

NIVEAU DEBUTANT

Apprendre à utiliser les variables et les constantes avec le langage Arduino. Afficher des variables numériques entières dans le Terminal Série.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'avez pas payé pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

1. Intro

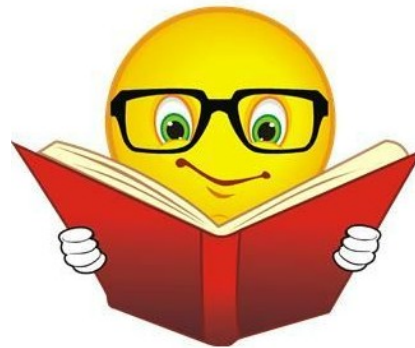
L'objectif ici est :

- de comprendre le concept de variable
- de comprendre la notion de « type » de variable
- d'apprendre à déclarer une variable
- de comprendre comment utiliser les variables avec les fonctions,
- d'apprendre à déclarer une constante
- de savoir afficher et visualiser le contenu de variables dans le Terminal Série,
-

... afin d'être en mesure d'utiliser et manipuler efficacement les variables et les constantes dans un programme Arduino.

Avertissement

Si vous ne comprenez pas tout, ne vous inquiétez pas : il s'agit ici d'une présentation de notions importantes que vous allez rencontrer souvent :
vous aurez donc tout loisir ultérieurement de tester, expérimenter à votre rythme et tranquillement tout ce que vous allez découvrir...
Vous vous familiariserez progressivement avec les différentes instructions Arduino et vous comprendrez alors concrètement la signification de ces concepts.



Prêt ? C'est parti !

Pratique :

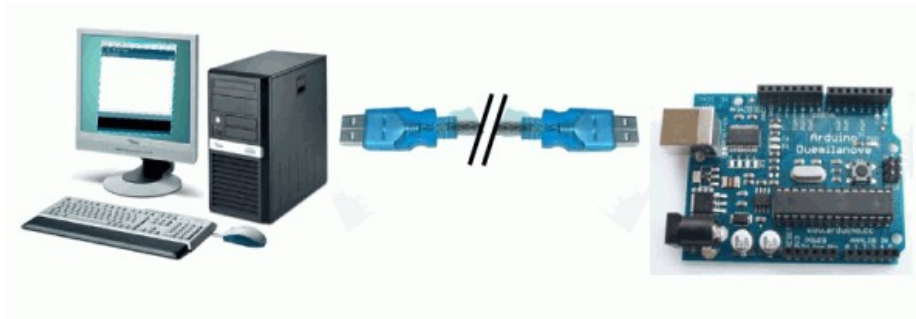
Les codes de cet atelier sont disponibles ici :

http://www.mon-club-elec.fr/mes_downloads/tutos_arduino/1b.atelier_arduino_variables.tar.gz

2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino

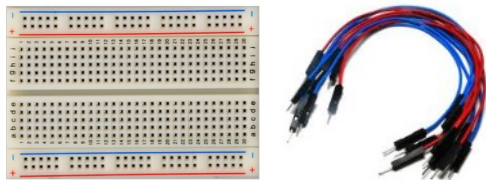


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

Du nécessaire pour réaliser des montages sans soudure

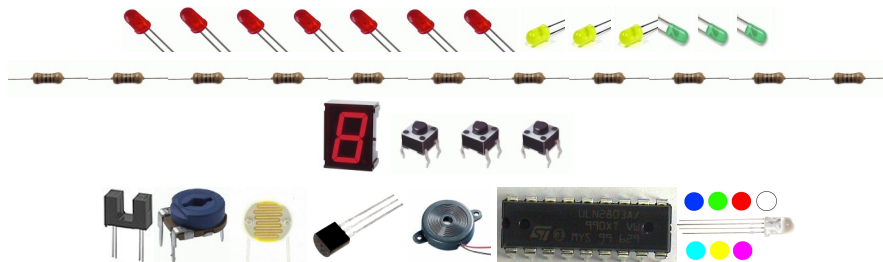


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

De quelques composants de base



Pour vous simplifier la vie, nous avons négocié ce kit pour vous !

