Historique

C'est dans un bar d'une petite ville du nord de l'Italie qu'est né le projet **Arduino** qui, de manière totalement inattendue, est en train de révolutionner le domaine de l'électronique à l'échelle mondiale, puisque pour la première fois tout le monde peut vraiment s'y essayer et découvrir qu'il aime ça ! Rien de tout ceci n'aurait été possible sans le choix initial des licences libres qui a conditionné non seulement son bas prix et sa massive diffusion mais également son approche et son état d'esprit.

Hiver 2005, Massimo Banzi enseigne dans une école de Design à Ivrea en Italie, et souvent ses étudiants se plaignent de ne pas avoir accès à des solutions bas prix pour accomplir leurs projets de robotique. Banzi en discute avec David Cuartielles, un ingénieur Espagnol spécialisé sur les micro-contrôleurs…

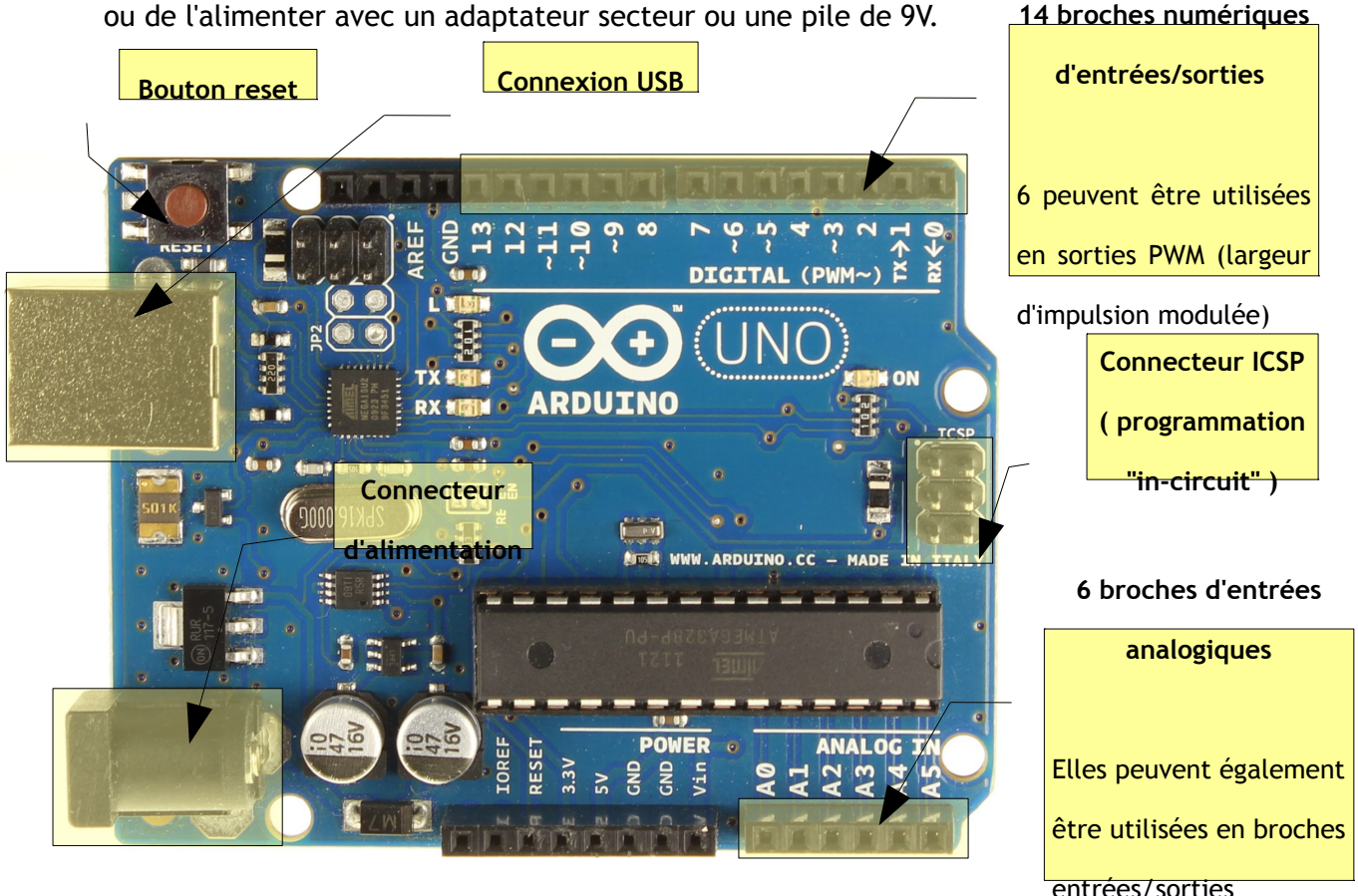
Ils décident de créer leur propre carte en embarquant dans leur histoire un des étudiants de Banzi, David Mellis, qui sera chargé de créer le langage de programmation allant avec la carte. En deux jours David écrira le code !

