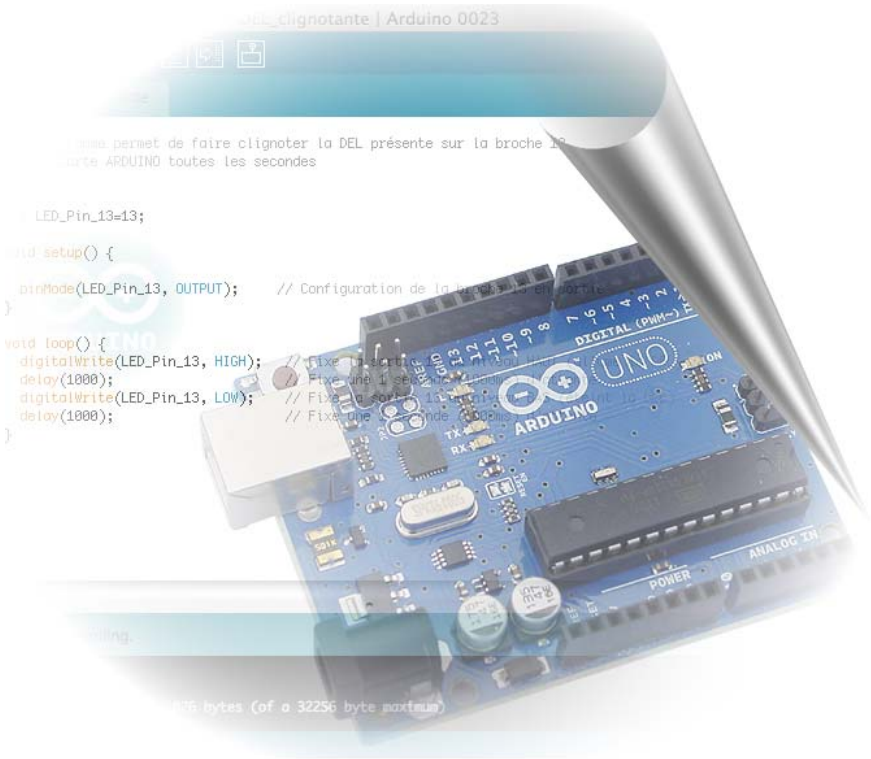
STI 2D

Systèmes d'Information et Numériques

Livret du débutant en environnement Arduino

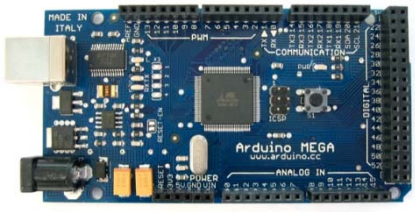


P@Eugé 2013-2014

1. Présentation des cartes ARDUINO

Le système **Arduino** est une carte électronique basée autour d'un microcontrôleur et de composants minimum pour réaliser des fonctions plus ou moins évoluées à bas coût. Elle possède une interface **USB** pour la programmer. C'est une plateforme **open-source** qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur.

Arduino Uno

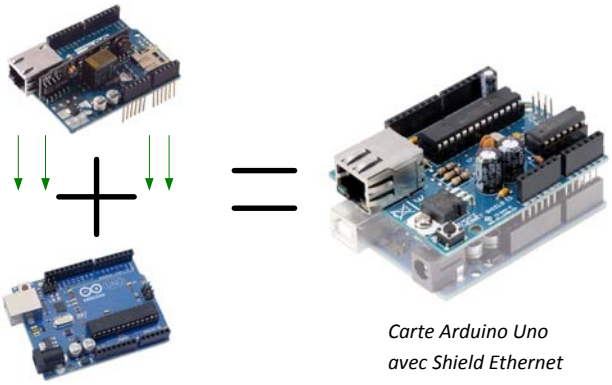


Arduino Mega2560

Arduino peut être utilisé pour développer des applications matérielles industrielles légères ou des objets interactifs (création artistiques par exemple), et peut recevoir en entrées une très grande variété de capteurs. Arduino peut aussi contrôler une grande variété d'actionneurs (lumières, moteurs ou toutes autres sorties matériels). Les projets Arduino peuvent être autonomes, ou communiquer avec des logiciels sur un ordinateur.