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire
<http://www.gotronic.fr/> avec le code express **701710**

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

3. La notion de variable

Vous avez tous déjà rangé des objets dans des boîtes et collé une étiquette dessus : alors vous allez facilement comprendre ce qu'est une variable !

Lorsque l'on écrit un programme en langage Arduino (ou dans n'importe quel autre langage de programmation d'ailleurs), on va manipuler des nombres entiers, des nombres à virgules, des chaînes de caractères.

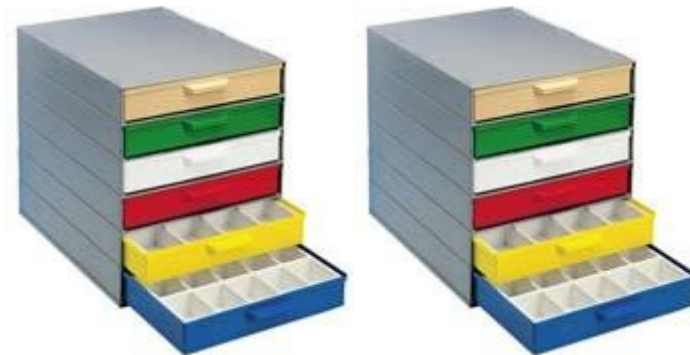
Pour pouvoir manipuler facilement toutes ces grandeurs, on va les mémoriser, les stocker, ce qui revient à les « ranger chacun dans une boîte » et on va « coller une étiquette dessus ».

En programmation, une « boîte mémoire » sur laquelle on colle une « étiquette » pour y mettre quelque chose : ça s'appelle une **variable !**

La carte Arduino dispose d'un processeur qui sait très bien utiliser ces « boîtes » mémoires : c'est même son principal travail !

La carte Arduino possède de nombreuses cases mémoires disponibles que vous allez pouvoir utiliser à volonté :

- vous disposez de plus de 1000 « boîtes » (ou octets) dans lesquelles vous allez pouvoir lire, écrire, effacer des valeurs ,
- cette « réserve » de boîtes, c'est la mémoire vive, ou RAM (*active lorsque Arduino est allumé et qui s'efface lorsque vous éteignez Arduino*). C'est elle que le programme Arduino utilise pour stocker les variables !
- pour info (usage avancé), la carte Arduino dispose également :
 - d'une petite mémoire Eeprom dans laquelle on peut lire/écrire et qui ne s'efface pas quand on éteint l'Arduino
 - d'une très grosse mémoire FLASH qui est écrite une fois pour toute et qui ne s'efface pas quand on éteint Arduino. On y met le programme mais on peut aussi y mettre des données fixes (texte par exemple).



4. La notion de « type » de variables

Imaginons que vous vouliez ranger des objets de toutes sortes : vous allez vous préparer un stock de boîtes de tailles diverses et ensuite vous allez ranger vos objets de façon adaptée dans vos boîtes.

Par exemple :

- vous allez mettre un bijou dans une petite boîte, vous mettrez des crayons dans une boîte moyenne, vous mettrez des livres dans une grosse boîte, etc...
- dans votre cuisine, vous mettrez un petit reste dans un petit tupperware, des oeufs dans une boîte à oeufs, du liquide dans une bouteille, etc...

En programmation, c'est exactement pareil :

- vous allez utiliser des « boîtes mémoires » différentes en fonction de la « taille » ou de la nature de ce que vous allez mettre dedans.
- les types possibles vont être des valeurs numériques entières (1,2,3,...), des nombres à virgules (3.14159), des caractères ('A', 'B', ..), des valeurs binaires (0 ou 1, vrai ou faux), etc..

En programmation, la « taille » d'une « boîte mémoire » (ou variable) s'appelle le « type ». Pour dire les choses autrement, le type d'une variable, c'est sa catégorie, son genre .

