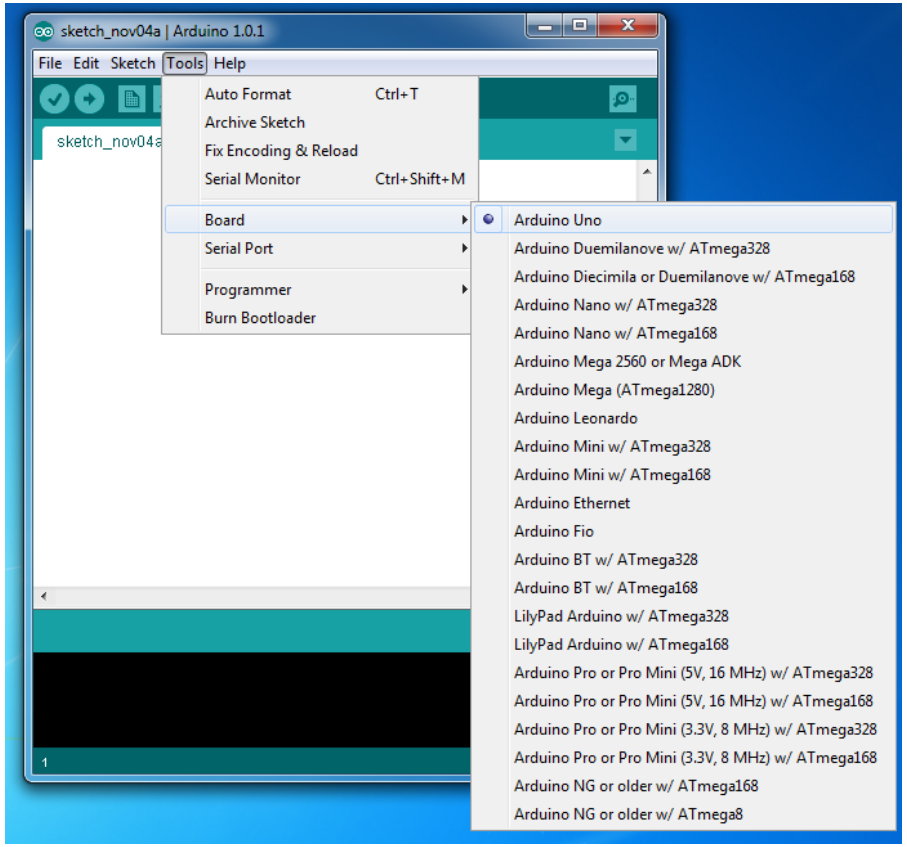
Arduino IDE



See: http://arduino.cc/en/Guide/Environment for more information

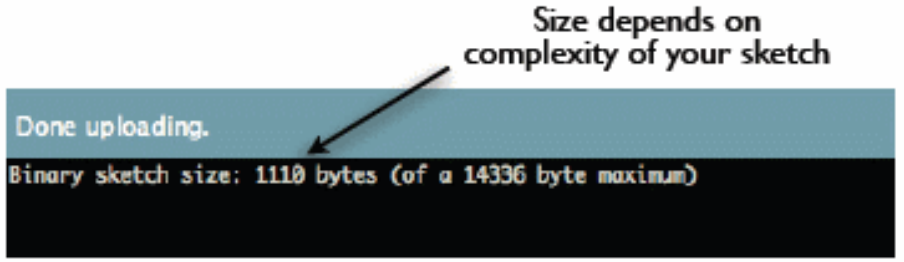
Select Serial Port and Board



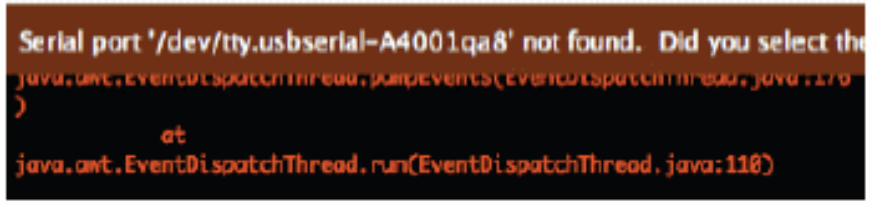


Status Messages

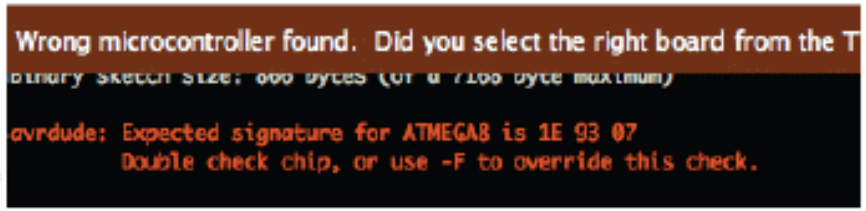
Uploading worked



Wrong serial port selected



Wrong board selected



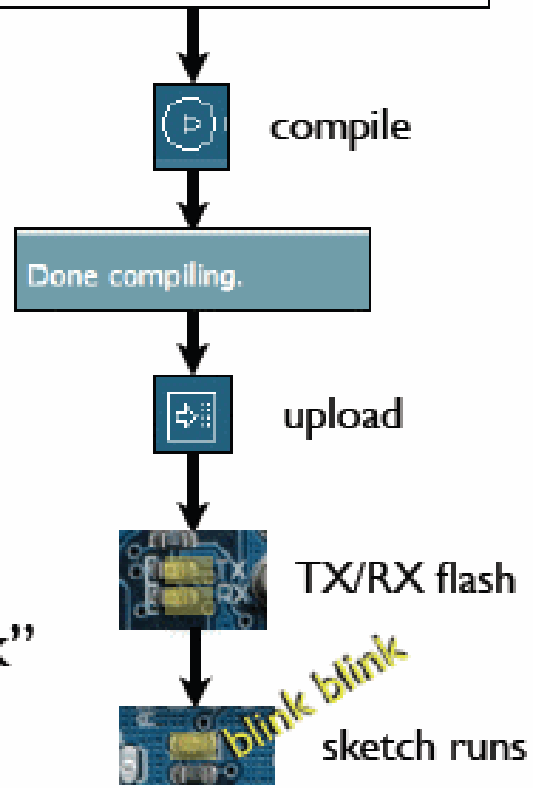
nerdy cryptic error messages

todbot.com/blog/bionicarduino

Using Arduino

- Write your sketch
- Press Compile button
  (to check for errors)
- Press Upload button to program
  Arduino board with your sketch

```
void setup() {
  pinMode(ledPin, OUTPUT);      // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```



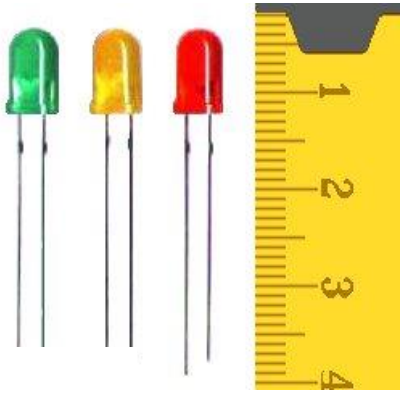
Try it out with the "Blink" sketch!

Load "File/Sketchbook/Examples/Digital/Blink"

La diode électroluminescente

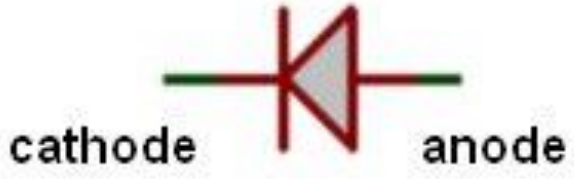
DEL / LED ?

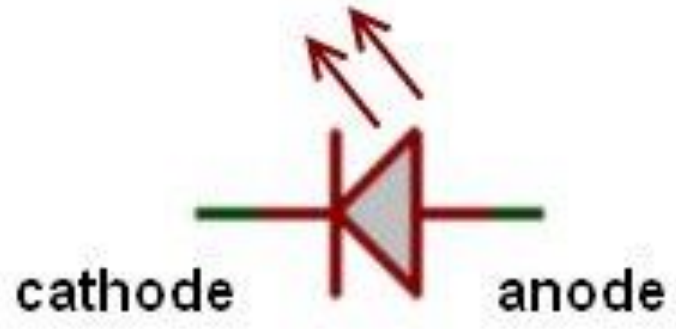
Une **DEL** / **LED** : **D**iode **E**lectro-**L**uminescente, ou bien "**L**ight **E**mitting **D**iode" en anglais.
Il s'agit d'un composant électronique qui crée de la lumière quand il est parcouru par un courant électrique.



Une LED est en fait une **diode** qui émet de la lumière. Je vais donc vous parler du fonctionnement des diodes en même temps que celui des LED.

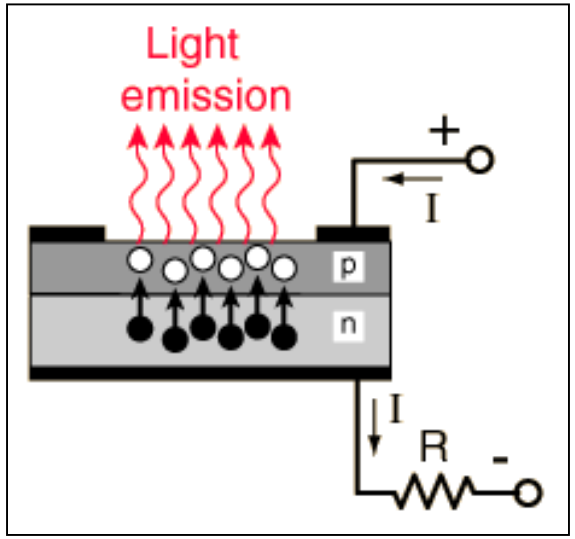
Symbole de la diode
Celui de la LED est :

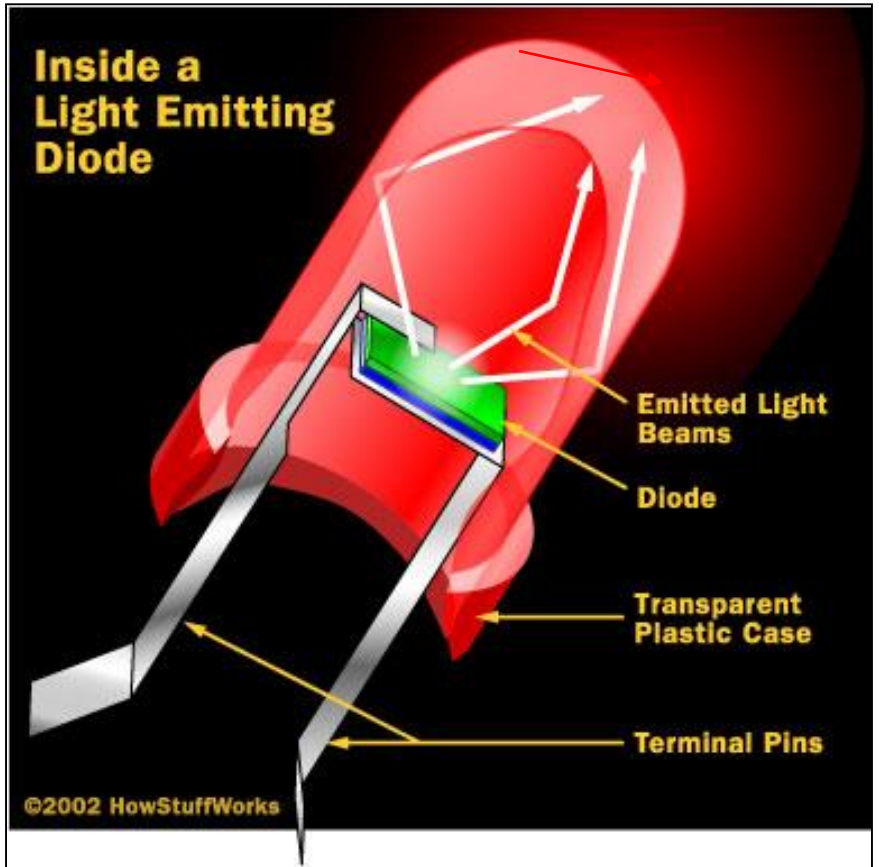




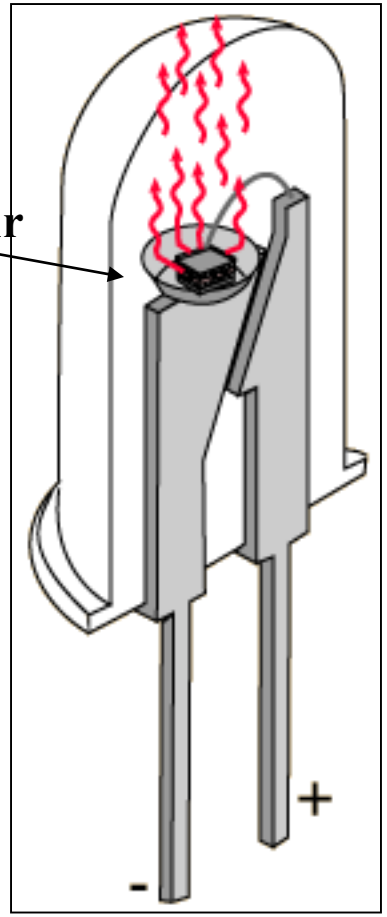
Symbole de la LED
Il y a donc très peu de différence entre les deux. La LED est simplement une diode qui émet de la lumière, d'où les flèches sur son symbole.