Les cartes électroniques peuvent être fabriquées manuellement ou bien être achetées pré assemblées ; le logiciel de développement **open-source** est téléchargeable gratuitement (http://arduino.cc/en/Main/Software) et permet de programmer **in-situ** la carte de développement via la liaison USB. De nombreuses bibliothèques ont été développées par des contributeurs au développement, et permettent de gérer l'environnement de ces cartes (Afficheur LCD, carte Ethernet, capteurs analogiques et numériques, moteurs CC, etc.)

La particularité des cartes de développement Arduino tient au fait de pouvoir ajouter d'autres modules compatibles avec la carte elle-même et l'environnement de développement Arduino. Ces modules, appelés "**Shield**" (bouclier en Anglais) sont ajoutés directement sur la carte par "*empilement*" :



Carte Arduino Uno
avec Shield Ethernet

Il est donc très facile de réaliser des modules de prototypage en empilant les différents modules dont on à besoin pour la réalisation du projet (http://www.lextronic.fr/R2545-arduino-version-officielle.html)

Module " Shield CAN-BUS "

Module "Shield GPS"

Module "Shield Wifi"

Module "Shield CMUcam4" *Reconnaissance vidéo*

Module "Shield GSM"

Module "motor shield Rev3"

2. Présentation de l'environnement de DEVELOPPEMENT Arduino

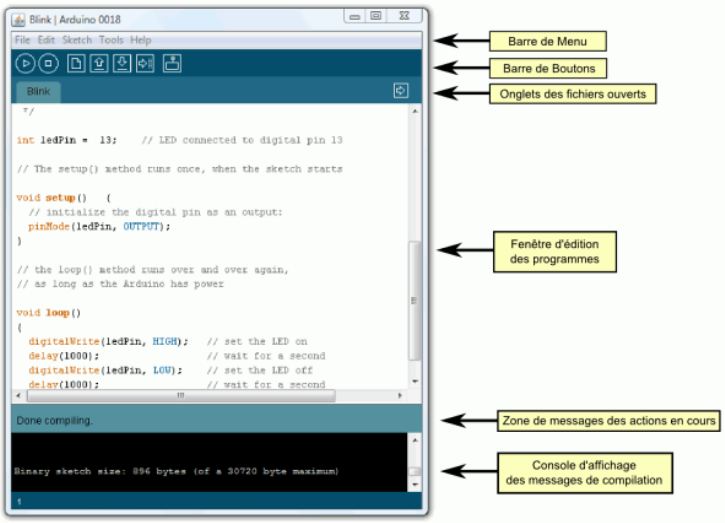
2.1 Description

Le logiciel Arduino a pour fonctions principales :

- de pouvoir **écrire** et **compiler** des programmes pour la carte Arduino
- de se **connecter** avec la carte Arduino pour y **transférer** les programmes
- de **communiquer** avec la carte Arduino

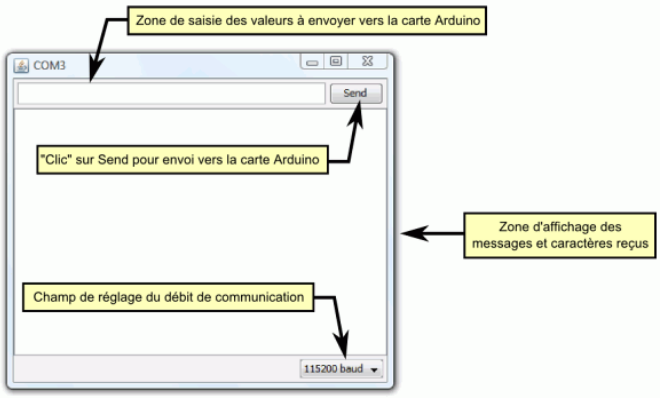
Cet espace de développement intégré (EDI) dédié au langage Arduino et à la programmation des cartes Arduino comporte :

- une **BARRE DE MENUS** comme pour tout logiciel une interface graphique (GUI),
- une **BARRE DE BOUTONS** qui donne un accès direct aux fonctions essentielles du logiciel et fait toute sa simplicité d'utilisation,
- un **EDITEUR** (à coloration syntaxique) pour écrire le code de vos programme, avec onglets de navigation,
- une **ZONE DE MESSAGES** qui affiche indique l'état des actions en cours,
- une **CONSOLE TEXTE** qui affiche les messages concernant le résultat de la compilation du programme



Le logiciel Arduino intègre également un **TERMINAL SERIE** (fenêtre séparée) qui permet d'afficher des messages textes reçus de la carte Arduino et d'envoyer des caractères vers la carte Arduino.