Trois jours de plus et la carte était crée…

La carte Uno R3

La carte **Arduino Uno R3** est une carte à microcontrôleur basée sur l'Atmega328, c'est la dernière d'une série de carte USB Arduino. C'est le modèle de référence des plateformes Arduino.

Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur. Pour pouvoir l'utiliser et se lancer, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble USB ou de l'alimenter avec un adaptateur secteur ou une pile de 9V.



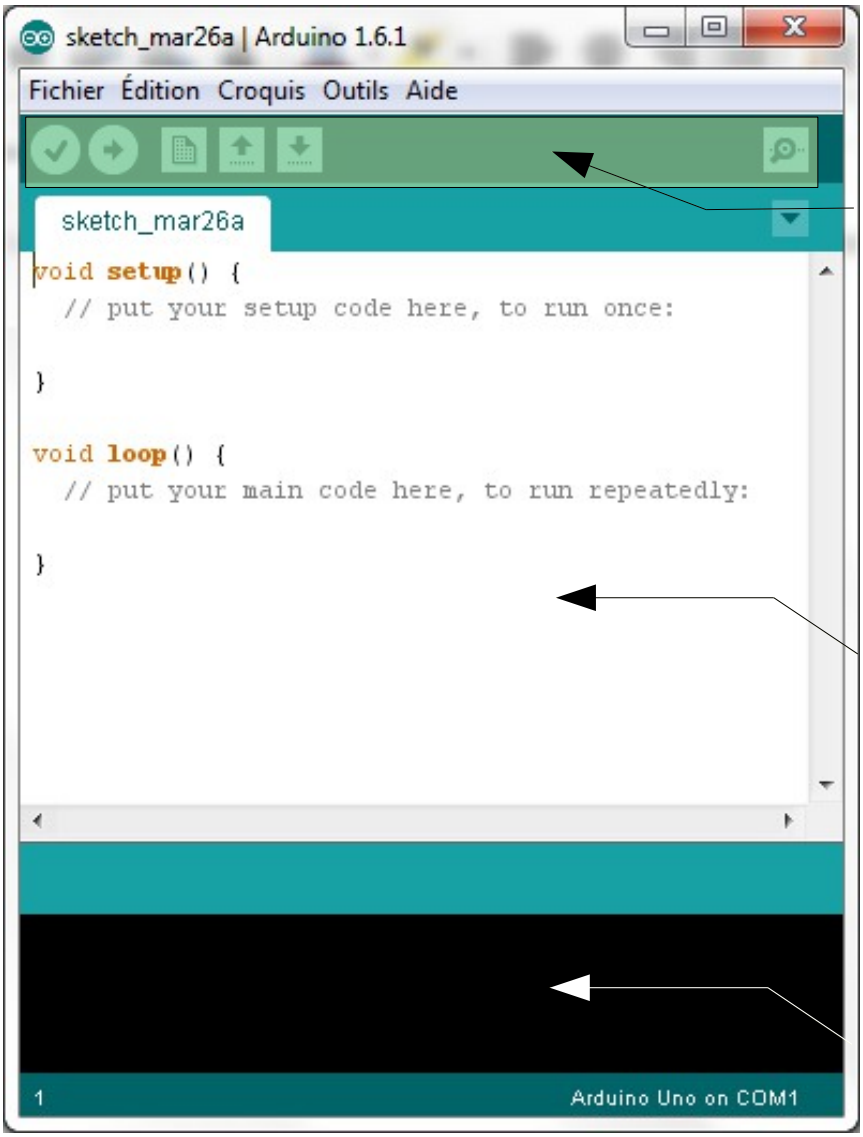
Il existe de nombreuses cartes compatibles ou équivalentes à la carte Uno R3. Nous utiliserons des cartes Funduino qui ont l'avantage de ne coûter qu'une quinzaine d'euros.