Diodes électroluminescentes

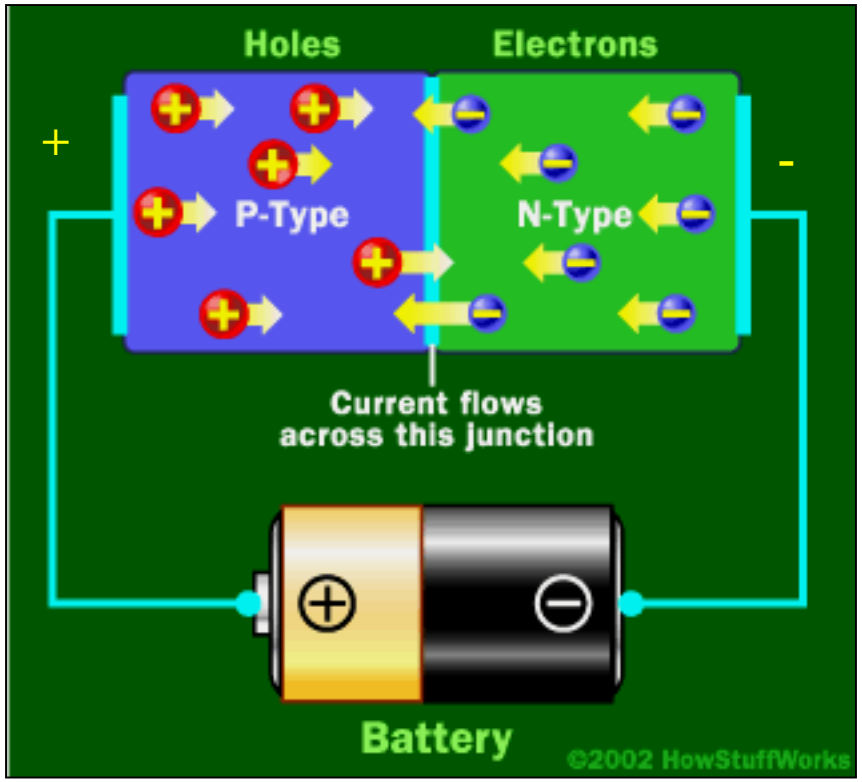


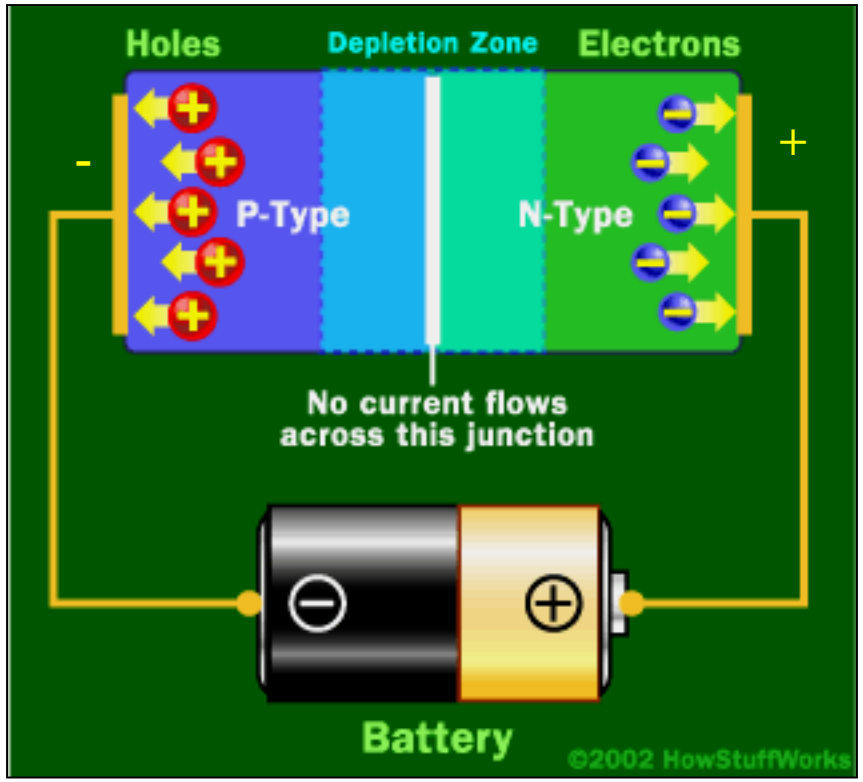


réflecteur



Diode Zener

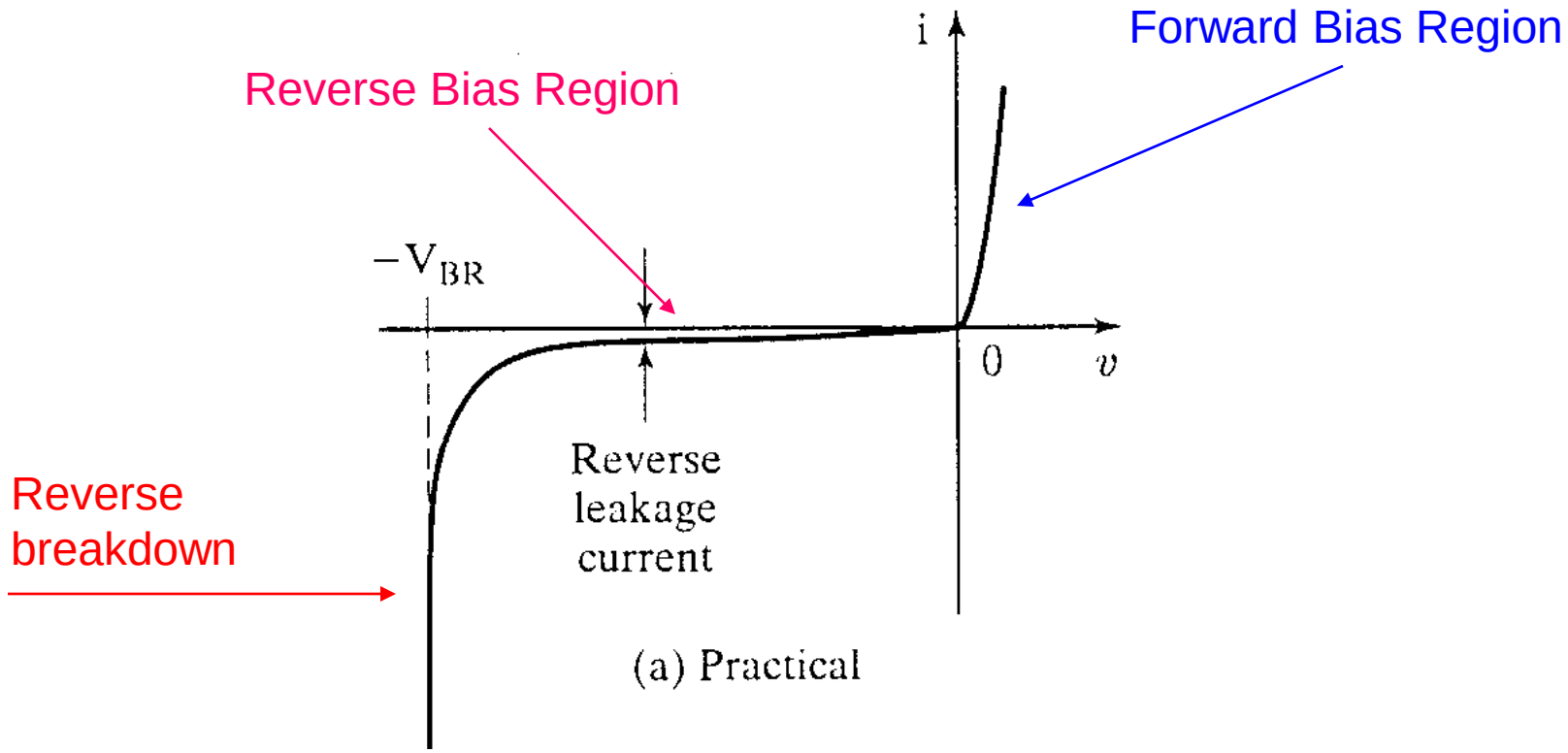




Polarisation directe **Polarisation inverse**

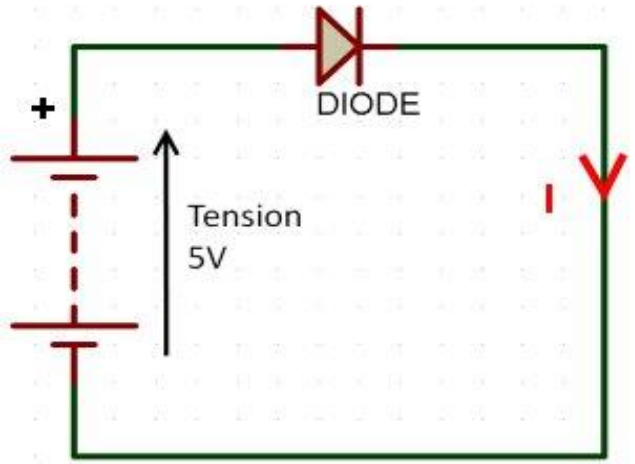
Redresseur de courant alternatif

Graphical
PN-Junction Diode V-I Characteristic

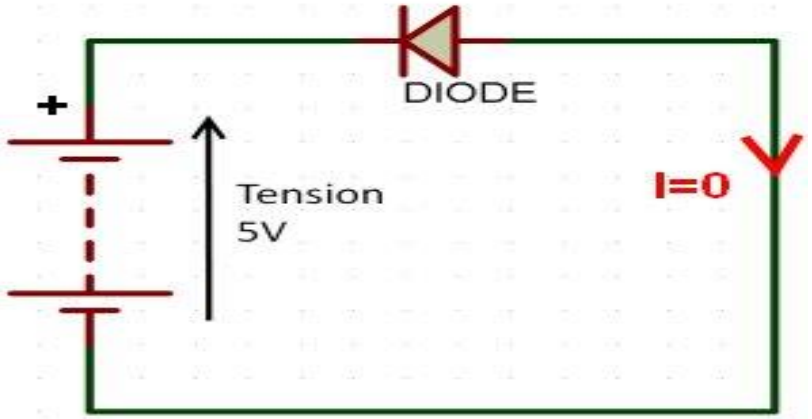


Fonctionnement

Polarisation directe
Pour polariser la diode, on doit faire en sorte que le courant la parcours de l'anode vers la cathode. Autrement dit, la tension doit être plus élevée à l'anode qu'à la cathode.



Polarisation inverse
La polarisation inverse d'une diode est l'opposé de la polarisation directe. Pour créer ce type de montage, il suffit simplement, dans notre cas, de "retourner" la diode enfin la brancher "à l'envers". Dans ce cas, le courant ne passe pas.



Diode polarisée en inverse
Note : une diode polarisée en inverse ne grillera pas si elle est utilisée dans de bonnes conditions. En fait, elle fonctionne de "la même façon" pour le courant positif et négatif.

le Courant et la Tension

La tension maximum directe

Lorsque l’on utilise un composant, on doit prendre l’habitude d’utiliser la *"datasheet"* qui nous donne toutes les caractéristiques sur le composant. Dans cette datasheet, on retrouvera quelque chose appelé "Forward Voltage", pour la diode. Cette indication représente la chute de tension aux bornes de la diode lorsque du courant la traverse en sens direct. Pour une diode classique (type 1N4148), cette tension sera d’environ 1V. Pour une LED, on considérera plutôt une tension de 1,2 à 1,6V.

La tension maximum inverse