Le langage Arduino va vous permettre :

- de **créer des variables de types différents** en fonction des besoins
- de **les nommer**
- d'y **mettre les valeurs** voulues



5. Principe général de déclaration d'une variable

Dans le langage Arduino, pour déclarer une variable, il faut au minimum :

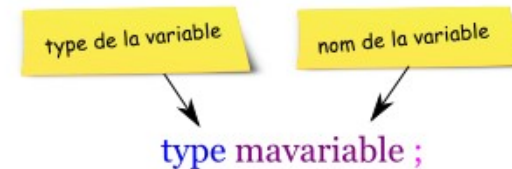
- indiquer son **type** (= ce que va contenir la « boîte »)
- indiquer son **nom** (= l'étiquette à coller sur la « boîte »)
- finir la ligne par un **point-virgule ;** (= ne pas l'oublier !)
- et c'est tout !
- Noter que le contenu initial de cette variable est inconnu... ce qui n'est jamais une bonne chose en programmation...

C'est pourquoi, le plus souvent, on préférera également fixer la valeur de la variable au moment de sa déclaration. Dans ce cas, il faudra :

- indiquer son **type** (= ce que va contenir la « boîte »)
- indiquer son **nom** (= l'étiquette à coller sur la « boîte »)
- suivi d'un signe égal = (ce qui revient à ouvrir la boîte pour mettre quelque chose dedans...)
- suivi de la **valeur initiale voulue**, (= ce que vous mettez dans la « boîte »)
- et finir la ligne par un **point-virgule ;** (= ne pas l'oublier !)

En pratique : initialisez vos variables (= attribuez une valeur connue) lors de la déclaration pour éviter les effets inattendus !

Déclaration d'une variable sans initialisation.



Déclaration d'une variable avec initialisation.



6. Pour info : Les principaux types de variables disponibles dans le langage Arduino

Vous connaissez déjà un type, le type **void**, qui est le type d'une fonction qui ne renvoie rien. On ne peut pas déclarer une variable void, puisque par définition une variable va contenir quelque chose.

Le type de variable le plus simple est celui d'une variable dite « **binaire** » qui ne peut contenir que rien ou quelque chose, autrement dit 0 ou 1, vrai ou faux : c'est le type **boolean**

Les types de variables que vous allez le plus utiliser sont les variables numériques entières :

- le type **int** (pour integer = entier) pour une **valeur entière** comprise entre -32536 et + 32535
- le type **long** pour une **valeur entière** comprise entre -2 147 483 648 et +2 147 483 647
- usage avancé : on fera précéder le type int ou long du mot clé unsigned si on souhaite n'utiliser que des valeurs positives. Ceci a pour effet de doubler la valeur positive utilisable (de 0 à 65535 pour un int par exemple).

Une variable qui va contenir un **nombre à virgule** sera de type **float**

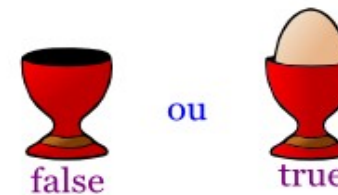
Une variable correspondant à **un caractère** sera de type **char**

A part, la classe **String** qui servira à stocker **les chaînes de caractères**.

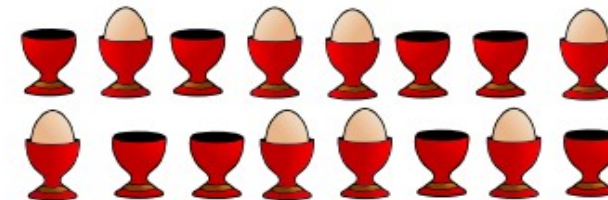
Les types vous sont présentés ici pour information : vous les utiliserez et les découvrirez au fur et à mesure des besoins de vos programmes.