Cette fonctionnalité permet une mise au point facilitée des programmes, permettant d'afficher sur l'ordinateur l'état de variables, de résultats de calculs ou de conversions analogique-numérique : un élément essentiel pour améliorer, tester et corriger ses programmes.

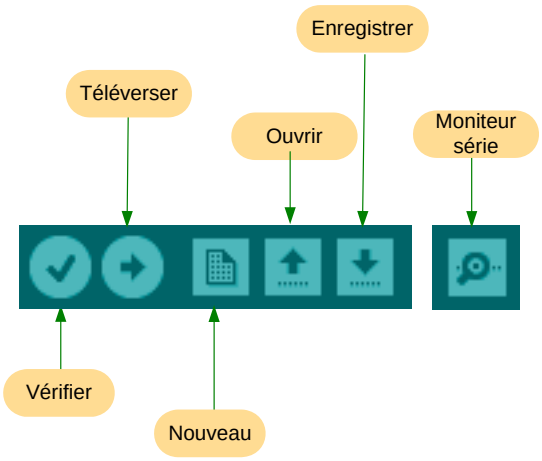


2.2 Principe général d'utilisation

Le code écrit avec le logiciel Arduino est appelé un programme (ou une séquence - sketch en anglais) :

- Ces programmes sont écrits dans **l'éditeur de texte**. Celui-ci a les fonctionnalités usuelles de copier/coller et de rechercher/remplacer le texte.
- la **zone de messages** donne l'état de l'opération en cours lors des sauvegardes, des exportation et affiche également les erreurs.
- La **console texte** affiche les messages produit par le logiciel Arduino incluant des messages d'erreur détaillés et autres informations utiles.
- la **barre de boutons** vous permet de vérifier la syntaxe et de transférer les programmes, créer, ouvrir et sauver votre code, et ouvrir le moniteur série.
- la **barre des menus** vous permet d'accéder à toutes les fonctionnalités du logiciel Arduino.

2.3 Description de la barre des boutons



Vérifier/compiler : Vérifie le code à la recherche d'erreur, puis compile le code si aucune erreur n'est détectée.

Nouveau : Crée un nouveau code (ouvre une fenêtre d'édition vide)

Ouvrir : Ouvre la liste de tous les programmes dans votre "livre de programmes". Cliquer sur l'un des programmes l'ouvre dans la fenêtre courante.

Sauver : Enregistre votre programme.

Transférer vers la carte : Compile votre code et le transfère vers la carte Arduino. Voir ci-dessous "Transférer les programmes" pour les détails.

Moniteur Série : Ouvre la fenêtre du moniteur (ou terminal) série.

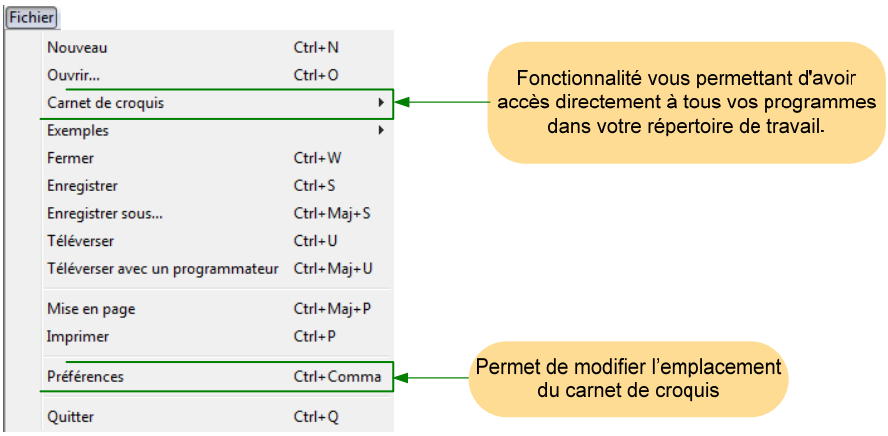
2.4 Description des menus

Des commandes complémentaires sont disponible dans cinq menus : Fichier Édition Croquis Outils Aide