L'environnement de développement

L'environnement de développement peut-être téléchargé à l'adresse suivante :

http://arduino.cc/en/Main/Software

Une fois lancé, l'ED s'affiche dans une simple fenêtre et reprend le dernier programme (on parle de croquis dans le jargon arduino) créé ou à défaut en propose un nouveau. En dehors des traditionnels menus on trouve une barre d'outils permettant d'acceder aux fonctions principales.



La barre d'outils

(de gauche à droite)

Vérifier (s'il y a des erreurs)

Téléverser le croquis sur la carte

Nouveau croquis

Ouvrir un croquis existant

Enregistrer le croquis

La zone d'édition de croquis

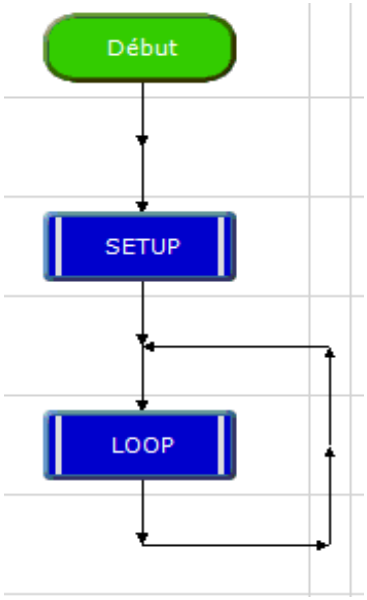
C'est l'éditeur de programme proprement dit, les deux fonctions **setup()** et **loop()** y sont par défaut

La zone de dialogue

C'est dans cette zone que l'environnement de développement affiche ses messages : messages de compilation, erreurs, transfert ...

La structure d'un croquis

Un croquis (sketch en anglais) comporte au minimum deux fonctions :



- La fonction **setup()** (configuration en anglais) qui est appelée au démarrage du programme.

  Cette fonction est utilisée pour initialiser les variables, le sens des broches, les librairies utilisées. La fonction setup n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino.

- La fonction **loop ()** (boucle en anglais) qui fait exactement ce que son nom suggère et s'exécute en boucle sans fin, permettant à votre programme de s'exécuter et de répondre.

  On utilise cette fonction pour contrôler activement la carte Arduino. Elle est exécutée immédiatement après la fonction **setup().**

Quelques éléments de langage

Le langage Arduino est basé sur le langage C défini par Kenneth Thompson & Dennis Ritchie en 1972. Comme Lui c'est un langage impératif structuré adapté à la programmation système. De nombreux autres langages de programmation (C++, Java,...) sont issus du langage C.

On se rapelera qu'un programme est une suite d'instructions logiquement ordonnées, et qu'une instruction est une action élémentaire qui peut être réalisée par une machine.

1. Syntaxe de base

; (point virgule) : la fin d'une instruction est toujours délimité par un point virgule

```
a = 13; // le point virgule indique la fin de l'instruction
```

{} (Accolades) : Bloc de contrôle

```
if (boolean expression){
   a = 13;      // ouverture du code de la condition if
   b = a+1;     // vos instructions (terminées par ";"  )
}               // fermeture du code de la condition if
```

2. Opérateurs arithmétiques

= (signe égal unique) : l'opérateur d'assignement (appelé aussi d'affectation) permet d'assigner à une variable une valeur ou le résultat d'une expression.

```
varA = 13; // on assigne à la variable  varA  la valeur 13
varB = varA+5; // on assigne à la variable  varB  le résultat
               // de l'addition du contenu de la variable varA
               // et de 5
```

+ (Addition), - (Soustraction), * (Multiplication), / (Division)

Ces opérateurs renvoient respectivement la somme, la différence, le produit ou le quotient entre deux opérandes (= entre deux termes). Cette opération est réalisée en utilisant le type des données des opérandes.