En pratique, au début, déclarez vos variables en type **int : ça couvrira tous les besoins courants !**

Le type boolean



Le type int



Le type long



7. A vous de jouer : Déclarations de variables et opérations de bases sur les variables

Exemples de déclarations de variables

Pour stocker une valeur pouvant prendre comme valeur « vrai ou faux », 1 ou 0, on pourra utiliser un boolean :

```
boolean maVariable=false;
```

Pour stocker une valeur entière pouvant prendre comme valeur maximale 1000, on utilisera un int :

```
int maVariable=0;
```

Pour stocker une valeur entière pouvant prendre comme valeur maximale 1 000 000, on utilisera un long :

```
long maVariable=0;
```

Pour stocker une valeur décimale, on utilisera un float :

```
float maVariable=0.0;
```

En pratique : déclarez vos variables en type `int` : ça couvrira tous les besoins courants ! initialisez vos variables lors de la déclaration pour éviter les effets inattendus !

Exemples d'opérations de base sur les variables

Pour attribuer une valeur à une variable, on fera :

```
maVariable=2500;
```

Entre 2 valeurs numériques, on pourra réaliser des opérations mathématiques. Si a, b et c sont 3 variables du même type (int par exemple), on pourra faire :

```
a=a+b;
```

```
c=b-a;
```

```
c=3*(a-b);
```

```
b=a/c;
```

Noter que la division renverra une valeur entière si on utilise 2 valeurs entières.

Usage avancé : une variable a une taille qui se mesure en « octets » (un ensemble 8 coquettiers...) Il est possible de connaître la taille d'une variable grâce à l'instruction `sizeof` (usage avancé) selon :

```
int taille = sizeof(maVariable); // récupère la taille en octets d'une variable
```

Bon à savoir : convertir un type dans un autre.

Dans certaines situations, on peut avoir besoin de transformer un type de variable dans un autre. Ceci est possible grâce à plusieurs fonctions dédiées du langage arduino : `int(x)`, `long(x)`, `float(x)`, `char(x)`, etc... où x représente une variable de tout type.

8. Où déclarer une variable ? Notion de portée des variables

Il est possible de déclarer une variable :

- soit **avant la fonction `setup()`**, en dehors de toute fonction, dans une partie que l'on pourra appeler « entête déclarative ». Les variables déclarées de cette façon seront accessibles de n'importe où dans le programme. On dit qu'elle sont « **globales** ».
- soit **au sein d'une fonction**, que ce soit la fonction `setup()`, `loop()` ou toute autre fonction. Dans ce cas, la variable n'existe que à l'intérieur de cette fonction : on dit qu'elle est « **locale** ».
- soit **au sein de certaines boucles**, comme nous le verrons plus tard, en tant que variable d'incrémentement : la portée est locale et se limite alors à l'intérieur de la boucle.

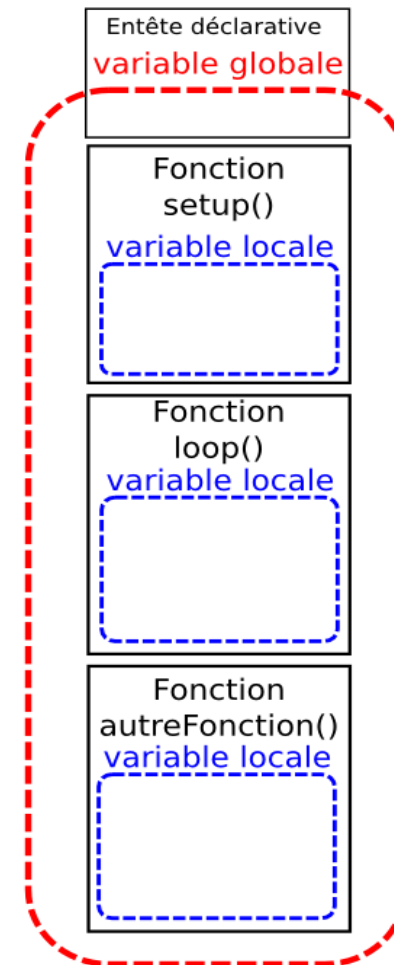
D'après ce que nous venons de voir, vous comprenez que la **portée d'une variable** correspond à son « périmètre de visibilité » dans un programme :

- **globale** : la variable est **visible et utilisable de n'importe quel endroit** du programme.
- **locale** : variable n'est **visible et utilisable que dans une certaine partie du programme**, une fonction ou une boucle. *A noter : une variable locale est détruite entre 2 appels de la fonction qui l'utilise.*

En pratique : déclarez vos variables et constantes AVANT les fonctions `setup()` et `loop()` : elles seront ainsi globales et accessibles de partout !