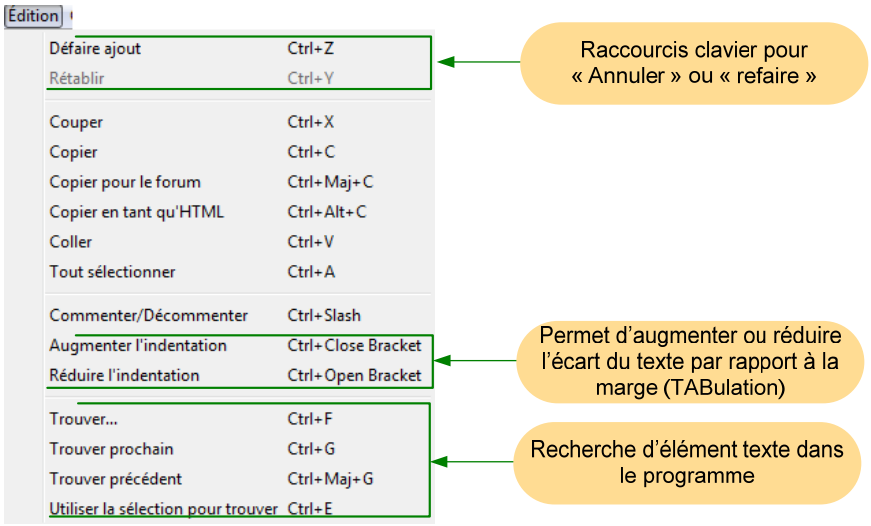
Le menu est sensible au contexte ce qui signifie que seulement les items correspondant au travail en cours sont disponibles.

Voici une liste restreinte des commandes très utiles au développement des programmes :

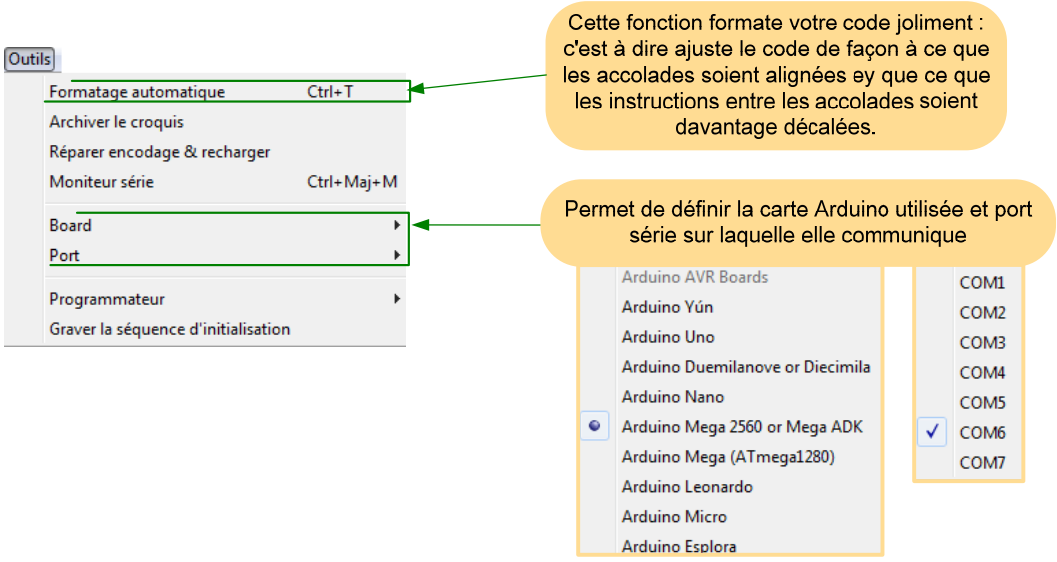
- Menu "**Fichier**" : Propose toutes les fonctionnalités usuelles pour gérer les fichiers et le transfert du programme



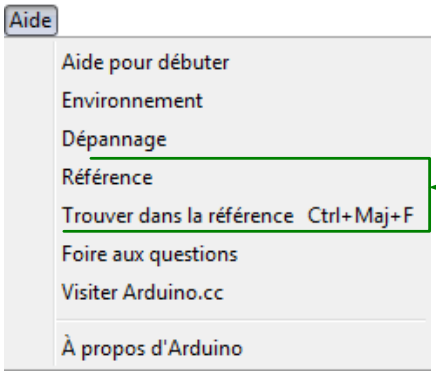
- Menu "**Edition**" : Propose toutes les fonctionnalités usuelles pour éditer le texte du programme en cours



- Menu "**Outils**" : Propose toutes les fonctionnalités usuelles pour définir le type de carte Arduino à programmer



- Menu "**Aide**" : Propose des liens internet pour vous apporter de l'aide sur les fonctions (en Anglais *of course!!*).



Propose tout ce qui peut être utile pour être aidé lors de l'écriture des programmes, notamment un lien vers la référence du langage en anglais.

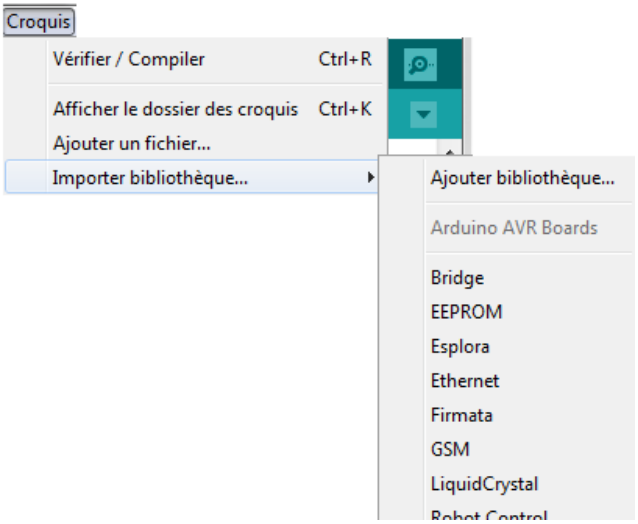
La connexion internet doit être activée !

2.5 Les Librairies

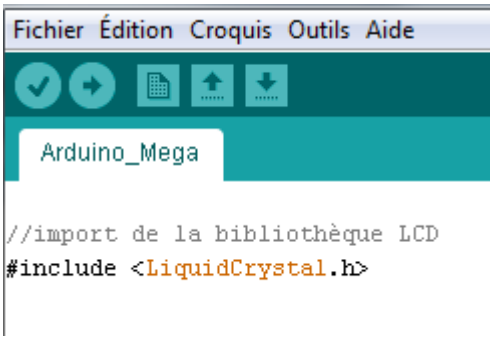
Tout comme java ou python, le logiciel de programmation Arduino utilise un ensemble de bibliothèque appelé **Librairie**. Les librairies fournissent des fonctions nouvelles que vous pouvez utiliser dans vos programmes, par exemple pour utiliser un matériel précis (un afficheur LCD, un module GPS, etc...) ou manipuler des données sans devoir écrire tout le code de programmation.

Pour utiliser une librairie, la sélectionner depuis le menu **Croquis** --> **Importer bibliothèque...**

Cela insèrera une ou plusieurs instructions **#include** au début de votre programme et compilera la librairie avec votre programme.



Exemple : utilisation de la librairie **LiquidCrystal** (afficheur LCD) :



En important cette librairie, vous accédez alors à toute une série de **fonctions** propres à celle-ci :

LiquidCrystal()	write()	noBlink()	autoscroll()
begin()	print()	display()	noAutoscroll()
clear()	cursor()	noDisplay()	leftToRight()
home()	noCursor()	scrollDisplayLeft()	rightToLeft()
setCursor()	blink()	scrollDisplayRight()	createChar()

3. Description de la structure d'un programme

3.1 Description générale des parties

Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne. La carte lit puis effectue les instructions les unes après les autres, dans l'ordre défini par les lignes de code, comme mors d'une programmation classique. Cette structure se décompose en trois ou quatre parties :

- Description des librairies utilisées, des constantes et variables du programme
- Fonction initialisation : configuration des entrées/sorties et éléments à configurer (cette partie ne sera exécutée qu'une seule fois) dans le partie **VOID SETUP( )**
- Le programme principal : **boucle** de fonctionnement général du programme (gestion des interactions entre les entrées/sorties) dans la partie **VOID LOOP( )**
- Les fonctions ou procédures : routines extérieures au programme principal, utilisées par ce dernier pour exécuter certaines commandes. Elles peuvent ne renvoyer aucune valeur (procédure) ou une valeur (fonction)

Exemple :

DEL_clignotante

```
/*
 Ce programme permet de faire clignoter la DEL présente sur la broche 13
 de la carte ARDUINO toutes les secondes
 */
```

Déclaration des variables

```
int LED_Pin_13=13;
```

Initialisation

```
void setup() {

  pinMode(LED_Pin_13, OUTPUT);     // Configuration de la broche 13 en sortie
}
```

Programme principal (s'exécute en boucle)

```
void loop() {
  digitalWrite(LED_Pin_13, HIGH);   // Fixe la sortie 13 au niveau HAUT (allume la DEL)
  delay(1000);                      // Fixe une 1 seconde (1000ms) d'attente
  digitalWrite(LED_Pin_13, LOW);    // Fixe la sortie 13 au niveau BAS (éteint la DEL)
  delay(1000);                      // Fixe une 1 seconde (1000ms) d'attente
}
```

3.2 Coloration syntaxique

Lorsque du code est écrit dans l'interface de programmation, certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

En **orange**, apparaissent les mots-clés reconnus par le langage Arduino comme des **fonctions** existantes. Lorsqu'on **sélectionne** un mot coloré en orange et qu'on effectue **un clic avec le bouton droit de la souris**, l'on a la possibilité de choisir « ***trouver dans la référence*** » : cette commande ouvre directement la documentation de la fonction sélectionnée.

En **bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des **constantes**.

En **gris**, apparaissent les **commentaires** qui ne seront **pas exécutés dans le programme**. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes. L'on peut déclarer un commentaire de deux manières différentes :

- dans une ligne de code, tout ce qui se trouve après « **//** » sera un commentaire.
- l'on peut encadrer des commentaires sur plusieurs lignes entre « **/*** » et « ***/** ».

3.3 La syntaxe du langage

Ponctuation

Le code est structuré par une ponctuation stricte :

- **toute ligne** de code se termine par un point-virgule « **;** »
- le contenu d'une **fonction** est délimité par des accolades « **{** » et « **}** »
- les **paramètres** d'une fonction sont contenus pas des parenthèses « **(** » et « **)** ».

Une erreur fréquente consiste à oublier un de ces éléments.

Les variables

Une variable est un espace réservé dans la mémoire de l'ordinateur. C'est comme un compartiment dont la taille n'est adéquate que pour un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement.

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont :

- nombres entiers (int)
- nombres à virgule flottante (float)
- texte (String)
- valeurs vrai/faux (boolean).

NOM DU TYPE	VALEUR MIN/MAX	TAILLE EN MEMOIRE
VALEURS BINAIRES		
boolean	0/1	1 octet
VALEURS NUMERIQUES ENTIERES SIGNEES		
int	-32 768 / +32 767	2 octets
long	-2 147 483 648 / +2 147 483 647	4 octets
VALEURS NUMERIQUES ENTIERES NON SIGNEES		
byte	0 / +255	1 octet
unsigned int	0 / +65535	2 octets
word	0 / +65535	2 octets
unsigned long	0/ +4 294 967 295	4 octets
VALEURS NUMERIQUES A VIRGULE		
float	-3.4028235E+38 / +3.4028235E+38	4 octets
double	-3.4028235E+38 / +3.4028235E+38	4 octets
CARACTERES		
char	-128 / +127 (ASCII)	1 octet
Chaine de caractères		
String	Classe d'objet permettant de manipuler plus facilement les chaines qu'un tableau de caractère (character arrays). Cette classe possède de nombreuses méthodes (fonctions) : String(), charAt() , compareTo() , concat() , endsWith() , equals() , equalsIgnoreCase() , getBytes() , indexOf() , lastIndexOf() , length() , replace() , setCharAt() , startsWith() , substring() , toCharArray() , toLowerCase() , toUpperCase() , trim()	

Exemple :

```
int         mavariable = 45;
 // int est le type,  mavariable le nom,  = 45 assigne une valeur.
```

Les Constantes

Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme. On la déclare de la façon suivante :

```
CONST   TYPE_DE_LA_DONNEE   NOM_DE_LA_DONNEE
```

Exemple :

```
// déclaration de la constante pi
const   float   pi = 3.1415926;
```

Lorsque l'on désire utiliser une Entrée/sortie de la carte Arduino, il est préférable de la définir comme une constante associée à une broche de la carte :

```
// déclaration des étiquettes de brochage (LED_pin sur broche 7 et BP_pin sur broche 53)
const   int   LED_pin  =  7;
const   int   BP_pin  = 53;
```

Les procédures et fonctions

Une fonction (également désignée sous le nom de procédure ou de sous-routine lorsqu'elle ne renvoie aucune valeur) est un bloc d'instructions que l'on peut appeler à tout endroit du programme.

Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple analogRead(), digitalWrite() ou delay().
Il est possible de déclarer ses propres fonctions ou procédures, par exemple une procédure pour réaliser un clignotement :

```
void clignote(){
  digitalWrite (brocheLED,  HIGH) ;
  delay (1000) ;
  digitalWrite (brocheLED,  LOW) ;
  delay (1000) ;
}
```

Pour exécuter cette fonction, il suffit de taper la commande :

```
clignote();
```

On peut faire intervenir un ou des **paramètres** dans une fonction ou procédure :

```
void clignote( int broche, int vitesse ){
  digitalWrite (broche, HIGH) ;
  delay (1000/vitesse) ;
  digitalWrite (broche, LOW) ;
  delay (1000/vitesse) ;
}
```

Dans ce cas, l'on peut moduler leurs valeurs depuis la commande qui l'appelle :

```
clignote(5,1000);     //la sortie 5 clignotera vite
clignote(3,250);      //la sortie 3 clignotera lentement
```

Les structures de contrôle

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction du respect d'un certain nombre de conditions. Il existe quatre types de structure :

if...else : exécute un code **si** certaines conditions sont remplies et éventuellement exécutera un autre code avec **sinon.**

exemple :

```
//si la valeur du capteur dépasse le seuil
if( valeurCapteur > seuil ){
   //appel de la fonction clignote
   clignote();
}
```

while : exécute un code **tant** que certaines conditions sont remplies.

exemple :

```
//tant que la valeur du capteur est supérieure à 250
while( valeurCapteur > 250 ){
   //allume la sortie 5
   digitalWrite(5,HIGH);
   //envoi le message "0" au port série
   Serial.println(1);
   //en boucle tant que valeurCapteur est supérieure à 250
}
Serial.println(0);
digitalWrite(5,LOW);
```

for : exécute un code **pour** un certain nombre de fois.

exemple :

```
//pour i de 0 à 255, par pas de 1
for (int i=0; i <= 255; i++){
   analogWrite(PWMpin, i);
   delay(10);
}
```

switch/case : fait un choix entre plusieurs codes **parmi une liste de possibilités**

exemple :

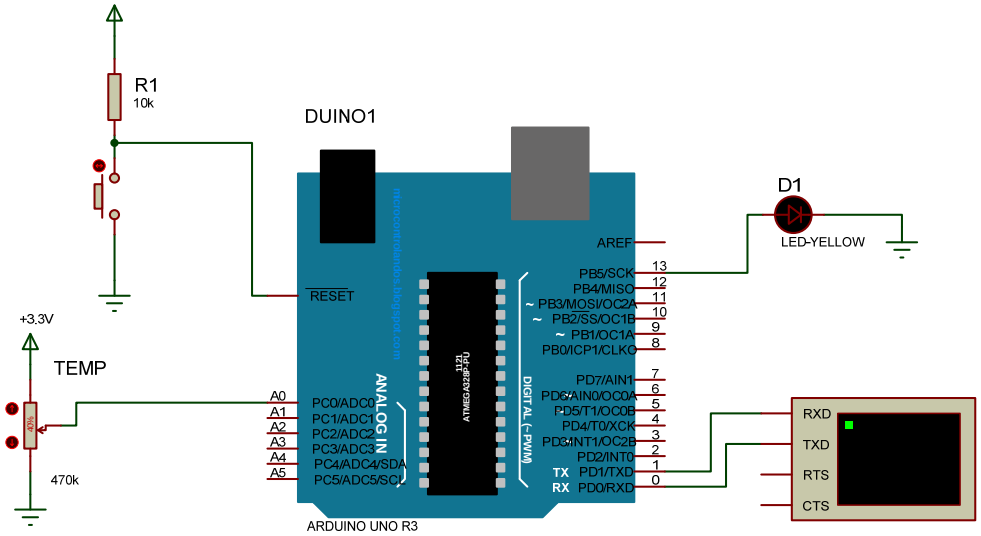
```
// fait un choix parmi plusieurs messages reçus
switch (message) {

  case 0: //si le message est "0"
    //allume que la sortie 3
    digitalWrite(3,HIGH);
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
    break;

  case 1: //si le message est "1"
    //allume que la sortie 4
    digitalWrite(3,HIGH);
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
    break;

  case 2: //si le message est "2"
    //allume que la sortie 5
    digitalWrite(3,LOW);
    digitalWrite(4,LOW);
    digitalWrite(5,HIGH);
    break;
  }
```

<u>Exemple d'application</u> : Détection de seuil

L'exemple qui suit montre l'utilisation de quelques éléments de la syntaxe du langage Arduino. Dans ce programme, un signal analogique provenant d'un capteur (potentiomètre) fait varier la vitesse de clignotement d'une LED, à partir d'un certain seuil.

Le principe de fonctionnement est le suivant :

1. Le signal est lu avec la fonction « analogRead () ».
2. La valeur du signal varie en 0 et 1023.
3. Si la valeur dépasse le seuil fixé par la variable SEUIL alors la LED est allumée et éteinte pendant un délai correspondant à la valeur du signal reçu du capteur.



Le programme est le suivant :

```
/* Dans ce programme, un signal analogique provenant d'un capteur (potentiomètre)
fait varier la vitesse de clignotement d'une LED, à partir d'un certain seuil. La donnée convertie est transmise en série
*/

//Partie 1 ---------- déclaration des variables ----------
int brocheCapteur = A0;          // sélection de la broche A0 sur laquelle est connectée le capteur
int brocheLED = 13;              // sélection de la broche 13 sur laquelle est connectée la LED

int valeurCapteur = 0;           // variable stockant la valeur du signal reçu du capteur
const int SEUIL= 200;            //seuil de déclenchement - ne peut être modifié (constante)


//Partie 2 ---------- initialisation ----------
void setup () {
   Serial.begin(57600);                          // Fixe la vitesse de transfert série à 57600 bauds
   pinMode (brocheCapteur, INPUT) ;              // broche du capteur configurée en entrée
   pinMode (brocheLED, OUTPUT) ;                 // broche de la LED configurée en sortie
}


//Partie 3 ----------   boucle principale ----------
void loop () {
   valeurCapteur = analogRead (brocheCapteur) ;      // conversion AN du signal du capteur et lecture de N

  Serial.println(valeurCapteur);                     //Transfert en série la valeur de conversion N avec RC

   if (valeurCapteur > SEUIL){                       //Test la condition de déclenchement (> SEUIL)
      clignote();                                    //Si vraie alors appel de la procédure clignote()
   }
}


//Partie 4  --------- Procédure de clignotement ----------
void clignote(){
   digitalWrite (brocheLED, HIGH) ;          // allume la LED
   delay (valeurCapteur) ;                   // temporise de «valeurCapteur" millisecondes
   digitalWrite (brocheLED, LOW) ;           // éteint la LED
   delay (valeurCapteur) ;                   // temporise de «valeurCapteur" millisecondes
}
```

A vos souris... prêt ..... PARTEZ !

Lancez Arduino, puis copiez ce programme. Compilez votre programme puis chargez le fichier .HEX dans le module Arduino Uno sous le logiciel de simulation Proteus ISIS. Testez alors votre programme...

4. Pour aller plus loin

La liste exhaustive des éléments de la syntaxe du langage Arduino est consultable sur le site :

http://arduino.cc/fr/Main/Reference

De nombreux sites et forum permettent de trouver des solutions de programmation, de la gestion des ports d'Entrées/Sorties, à la gestion du bus I2C, en passant par la gestion des interruptions.

Pour programmer en Arduino, il faut appliquer les 4 articles du SIN :

article 1 : Google est ton ami
article 2 : il n'y a pas de problème, mais que des solutions
article 3 : on ne récolte que ce que l'on sème
article 22 : .... voir article 1

Enfin quelques sources utiles :

SUR INTERNET:

http://www.arduino.cc/
http://www.arduino.cc/en/Reference/Libraries
http://www.mon-club-elec.fr/
http://fr.wikipedia.org/wiki/Arduino
http://www.lextronic.fr/R2386-modules-arduino.html
http://fr.openclassrooms.com/sciences/cours/arduino-pour-bien-commencer-en-electronique-et-en-programmation
http://fr.flossmanuals.net/arduino/index

et un tas d'autre .....

BIBLIOGRAPHIE:

Christian Tavernier, *Arduino Maîtriser sa programmation et ses cartes d'interface (shields)*, Paris, 2011, DUNOD
François Auger , *Présentation de l'environnement ARDUINO*