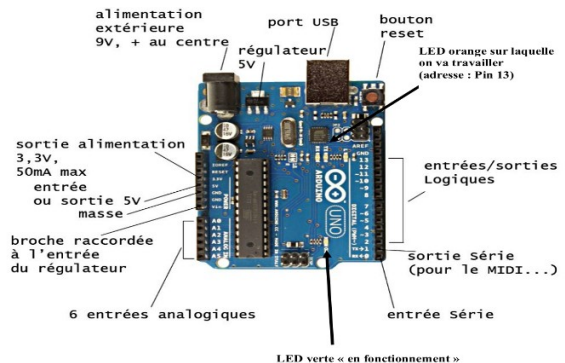


La carte arduino

La carte Arduino Uno que nous allons utiliser est une sorte de petit ordinateur. Il possède un microcontrôleur (microprocesseur + mémoire + interface de gestion des entrées/sorties).

Il possède donc des entrées / sorties logiques (« digital » en anglais), des entrées analogiques (0 à 5V), des moyens de communication série et bien sûr un résonateur à quartz d'une fréquence de 16MHz.



L'algorithmie :

L'algorithme est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné. Pour comprendre, expliquer, analyser un programme nous allons utiliser l'algorithmie sous forme graphique (organigramme) plus facilement compréhensible quand la problématique est simple.

Structure d'un programme Arduino:

La structure de base d'un programme Arduino (sketch) :

variables (déclaration des variables)

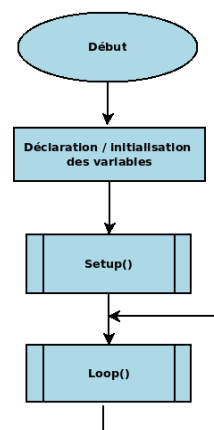
void setup() {

Commandes d'initialisation (entrées, sorties, ...)}

void loop() {

Instructions ;

}



- **A savoir :**

- La structure void setup() n'est exécutée qu'une seule fois.
- La structure void loop() est exécuté indéfiniment (Loop = boucle)
- La ligne de commentaires : // ... tout texte après les deux slashes est un commentaire.
- Le bloc de commentaires : /* */ tout texte entre les deux étoiles est un commentaire, le texte peut s'étendre sur plusieurs lignes.



L'instruction pinMode

pinMode (n° de broche, [OUTPUT ou INPUT]) ; déclare la broche « n° de broche » en entrée ou en sortie. Cela ne concerne que les entrées numériques (appelées aussi Tout Ou Rien. En anglais: digital)

L'instruction digitalWrite:

digitalWrite (n° de broche, [HIGH ou LOW]) met un niveau haut (1) ou bas (0) sur la broche désignée. Il s'agit bien d'une sortie logique appelée aussi TOR (Tout Ou Rien) ou digital (en anglais).

L'instruction delay ():

Delay (100) : le microcontrôleur de la carte Arduino attend 100 millisecondes.

On remarque:



- que les broches numériques ("Digital Pin" en anglais) peuvent être soit des entrées soit des sorties.
- quand on veut savoir ce que fait une instruction on se reporte au site internet Arduino qui est la seule source fiable (site en anglais)

Les variables

Utilisation de variables

En informatique on a l'habitude d'utiliser des variables que l'on affecte aux valeurs que l'on veut et que l'on peut même modifier tout au long d'un programme. Ces variables sont des symboles qui associent un nom (l'identifiant) à une valeur. Ces variables sont définies en tête de programme (on choisit obligatoirement leur type et on peut aussi leur donner une valeur. Cette valeur peut être ensuite modifiée dans le programme.

Chaque variable a donc une place réservée en mémoire.