*Usage avancé : La portée des variables et leur durée de vie est modifiable également par certains mots-clés, appelé « **qualificateurs** », à mettre avant le type de la variable lors de la déclaration :*

- ***static** : permet de rendre une variable locale persistante entre 2 appels d'une fonction utilisant une variable locale. Est inaccessible en dehors de la fonction qui l'utilise.*
- ***volatile** : rend persistante une variable globale quelque soit le moment où elle sera appelée. Utile avec les interruptions.*



9. Les « variables fixes » : les constantes !

Pour tous les types de variables, il existe une situation particulière : celle où le contenu de la variable ne peut et ne doit pas changer au cours du programme.

Par exemple :

- si la variable représente le numéro d'une broche,
- si la variable représente le nombre PI, etc...

Dans cette situation particulière, on va pouvoir préciser que le contenu de la variable ne changera pas au cours du programme en créant une **constante** ! (à noter que ce n'est pas obligatoire, mais c'est mieux !)

Pour déclarer une constante, c'est simple : on ajoute le mot clé **const** devant type de la variable qui devient ainsi une constante ! Exemple :

```
const int maBroche=13; // déclare une constante de type int valant 13
```

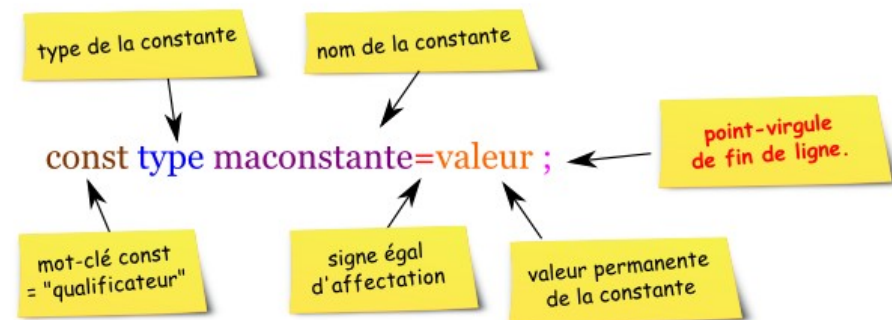
Une fois déclarée, la valeur d'une constante ne pourra plus être modifiée !

A savoir : Le langage Arduino intègre plusieurs constantes prédéfinies :

- **HIGH** et **LOW** correspondant au niveau des broches numériques
- **INPUT** et **OUTPUT** correspondant au sens des broches numériques
- **true** et **false** correspondant aux états logiques VRAI et FAUX

En pratique : utiliser une constante pour renommer une broche Arduino !

Déclaration d'une constante (initialisation obligatoire !).



10. Variables et fonctions : 1. passer une variable en tant que paramètre d'une fonction !

Il est possible d'utiliser les variables avec les fonctions et donc avec les instructions Arduino, de la même façon qu'on le ferait avec des valeurs numériques brutes. On va ainsi pouvoir :

- passer une variable en tant que paramètre d'une fonction,
- récupérer le résultat renvoyé par une fonction dans une variable.

Fréquemment, on va « passer » à une fonction une variable en tant que paramètre. Dans ce cas, **il faut impérativement que la variable passée en paramètre soit du même type que ce que la définition de la fonction prévoit pour ce paramètre** (autrement dit utiliser une variable int si la fonction attend un paramètre de type int).

