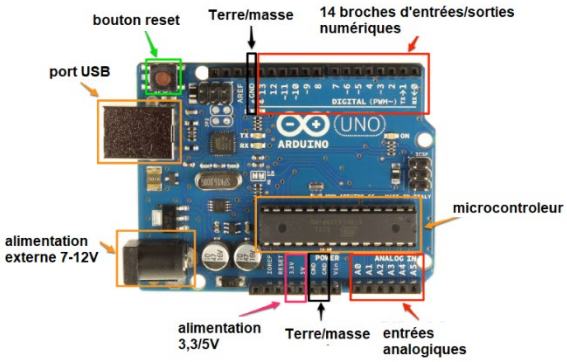
Fiche 2 : Utilisation d'une platine de développement : Arduino

▪ Une platine de développement est un circuit imprimé équipé d'un **microcontrôleur**, dispositif « cerveau » de la platine capable de recevoir un programme, le stoker dans sa mémoire avant de l'exécuter.



▪ Grace à ce programme, le microcontrôleur va pouvoir, par exemple, faire clignoter une LED, afficher des caractères sur un écran, envoyer des données à un ordinateur, mettre en route ou arrêter un moteur….

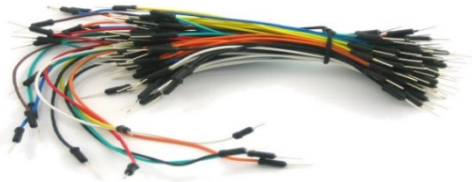
A : Les éléments principaux du montage électronique

▪ Afin de réaliser des montages complets, différents éléments sont utilisés en plus de la platine

► L'alimentation

▪ Pour fonctionner, une carte Arduino a besoin d'une alimentation. Le microcontrôleur fonctionnant sous 5V, la carte peut être alimentée en 5V par le port USB ou bien par une alimentation externe qui est comprise entre 7V et 12V. Un régulateur se charge ensuite de réduire la tension à 5V pour le bon fonctionnement de la carte.

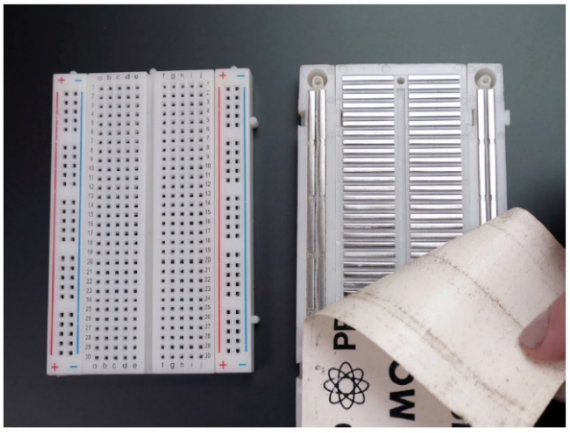
► La connectique



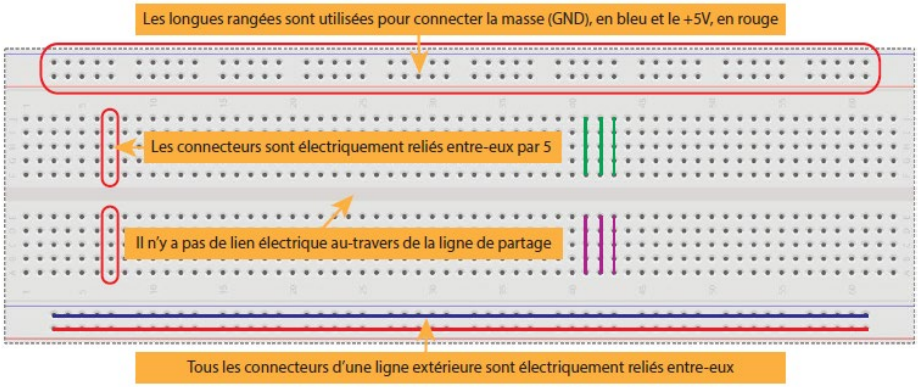
▪ A part une LED sur la broche 13, la carte Arduino ne possède pas de composants (résistances, diodes, moteurs...) qui peuvent être utilisés pour un programme. Il est nécessaire de les rajouter. Mais pour cela, il faut les connecter à la carte. C'est là qu'interviennent les connecteurs, aussi appelés broches (pins, en anglais).
Les connexions entre les composants sont réalisées par des jumpers, sortes de petits câbles.

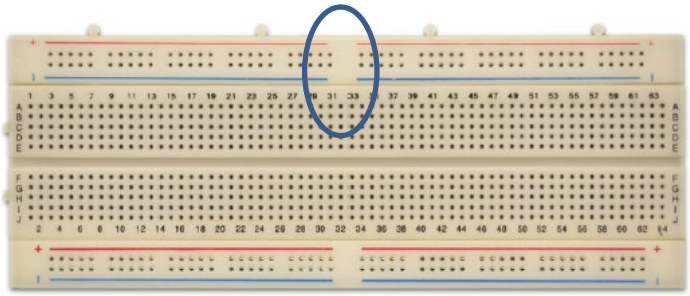
► La platine d'expérimentation

▪ Une platine d'expérimentation (appelée breadboard) permet de réaliser des prototypes de montages électroniques sans soudure et donc de pouvoir réutiliser les composants.

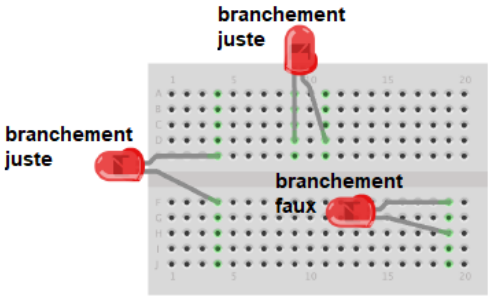


▪ Tous les connecteurs dans une rangée de 5 sont reliés entre eux. Donc si on branche deux éléments dans un groupe de cinq connecteurs, ils seront reliés entre eux. Il en est de même des alignements de connecteurs rouges (pour l'alimentation) et bleus (pour la terre). Ainsi, les liens peuvent être schématisés ainsi:





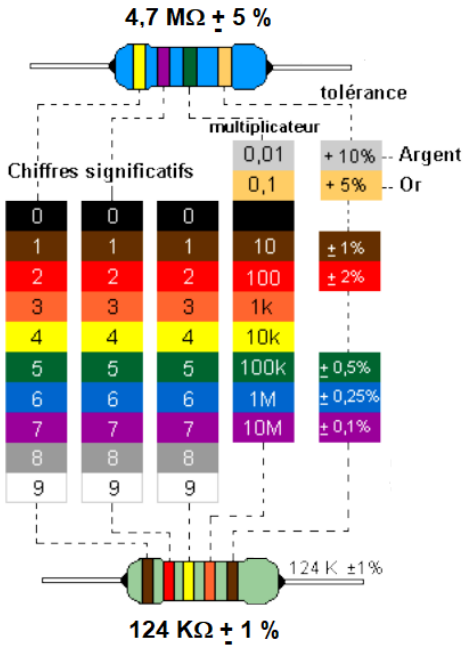
Attention sur certaines plaques les longues rangées des lignes extérieures sont interrompues au milieu de la plaque



▪ Les composants doivent ainsi être à cheval sur des connecteurs qui n'ont pas de liens électriques entre eux, comme sur le schéma ci-contre.

► La diode

▪ La DEL (diode électroluminescente) a la particularité de ne laisser passer le courant que dans un sens : de l'anode vers la cathode
On reconnaît l'anode, car il s'agit de la broche la plus longue. Lorsque les deux broches sont de même longueur, on peut distinguer l'anode de la cathode, par un méplat du côté de cette dernière. Le symbole de la LED est donné ci-contre:



Attention: *le courant produit par l'Arduino est trop important pour y brancher directement une LED dessus.*
L'utilisation d'une résistance est obligatoire, pour ne pas griller la LED.

► Les résistances

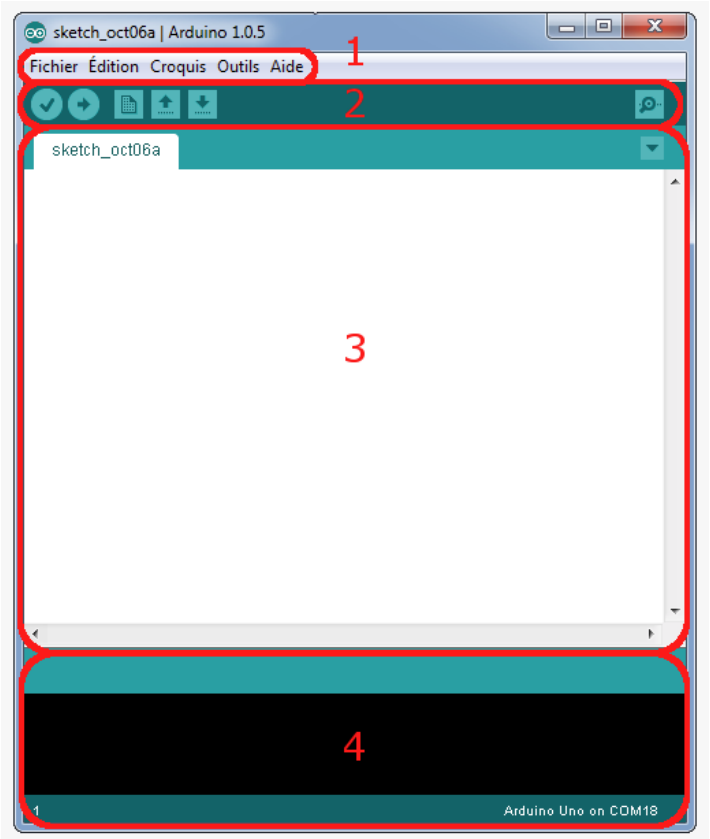
▪ Une résistance est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms: Ω) à la circulation du courant électrique.

Calculateur de valeur de résistance
http://edurobot.ch/resistance/

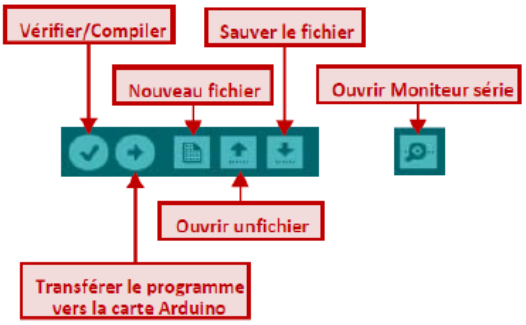
B : Eléments de programmation

▪ Afin de faire fonctionner le montage électronique, il faut rédiger un programme dans un logiciel installé sur un ordinateur qui sera ensuite envoyé (« téléversé ») dans le micro-controleur via un câble USB.

► La fenêtre du logiciel



(1) Barre de menu
(2) Barre des boutons
(3) Fenêtre d'édition des programmes
(4) Zone de messages des actions en cours et console d'affichage des messages de compilation



Le bouton « vérifier/compiler » permet de vérifier le programme ; il actionne un module qui cherche les erreurs dans le programme

► Structure du programme

▪ Un programme destiné à une carte ARDUINO est constitué de 3 parties :

- **La première partie** permet de définir les constantes et les variables en déclarant leur type.

- **La fonction setup()** contient les instructions d'initialisation ou de configuration des ressources de la carte comme par exemple, la configuration en entrée ou sorties des broches d'E/S numériques, la définition de la vitesse de communication de l'interface série, etc..

Tout le code entre les 2 accolades sera exécuté une fois au démarrage du programme

- **La fonction loop()** contient les instructions du programme à proprement parlé.

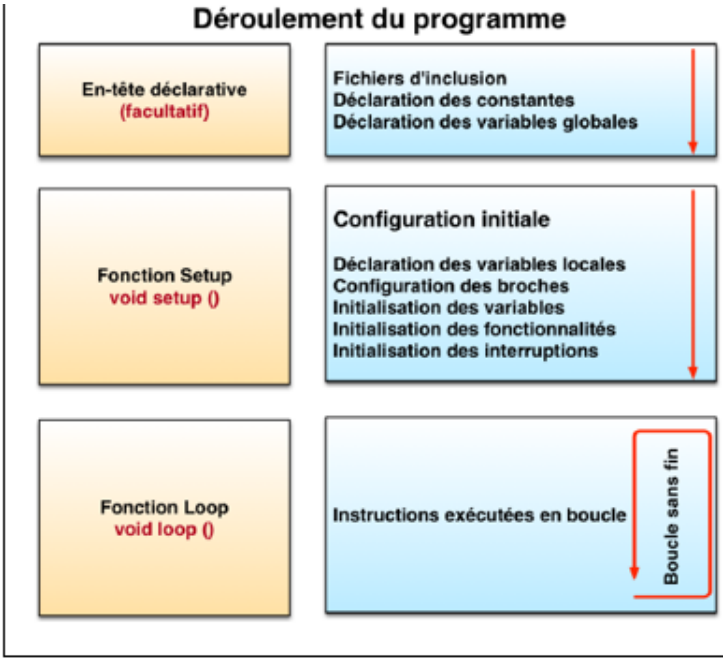
Cette fonction s'exécute après la fin du setup ; après une première exécution, il est ré-exécuté encore et encore jusqu'à ce que l'alimentation soit coupée.

Zone de définition des constantes ou des variables ou d'inclusion des bibliothèques

```
/* Ce programme fait clignoter une LED branchée sur la broche n°13
et fait également clignoter la diode de test de la carte */

int ledPin = 13;              // LED connectée à la broche n°13

void setup()
{
      pinMode(ledPin,OUTPUT) ;        // Définit la broche n°13 comme une sortie
}

void loop()
{
      digitalWrite(ledPin,HIGH) ;     // Met la sortie ledPin au NL1 (diode allumée)
      delay(3000) ;                   // Attendre 3 s
      digitalWrite(ledPin,LOW) ;      // Met la sortie ledPin au NL0 (diode éteinte)
      delay(1000) ;                   // Attendre 1 s
}
```

Fonction setup()
qui contient les instructions d'initialisation

Fonction loop()
qui contient les instructions du programme



Le programme se déroule de la façon suivante :

1. Prise en compte des instructions de la partie déclarative
2. Exécution de la partie configuration (*fonction setup( )*),
3. Exécution de la boucle sans fin (*fonction Loop ( )*): le code compris dans la boucle sans fin est exécuté indéfiniment.

► Les commentaires

- Il est souvent très utile d'écrire des notes au fur et à mesure de l'élaboration du programme.

Pour placer des commentaires sur une ligne unique ou en fin de ligne, il faut utiliser la syntaxe suivante :

```
// Cette ligne est un commentaire sur UNE SEULE ligne
```

Pour placer des commentaires sur plusieurs lignes :

```
/* commentaire sur PLUSIEURS lignes qui sera ignoré par le programme, mais pas par
celui qui lit le code */
```

Tout ce que l'on inscrit après 2 slashs sera ignoré par le programme.

► Quelques éléments de la syntaxe ARDUINO

- La cinquantaine d'éléments de la syntaxe ARDUINO est visible ici : https://www.arduino.cc/reference/en/

QUELQUES TYPES DE VARIABLES	
Nom	**Description**
`boolean`	Donnée logique (true ou false) sur 8 bits
`byte`	Nombre entier positif sur 8 bits (de 0 à 255)
`int`	Nombre entier (positif et négatif) sur 16 bits
`unsigned int`	Nombre entier positif sur 16 bits
`float`	Nombre décimal sur 32 bits
`long`	Nombre entier (positif et négatif) sur 16 bits (de -65535 à 65535)

DEFINITION D'UNE CONSTANTE	
Nom	**Description**
`#define N2`	La variable N rencontrée dans le programme sera remplacée par la valeur 2
`Const byte N=2`	La variable N rencontrée dans le programme est de type « byte » et sera remplacée par la valeur 2

Un certain nombre de noms de constantes sont prédéfinis dans le langage « ARDUINO » :

Nom	Description
`true`	Donnée binaire égale à 1 ou donnée numérique différente de 0
`false`	Donnée binaire ou numérique égale à 0
`HIGH`	Génération d'un niveau de tension haut sur une des sorties numériques
`LOW`	Génération d'un niveau de tension bas sur une des sorties numériques
`INPUT`	Configuration d'une broche numérique en entrée
`OUTPUT`	Configuration d'une broche numérique en sortie

OPERATEURS ARITHMETIQUES		
Symbole	**Description**	**Exemple**
+	Addition	c = a + b ;
-	Soustraction	c = a - b ;
*	Multiplication	c = a * b ;
/	Division	c = a / b ;

OPERATEURS LOGIQUES		
Symbole	**Description**	**Exemple**
+	Addition	c = a + b ;
-	Soustraction	c = a - b ;
*	Multiplication	c = a * b ;
/	Division	c = a / b ;

Symbole	Description
{ }	On les utilise pour définir où commence et où se termine un bloc de code (utilisé dans les fonctions aussi bien que dans les boucles) A utiliser lorsque le commentaire occupe plusieurs lignes ; tout ce qui se trouve entre ces 2 symboles sera ignoré par le programme.
;	**Chaque ligne doit se terminer par un point-virgule (l'oubli d'un point-virgule est souvent la raison d'un refus du programme de compiler)**