Type de variable	Taille en mémoire (8 ou 16 ou 32 bits)	Type de nombre	Valeurs min/max
boolean		binaire	0 à 1
char	8	entier	-127 à 127
unsigned char ou byte	8	entier	0 à 255
int	16	entier	-32767 à 32767
unsigned int ou word	16	entier	0 à 65535
long	32	entier	-2 147 483 647 à 2 147 483 647
unsigned long	32	entier	0 à 4 294 967 295
float	32	réel	$-3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$

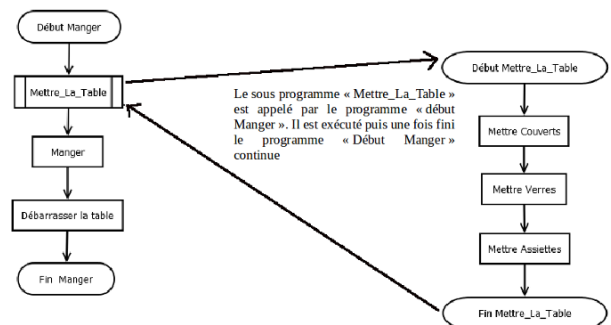
Utilisation de #define

On va aussi utiliser la fonction « #define ». Celle-ci, bien que moins intéressante que la définition précédente, est très utilisée. Elle ne définit pas la variable mais simplement, lors de la compilation, elle remplace le nom de la variable par la valeur écrite à côté.

Les fonctions/sous programmes

Lorsqu'on a un ensemble de lignes de programme qui doivent être exécutées à différents endroits dans un programme, au lieu de réécrire les mêmes lignes de code, il est intéressant de créer des fonctions.

Exemple du quotidien : le programme « manger » fait appel à la fonction « mettre_la_table »



Fonctionnement de la boucle « For »

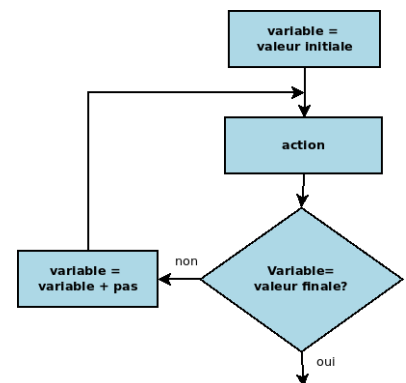
Voici un extrait de la documentation [Arduino](#):

Le fonctionnement de la fonction For :

for (initialisation ; condition ; incrément)

```
{
    bloc d'instructions (actions) ;
}
```

Fonctionnement : l'initialisation se produit en premier et une seule fois. A chaque passage dans la boucle, la condition est testée; si elle est vraie, le bloc d'instruction et l'incrément (le « pas ») sont exécutés, puis la condition est testée à nouveau. Quand la condition devient fausse, la boucle se termine.



Le test conditionnel IF / ELSE

Le principe :

If (Test) { //Si la condition est vrai alors

Instruction 1 exécutée

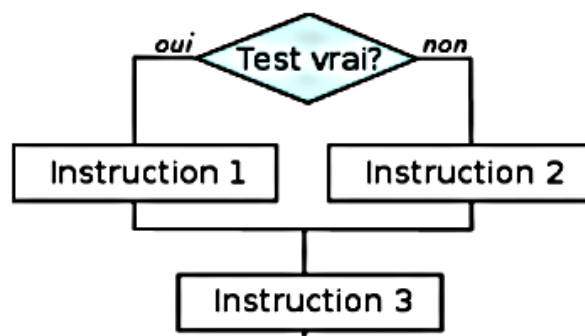
}

SINON { // Sinon exécuter

Instruction 2 exécutée

}

Instruction 3 (suite du programme)



Opérateurs de comparaisons:

`x == y` (x est égal à y)

`x != y` (x est différent à y)

`x < y` (x est inférieur à y)

`x > y` (x est supérieur à y)

`x <= y` (x est inférieur ou égal à y)

`x >= y` (x est supérieur ou égal à y)

Le moniteur série

Arduino possède un outil très pratique pour déboguer un programme (déboguer = trouver les erreurs ou contrôler un programme). Il s'agit d'une liaison série qui peut envoyer les informations que l'on veut et que l'on peut lire ensuite à l'aide du logiciel Arduino sur notre ordinateur.

Dans le **setup** il faut mettre:

`Serial.begin(9600);` // mise en route de la liaison série à 9600 bauds//

9600 correspond à la vitesse de communication. 9600 bauds est une vitesse rapide fiable

Dans le **void loop** :

Pour afficher le texte "le contenu de ma variable est : "

`Serial.print("le contenu de ma variable est : ");`

Pour afficher le contenu d'une variable (par exemple ici elle s'appelle var1)

`Serial.print(var1);`

Pour aller à la ligne (sinon chaque "Serial.print" affiche les données les unes à côté des autres) :

`Serial.println(var1);`