Dans le cas des instructions Arduino, la documentation indique le type attendu pour chaque paramètre de la fonction :

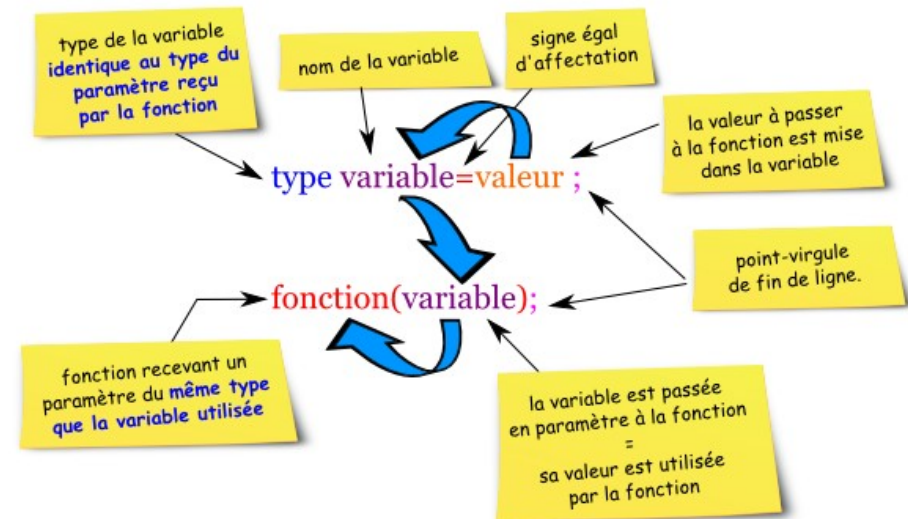
- Par exemple, la fonction `delay()` du langage Arduino reçoit un paramètre correspondant à la durée en millisecondes.
- La documentation nous indique que ce paramètre doit être de type `int`.
- On pourra ainsi déclarer une variable `int` et la passer à l'instruction `delay()`

Exemple :

```
int duree=1000; // déclare une variable int valant 1000
delay(duree); // exécute l'instruction delay avec la valeur de la variable
```

La plupart des instructions du langage Arduino supportent le type `int`.

Utiliser une variable en tant que paramètre d'une fonction.



11. Variables et fonctions : 2. récupérer le résultat renvoyé par une fonction dans une variable !

A l'inverse, de nombreuses instructions ou fonctions renvoient une valeur résultat d'un type donné (int, long, etc...) : **on pourra récupérer le résultat renvoyé par une fonction dans une variable du même type, simplement en posant l'égalité entre la variable et la fonction.**

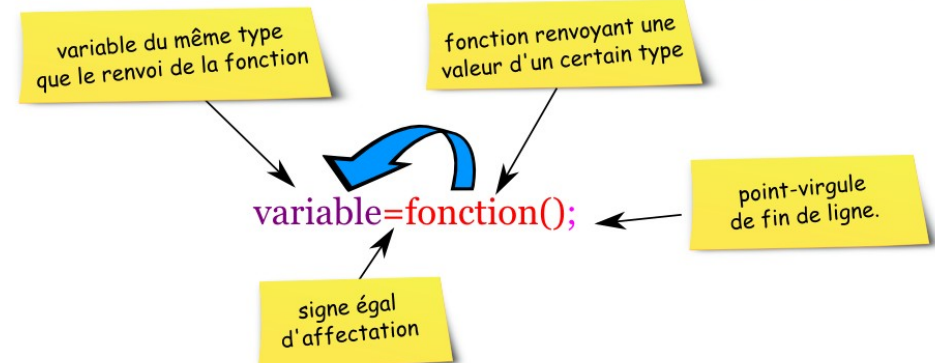
Là encore, pour chaque fonction, la documentation du langage Arduino précise quel est le type de la valeur renvoyée :

- Par exemple, la fonction **millis()** du langage Arduino renvoie une valeur correspondant au nombre de millisecondes écoulées depuis le début du programme.
- La documentation nous indique que la valeur renvoyée est de type **long**.
- On pourra ainsi déclarer une variable **long** et récupérer le nombre de millisecondes écoulées depuis le début du programme dans la variable, simplement en posant l'égalité entre la variable et la fonction !