Cette tension représente la différence maximum admissible entre l’anode et la cathode lorsque celle-ci est branchée "à l’envers". En effet, si vous mettez une tension trop importante à ces bornes, la jonction ne pourra pas le supporter et partira en fumée. En anglais, on retrouve cette tension sous le nom de "Reverse Voltage" (ou même "Breakdown Voltage"). Si l’on reprend la diode 1N4148, elle sera comprise entre 75 et 100V. Au-delà de cette tension, la jonction casse et la diode devient inutilisable. Dans ce cas, la diode devient soit un court-circuit, soit un circuit ouvert.

Le courant de passage

Le courant qui traverse une LED a son importance. Si l'on branche directement la LED sur une pile, elle va s'allumer, puis tôt ou tard finira par s'éteindre… définitivement. En effet, si on ne limite pas le courant traversant la LED, elle prendra le courant maximum, et ça c'est pas bon car ce n'est pas le courant maximum qu'elle peut supporter. Pour limiter le courant, on place une résistance avant (ou après) la LED. Cette résistance, savamment calculée, lui permettra d'assurer un fonctionnement optimal.

Simplement avec la formule de base, la loi d'Ohm.

$$U=R*I$$

Dans le cas d'une LED, on considère, en général, que l'intensité la traversant doit être de 20 mA. Si on veut être rigoureux, il faut aller chercher cette valeur dans le datasheet. On a donc $I=20mAI=20mA$. Ensuite, on prendra pour l'exemple une tension d'alimentation de 5V (en sortie de l'Arduino, par exemple) et une tension aux bornes de la LED de 1,2V en fonctionnement normal. On peut donc calculer la tension qui sera aux bornes de la résistance : $Ur=5−1,2=3,8VUr=5−1,2=3,8V$. Enfin, on peut calculer la valeur de la résistance à utiliser :