Ainsi par exemple, **9 / 4** donne 2 pour résultat si 9 et 4 sont de type int (entier).

3. Les opérateurs de comparaison

x == y	(x est égal à y)
x != y	(x est différent de 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)

Attention à ne pas utiliser accidentellement le signe '=' (égal unique) à la place de '=='

Ex : if (**varX** = 10)

Le signe égal unique est l'opérateur d'attribution d'une valeur. Dans l'exemple, il fixe la valeur de la variable **varX** à 10 (autrement dit, on met la valeur 10 dans la variable **varX**).

Utilisez bien au lieu de cela le signe double égal == (c'est à dire if (**varX** == 10), le == étant l'opérateur logique de comparaison, et qui test si **varX** est bien égal à 10 ou non. Cette dernière condition (**varX**==10) est vraie uniquement si **varX** est égal à 10, alors que la première condition (**varX**=10) sera toujours vraie puisque **varX** est différent de 0 (zéro).

4. Les constantes

*Quelques constantes sont définies dans le langage Arduino. Notez que les constantes **true** et **false** sont écrites en minuscules à la différence des constantes **HIGH**, **LOW**, **INPUT** et **OUTPUT**.*

True/False (définition des valeurs logiques)

Il existe deux constantes utilisées pour représenté le VRAI et le FAUX dans le langage arduino : true et false.

false (FAUX) : la constante **false** est définie comme le 0 (zéro).

true (VRAI) : La constante true peut être toute valeur différente de 0 : en logique ce qui n'est pas faux est vrai !

HIGH/LOW (définition des niveaux de broche)

Lorsqu'on lit ou on écrit sur une broche numérique, seuls deux états distincts sont possibles HIGH (HAUT) ou LOW (BAS). On pourrait faire une analogie avec un interrupteur qui serait respectivement fermé ou ouvert :

HIGH : la broche es t active

LOW : la broche est inactive

INPUT/OUTPUT (mode d'utilisation des broches numériques)

Les broches numériques peuvent être utilisées soit en entrée (INPUT), soit en sortie (OUTPUT). Pour configurer le mode de fonctionnement des broches numériques, on utilise la fonction **pinMode**(**broche**, **mode**) où l'on précisera le numéro de la **broche** ainsi que son **mode** d'utilisation.

Ex : `pinMode(13, OUTPUT);`

Dans cet exemple on configure la broche numérique 13 en sortie ; on y connectera un effecteur.

5. Les fonctions

Les entrées/sorties numériques

pinMode(broche, mode) : Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie (cf INPUT/OUTPUT)

digitalWrite(broche**,** valeur) : Met un niveau logique HIGH (HAUT en anglais) ou LOW (BAS en anglais) sur une broche numérique. Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur correspondante : 5V (ou 3.3V sur les cartes Arduino 3.3V) pour le niveau HAUT, 0V (masse) pour le niveau BAS.

digitalRead(broche) : Lit l''état (le niveau logique) d'une broche précise en entrée numérique, et renvoie la valeur HIGH ou LOW.

Temps

delay(ms) : Réalise une pause dans l'exécution du programme pour la durée (en millisecondes) indiquée en paramètre. (Pour mémoire, il y a 1000 millisecondes dans une seconde...!

Blink : le premier programme

Le programme ci dessous est généralement le premier programme qu'on téléverse dans la carte Arduino. Il permet entre autre de vérifier que la carte répond bien et que l'environnement de programmation est bien configuré.

Son seul objectif est de faire clignoter la LED soudée sur le circuit imprimé qui est raccordée à la broche numérique 13 de la carte (pour la carte Uno ou pour la carte Leonardo notamment).

Les commentaires sont assez nombreux et explicites.