Exemple :

```
long time=0; // déclare une variable time valant 0  
time = millis(); // le résultat renvoyé par la fonction est mis dans la variable
```

Récupérer le résultat d'une fonction dans une variable.



Encore une fois...

Si vous n'avez pas tout compris, ne vous inquiétez pas : il s'agit ici d'une présentation de notions importantes que vous allez rencontrer souvent : vous aurez donc tout loisir ultérieurement de tester, expérimenter à votre rythme et tranquillement tout ce que vous venez de voir... Vous vous familiariserez progressivement avec les différentes instructions Arduino et vous comprendrez alors concrètement la signification de ces concepts.

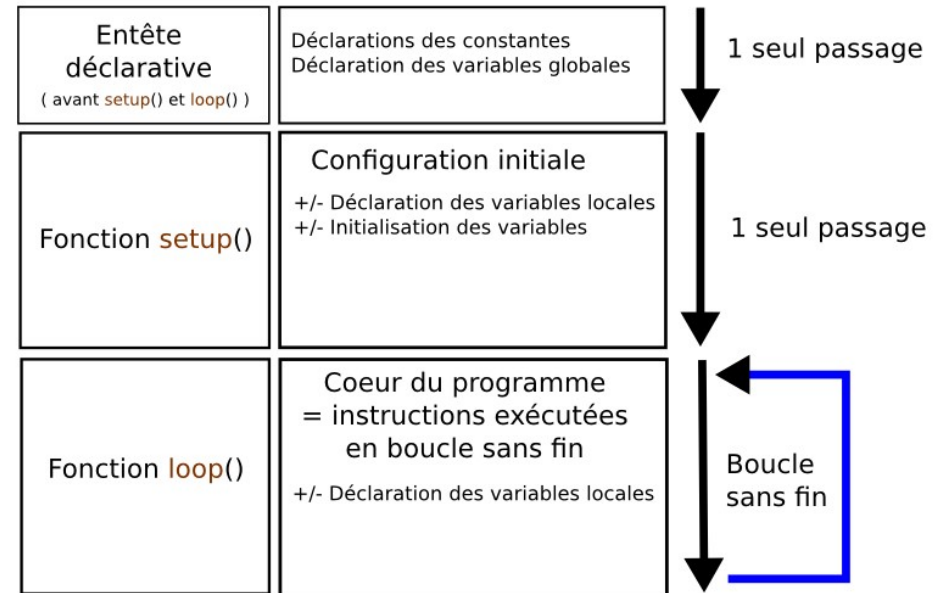
12. Structure type d'un programme Arduino utilisant des variables et des constantes.

Souvenez-vous, la structure du programme minimum Arduino associait successivement :

- la fonction `setup()` qui ne s'exécute qu'une seule fois
- suivie de la fonction `loop()` qui s'exécute ensuite en boucle sans fin

A présent, la structure d'un programme Arduino « type » (que je vous conseille vivement d'utiliser) va s'étoffer un petit peu avec :

- une **entête déclarative** avant les 2 fonctions `setup()` et `loop()` au niveau de laquelle on déclarera :
 - les constantes dont la valeur ne changera pas et seront globales,
 - et les variables globales qui seront accessibles de partout.
- la **fonction `setup()`** qui ne s'exécute qu'une seule fois dans laquelle on pourra :
 - également déclarer des variables locales si besoin,
 - initialiser si besoin les variables globales,
 - utiliser les constantes et les variables globales à volonté
- suivie de la **fonction `loop()`** qui s'exécute ensuite en boucle sans fin, dans laquelle on pourra :
 - également déclarer des variables locales également si besoin,
 - utiliser les constantes et les variables globales à volonté.