$$R=UIR=3,80,02\ R=190ΩR=UIR=3,80,02\ R=190Ω$$

À votre avis, vaut-il mieux utiliser une résistance de plus forte valeur ou de plus faible valeur ?

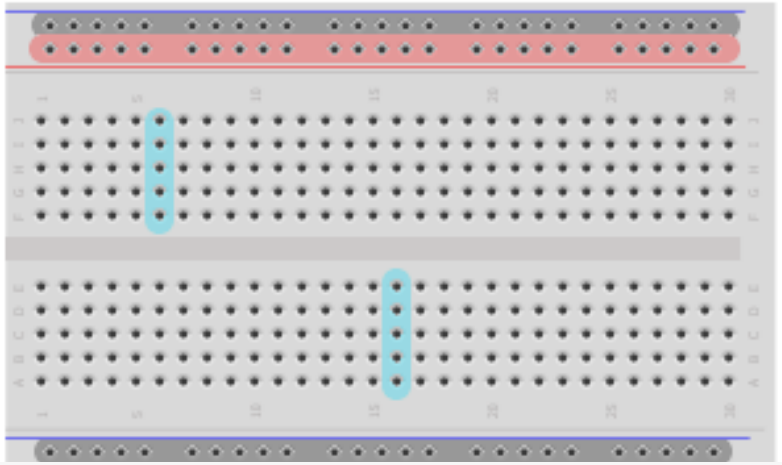
- **Objectif**
- L’objectif de ce premier programme va consister à allumer une LED.

- **Matériel**
- Pour pouvoir programmer, il vous faut, bien évidemment, une carte Arduino et un câble USB pour relier la carte au PC. Mais pour voir le résultat de votre programme, vous aurez besoin d’éléments supplémentaires. Notamment, une LED et une résistance.

la breadboard !

Principe de la breadboard

Certes la plaque est pleine de trous, mais pas de manière innocente !



Une breadboard

- Les zones rouges et noires correspondent à l'alimentation. Souvent, on retrouve deux lignes comme celles-ci permettant de relier ses composants aux alimentations nécessaires.
- Par convention, le noir représente la masse et le rouge est l'alimentation (+5V, +12V, -5V… ce que vous voulez y amener).
- Habituellement tous les trous d'une même **ligne** sont reliés sur cette zone. Ainsi, vous avez une ligne d'alimentation parcourant tout le long de la carte.

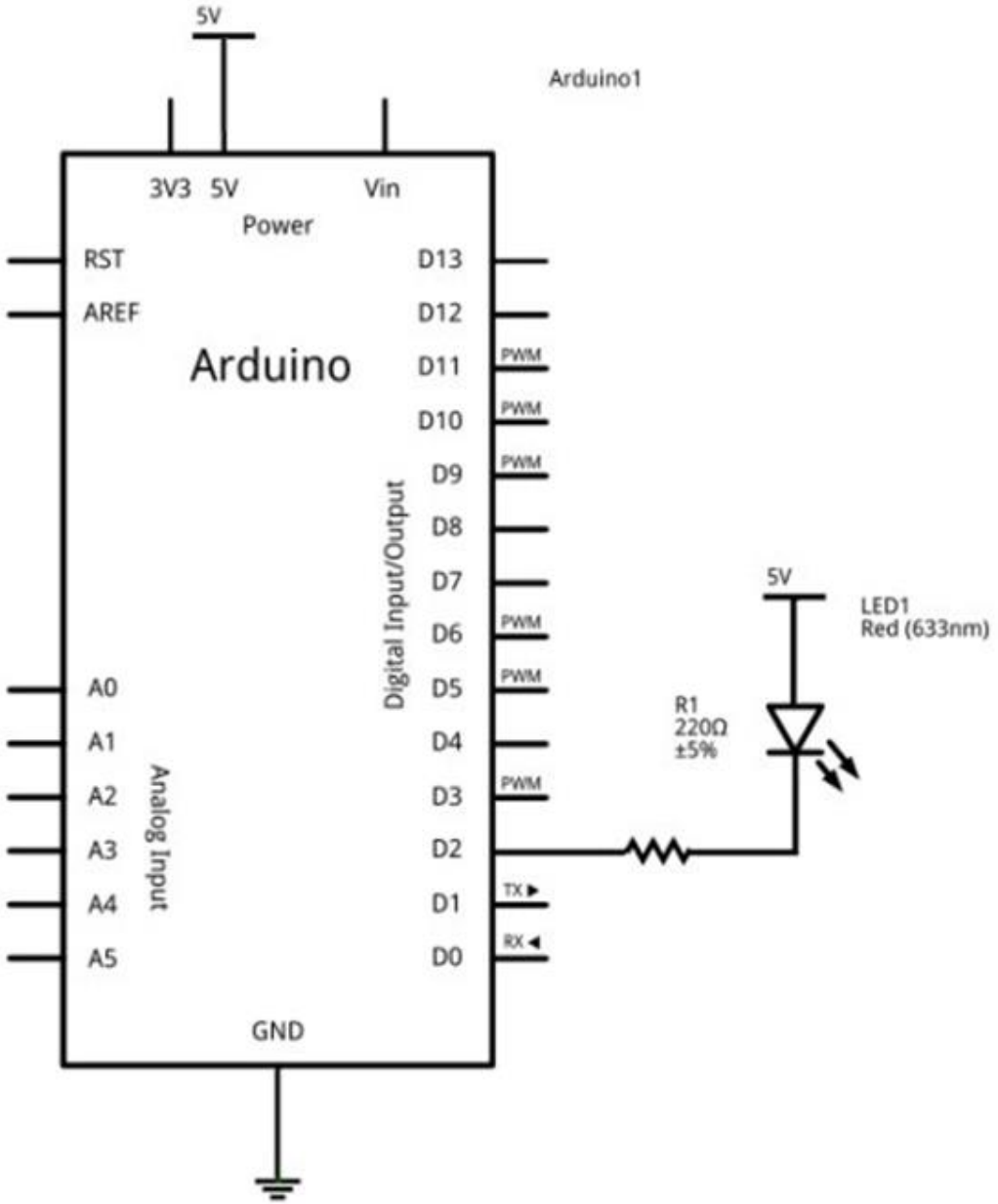
Ensuite, on peut voir des zones en bleu.

- Ces zones sont reliées entre elles par **colonne**. Ainsi, tous les trous sur une même colonne sont reliés entre eux.

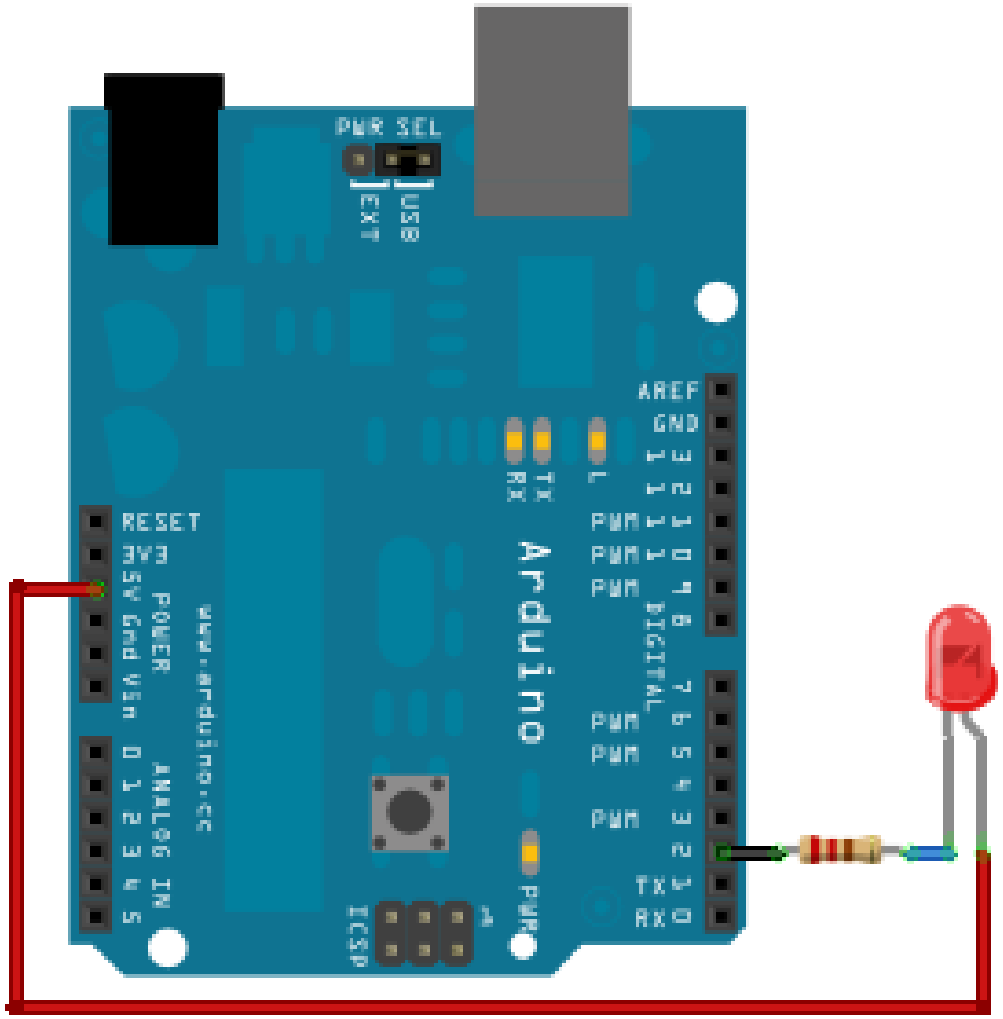
- En revanche, chaque colonne est distincte. En faisant chevaucher des composants sur plusieurs colonnes vous pouvez les connecter entre eux.
- Dernier point, vous pouvez remarquer un espace coupant la carte en deux de manière symétrique. Cet espace coupe aussi la liaison des colonnes. Ainsi, sur le dessin ci-dessus on peut voir que chaque colonne possède cinq trous reliés entre eux. Cet espace au milieu est normalisé et doit faire la largeur des circuits intégrés standards. En posant un circuit intégré à cheval au milieu, chaque patte de ce dernier se retrouve donc sur une colonne, isolée de la précédente et de la suivante.

Réalisation

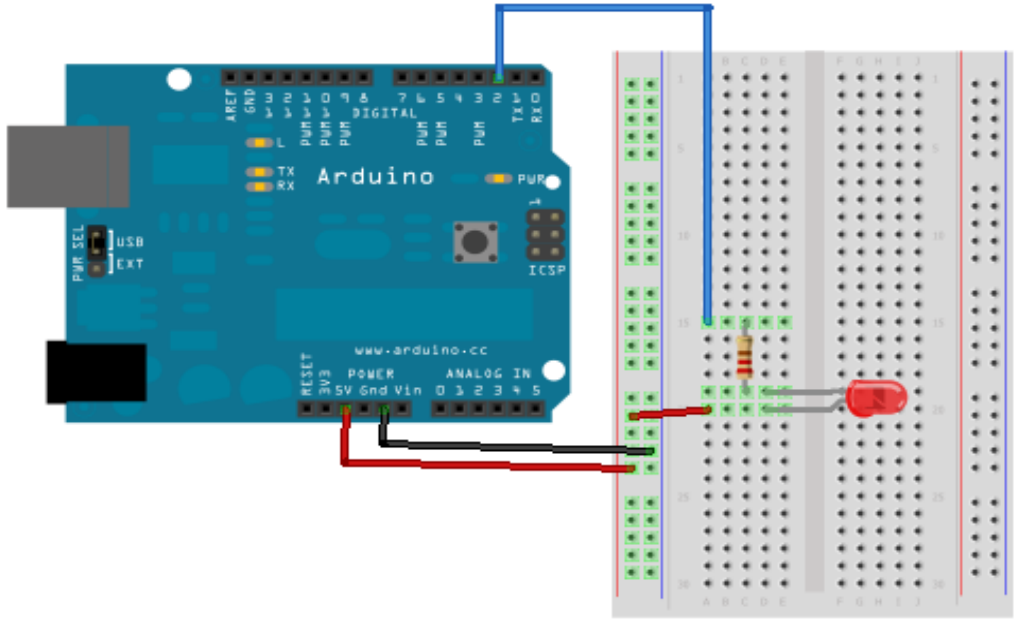
- Avec le brochage de la carte Arduino, vous devrez connecter la plus grande patte au +5V (broche *5V*). La plus petite patte étant reliée à la résistance, elle-même reliée à la broche numéro 2 de la carte.
- En effet, on pourrait faire le contraire, brancher la LED vers la masse et l'allumer en fournissant le 5V depuis la broche de signal.
- Cependant, les composants comme les microcontrôleurs n'aiment pas trop délivrer du courant, ils préfèrent l'absorber. Pour cela, on préférera donc alimenter la LED en la placant au +5V et en mettant la broche de Arduino à la masse pour faire passer le courant. Si on met la broche à 5V, dans ce cas le potentiel est le même de chaque côté de la LED et elle ne s'allume pas !

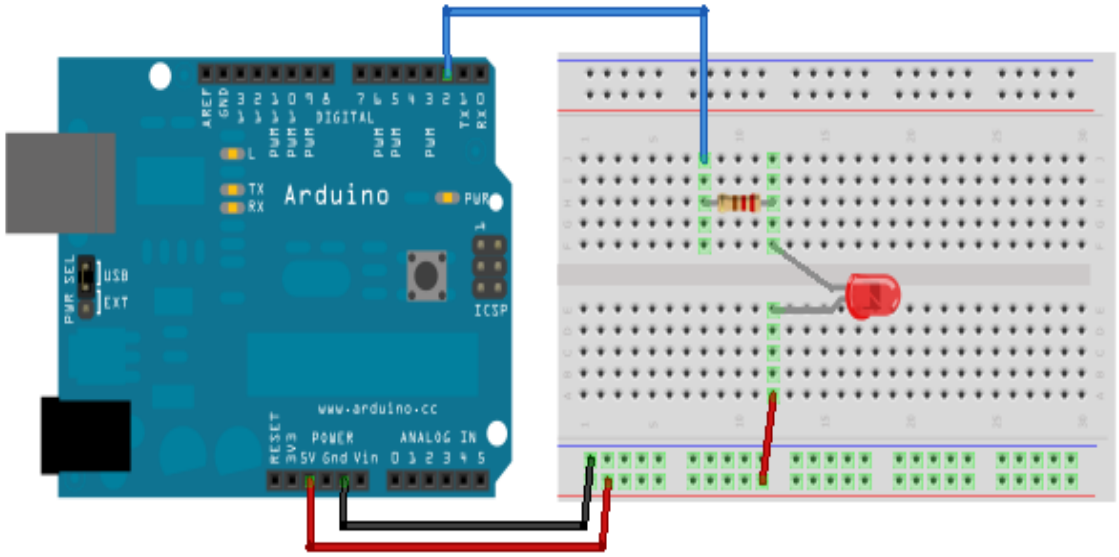


Réalisation



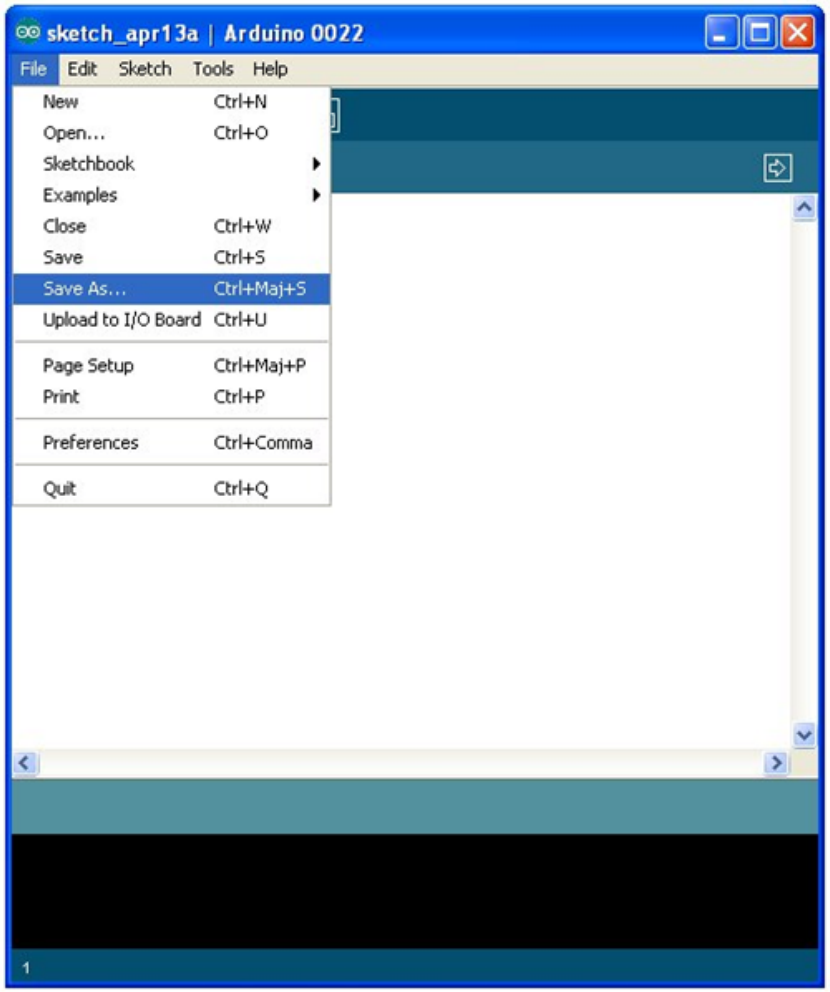
Réalisation





Créer un nouveau projet

- Pour pouvoir programmer notre carte, il faut que l'on créer un nouveau programme. Ouvrez votre logiciel Arduino. Allez dans le menu *File* et choisissez l'option *Save as…* :



Le code minimal

Pour commencer le programme, il nous faut un code minimal. Ce code va nous permettre d'initialiser la carte et va servir à écrire notre propre programme. Ce code, le voici :
Squelette minimal d'un programme Arduino

```
1  // fonction d'initialisation de la carte
2  void setup()
3  {
4    // contenu de l'initialisation
5  }
6
7  // fonction principale, elle se répète (s'exécute) à l'infini
8  void loop()
9  {
10   // contenu de votre programme
11 }
```

La référence Arduino

- **Qu’est ce que c’est ?**
- L’Arduino étant un projet dont la communauté est très active, nous offre sur son site internet une **référence**. Mais qu’est ce que c’est ? Eh bien il s’agit simplement de "la notice d’utilisation" du langage Arduino. Plus exactement, une page internet de leur site est dédiée au référencement de chaque code que l’on peut utiliser pour faire un programme.
- **Comment l’utiliser ?**
  - *Structure* : cette colonne référence les éléments de la structure du langage Arduino. On y retrouve les conditions, les opérations, etc.
  - *Variables* : comme son nom l’indique, elle regroupe les différents types de variables utilisables, ainsi que certaines opérations particulières
  - *Functions* : ici c’est tout le reste, mais surtout les fonctions de lecture/écriture des broches du microcontrôleur (ainsi que d’autres fonctions bien utiles)
  -

Allumer notre LED

1ère étape

Il faut avant tout définir les broches du micro-contrôleur. Cette étape constitue elle-même deux sous étapes. La première étant de créer une variable définissant la broche utilisée, ensuite, définir si la broche utilisée doit être une entrée du micro-contrôleur ou une sortie.

1. Premièrement, donc, définissons la broche utilisée du microcontrôleur :
   Le terme `const` signifie que l'on définit la variable comme étant constante. Par conséquent, on change la nature de la variable qui devient alors constante et sa valeur ne pourra jamais être changée. Le terme `int` correspond à un type de variable. En définissant une variable de ce type, elle peut stocker un nombre allant de -2147483648 à +2147483647 ! Cela nous suffit amplement !

Nous sommes donc en présence d'une variable, nommée `led_rouge`, qui est en fait une constante, qui peut prendre une valeur allant de -2147483648 à +2147483647. Dans notre cas, cette variable, constante, est assignée à 2.

Lorsque votre code sera compilé, le micro-contrôleur saura ainsi que sur sa broche numéro 2, il y a un élément connecté.

```
1 const int led_rouge = 2;   // définition de la broche 2 de la carte en tant que variable
```

Il faut maintenant dire si cette broche est une **entrée** ou une **sortie**.
il suffit simplement d'interchanger UNE ligne de code pour dire qu'il faut utiliser une broche en entrée (récupération de données) ou en sortie (envoi de données).
Elle doit se trouver dans la fonction `setup()`.
Dans la référence, ce dont nous avons besoin se trouve dans la catégorie **Functions**, puis dans **Digital I/O**. Entrée/Sortie.

La fonction: `pinMode()`.

Pour utiliser cette fonction, il faut lui envoyer deux paramètres :
- Le nom de la variable que l'on a défini à la broche
- Le type de broche que cela va être (entrée ou sortie)

```
1 // fonction d'initialisation de la carte
2 void setup()
3 {
4   // initialisation de la broche 2 comme étant une sortie
5   pinMode(led_rouge, OUTPUT);
6 }
```

Initialisation de la sortie:

Ce code va donc définir la led_rouge (qui est la broche numéro 2 du micro-contrôleur) en sortie, car `OUTPUT`.

```
1  // définition de la broche 2 de la carte en tant que variable
2  const int led_rouge = 2;
3
4  // fonction d'initialisation de la carte
5  void setup()
6  {
7    // initialisation de la broche 2 comme étant une sortie
8    pinMode(led_rouge, OUTPUT);
9  }
10
11 // fonction principale, elle se répète (s'exécute) à l'infini
12 void loop()
13 {
14   // contenu de votre programme
15 }
```

2e étape

On cherche une fonction qui va nous permettre d'allumer cette LED.

`digitalWrite()`.

C'est donc l'écriture d'un état logique (0 ou 1). Quelle se trouve être la première phrase dans la description de cette fonction ? Celle-ci : "Write a HIGH or a LOW value to a digital pin".

En électronique numérique, un niveau haut correspondra à une tension de +5V et un niveau dit bas sera une tension de 0V (généralement la masse). Sauf qu'on a connecté la LED au pôle positif de l'alimentation, donc pour qu'elle s'allume, il faut qu'elle soit reliée au 0V.

Par conséquent, on doit mettre un état bas sur la broche du microcontrôleur. Ainsi, la différence de potentiel aux bornes de la LED permettra à celle-ci de s'allumer.

`digitalWrite()` : Elle requiert deux paramètres. Le nom de la broche que l'on veut mettre à un état logique et la valeur de cet état logique. Nous allons donc écrire le code qui suit, d'après cette syntaxe :

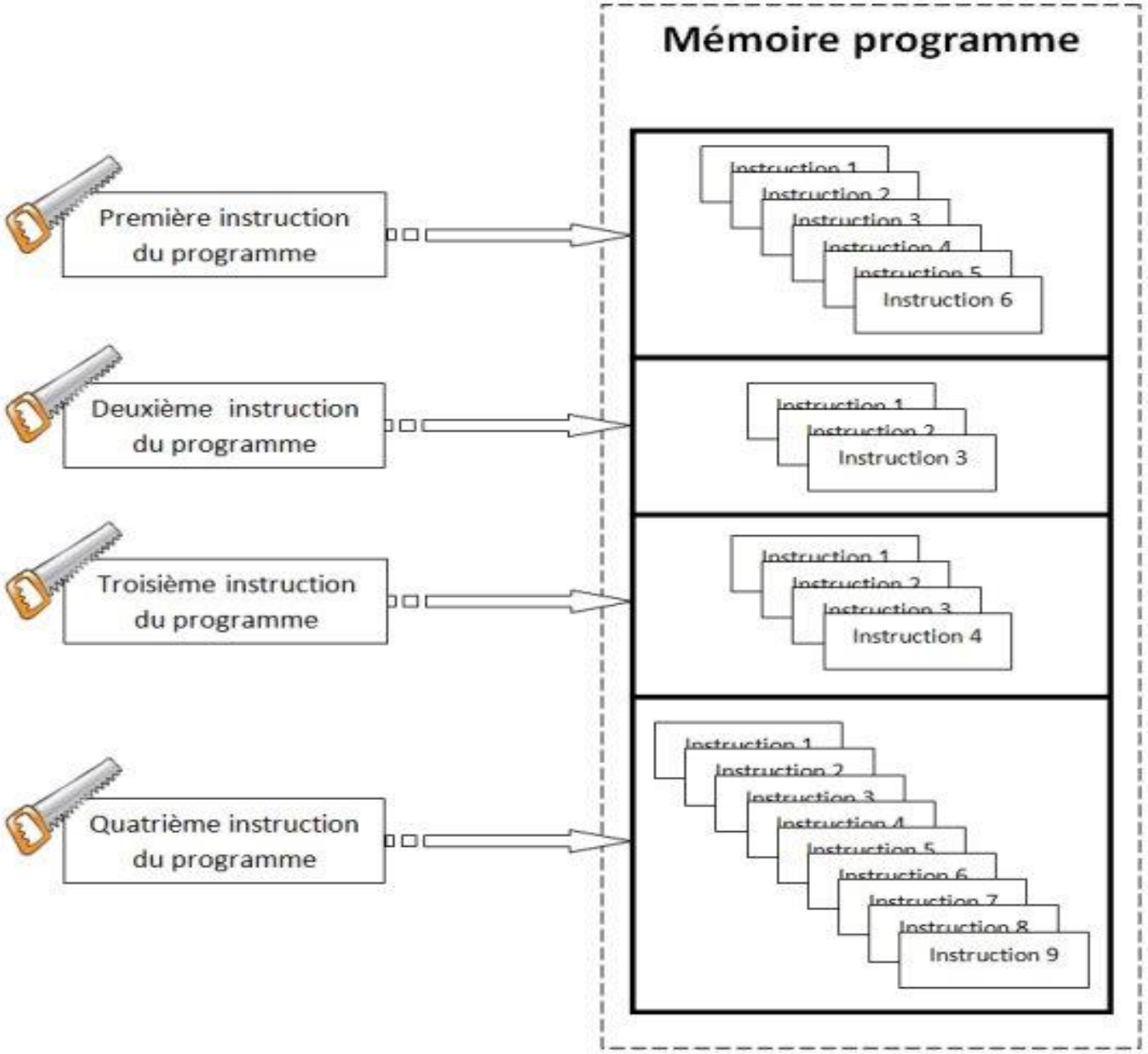
```
1 digitalWrite(led_rouge, LOW); // écriture en sortie (broche 2) d'un état BAS
```

```
// définition de la broche 2 de la carte en tant que variable
const int led_rouge = 2;

// fonction d'initialisation de la carte
void setup()
{
    // initialisation de la broche 2 comme étant une sortie
    pinMode(led_rouge, OUTPUT);
}

// fonction principale, elle se répète (s’exécute) à l'infini
void loop()
{
    // écriture en sortie (broche 2) d'un état BAS
    digitalWrite(led_rouge, LOW);
}
```

Programme Arduino



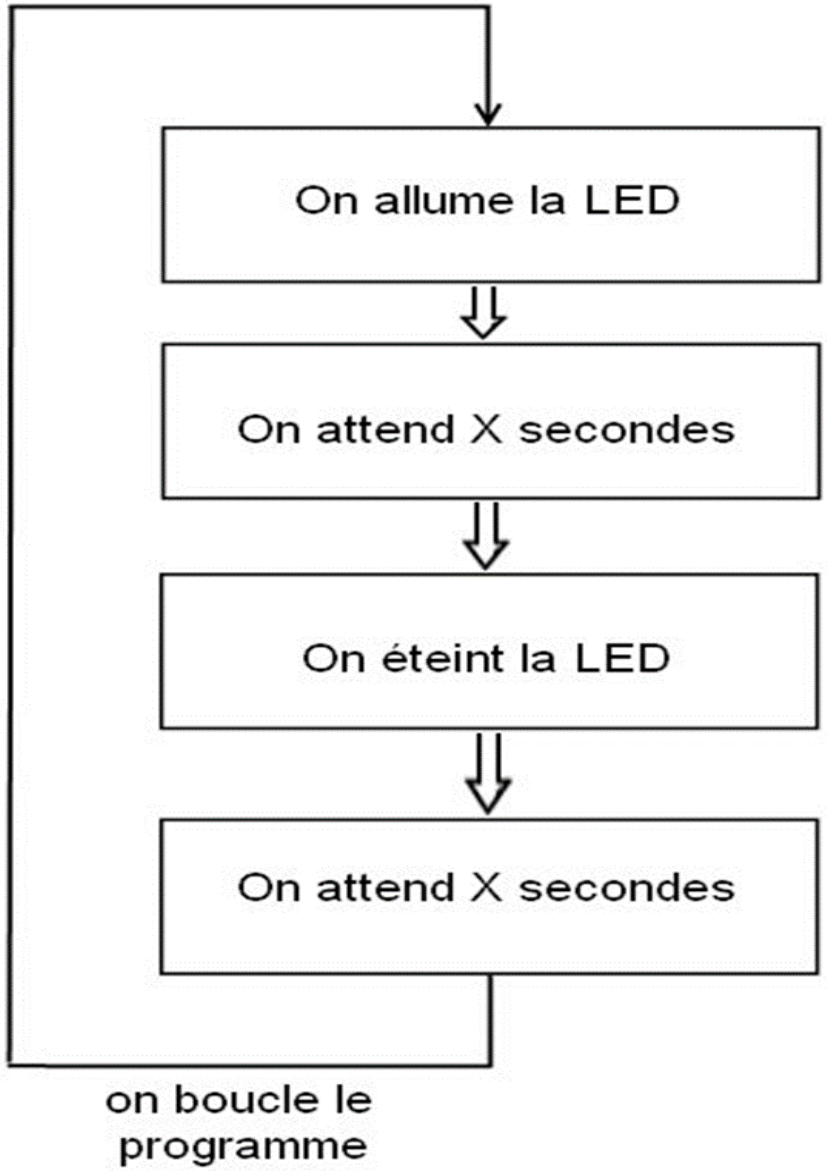
Introduire le temps

- **Trouver la commande…delay()**

La fonction admet un paramètre qui est le temps pendant lequel on veut mettre en pause le programme. Ce temps doit être donné en millisecondes. C'est-à-dire que si vous voulez arrêter le programme pendant une seconde, il va falloir donner à la fonction ce même temps, écrit en millisecondes, soit 1000ms. La fonction est simple à utiliser :

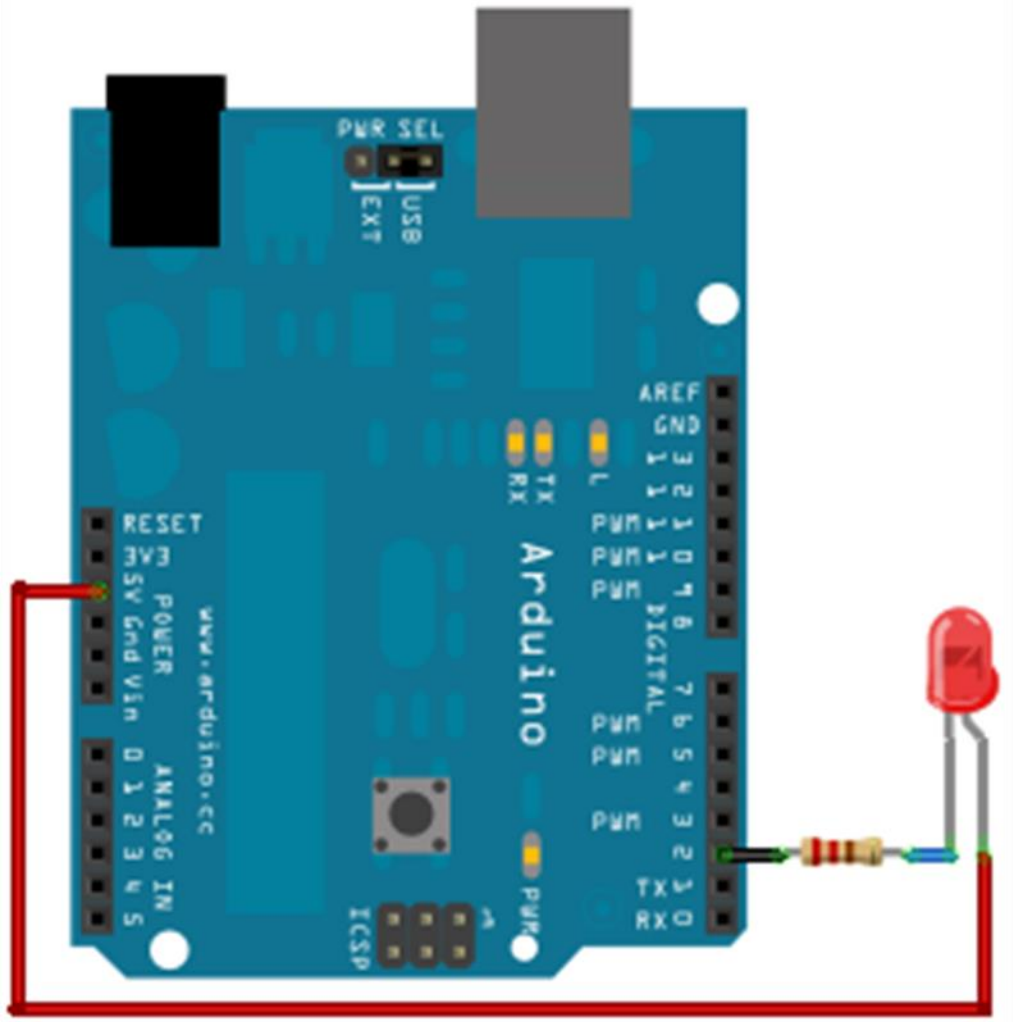
```
1 // on fait une pause du programme pendant 1000ms, soit 1 seconde
2 delay(1000);
```

Mettre en pratique : faire clignoter une LED



Programme

```
1 // allume la LED
2 digitalWrite(led_rouge, LOW);
3 // fait une pause de 1 seconde
4 delay(1000);
5 // éteint la LED
6 digitalWrite(led_rouge, HIGH);
7 // fait une pause de 1 seconde
8 delay(1000);
```




```
1  // définition de la broche 2 de la carte en tant que variable
2  const int led_rouge = 2;
3
4  // fonction d'initialisation de la carte
5  void setup()
6  {
7    // initialisation de la broche 2 comme étant une sortie
8    pinMode(led_rouge, OUTPUT);
9  }
10
11 void loop()
12 {
13   // allume la LED
14   digitalWrite(led_rouge, LOW);
15   // fait une pause de 1 seconde
16   delay(1000);
17   // éteint la LED
18   digitalWrite(led_rouge, HIGH);
19   // fait une pause de 1 seconde
20   delay(1000);
21 }
```