13. Afficher des variables dans le Terminal Série

Dans un programme qui va afficher des messages utilisant des variables dans le Terminal série sur le PC, la structure sera la suivante :

Entête déclarative

A ce niveau, on va déclarer et initialiser les variables utilisées, par exemple :

- une variable **int**
- une variable **long**
- une variable **float**
- une constante **int**

Fonction setup()

A ce niveau, on va :

- initialiser la communication série :
- +/- afficher les messages de début de programme (1 seul affichage)

Fonction loop()

On ce niveau on va :

- afficher les messages utilisant les variables
- faire une pause de 1 seconde entre 2 passages.

```
// --- entete déclarative ---
int myInt=-123; // variable entière "petit format"
long myLong=456789; // variable entière "grand format"
float myFloat=3.14159; // variable à virgule
const int myConst=10; // constante entière

void setup() { // fonction setup() : exécutée 1 seule fois en premier

    Serial.begin(115200); // initialise la communication série

} // fin setup

void loop() { // fonction loop() : exécutée ensuite en boucle sans fin

    Serial.print("myInt = "), Serial.println(myInt); // affiche message
    Serial.print("myLong = "), Serial.println(myLong); // affiche message
    Serial.print("myFloat = "), Serial.println(myFloat,4); // affiche message
    Serial.print("myConst = "), Serial.println(myConst); // affiche message
    Serial.println(); // saut de ligne vide

    delay(1000); // pause de 1 seconde

} // fin loop
```


14. Exercice

- Ecrire un programme Arduino qui incrémente une variable chaque seconde et affiche la valeur dans le Terminal Série

15. Liens d'exemples utiles sur www.mon-club-elec.fr

- [Afficher des messages texte sur le PC](#)
- [Affichage des nombres entiers \(binaire, hexa et décimal\) et à virgule sur le PC](#)
- [Affichage de résultats de fonctions mathématiques sur le PC](#)
- [Manipulations de bits et octets visualisées sur le PC](#)
- [Afficher sur le PC des messages texte stockés en mémoire programme FLASH](#)
- [Afficher la taille des variables du langage Arduino](#)
- [Afficher le comptage simple des secondes dans la fenêtre Terminal du PC \(utilise millis\)](#)

16. Les éléments du langage Arduino étudiés dans cet atelier :

Structure

Opérateurs arithmétiques

- `=` (égalité)
- `+` (addition)
- `-` (soustraction)
- `*` (multiplication)
- `/` (division)

Variables et constantes

Constantes prédéfinies

- `HIGH` | `LOW`
- `INPUT` | `OUTPUT`
- `true` | `false`

Types des données

- `void`
- `boolean`
- `char`
- `int`
- `unsigned int`
- `long`
- `unsigned long`
- `float` (nombres à virgules)

Qualificateurs

- `const`
- `static`, `volatile`

Divers

- `sizeof()` (opérateur `sizeof`)

Fonctions

Temps

- `delay(ms)`
- `millis()`

Communication

- Classe `Serial`
 - `begin()`
 - `print()`
 - `println()`

La documentation complète du langage Arduino en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi

17. A présent, vous devriez être capable :

- De déclarer des variables en fonction de vos besoins dans un programme, notamment de type int.
- Afficher des variables dans le Terminal Série du logiciel Arduino

Table des matières

Apprendre à utiliser les variables et les constantes avec le langage Arduino. Afficher des variables numériques entière dans le Terminal Série.

Intro |

La notion de variable |

La notion de « type » de variables |

Principe général de déclaration d'une variable |

Pour info : Les principaux types de variables disponibles dans le langage Arduino |

A vous de jouer : Déclarations de variables et opérations de bases sur les variables |

Où déclarer une variable ? Notion de portée des variables |

Les « variables fixes » : les constantes ! |

Variables et fonctions : 1. passer une variable en tant que paramètre d'une fonction ! |

Variables et fonctions : 2. récupérer le résultat renvoyé par une fonction dans une variable ! |

Structure type d'un programme Arduino utilisant des variables et des constantes. |

Afficher des variables dans le Terminal Série |

Exercice |

Liens d'exemples utiles sur www.mon-club-elec.fr |

Les éléments du langage Arduino étudiés dans cet atelier : |

A présent, vous devriez être capable : |

Bravo !
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :
http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS