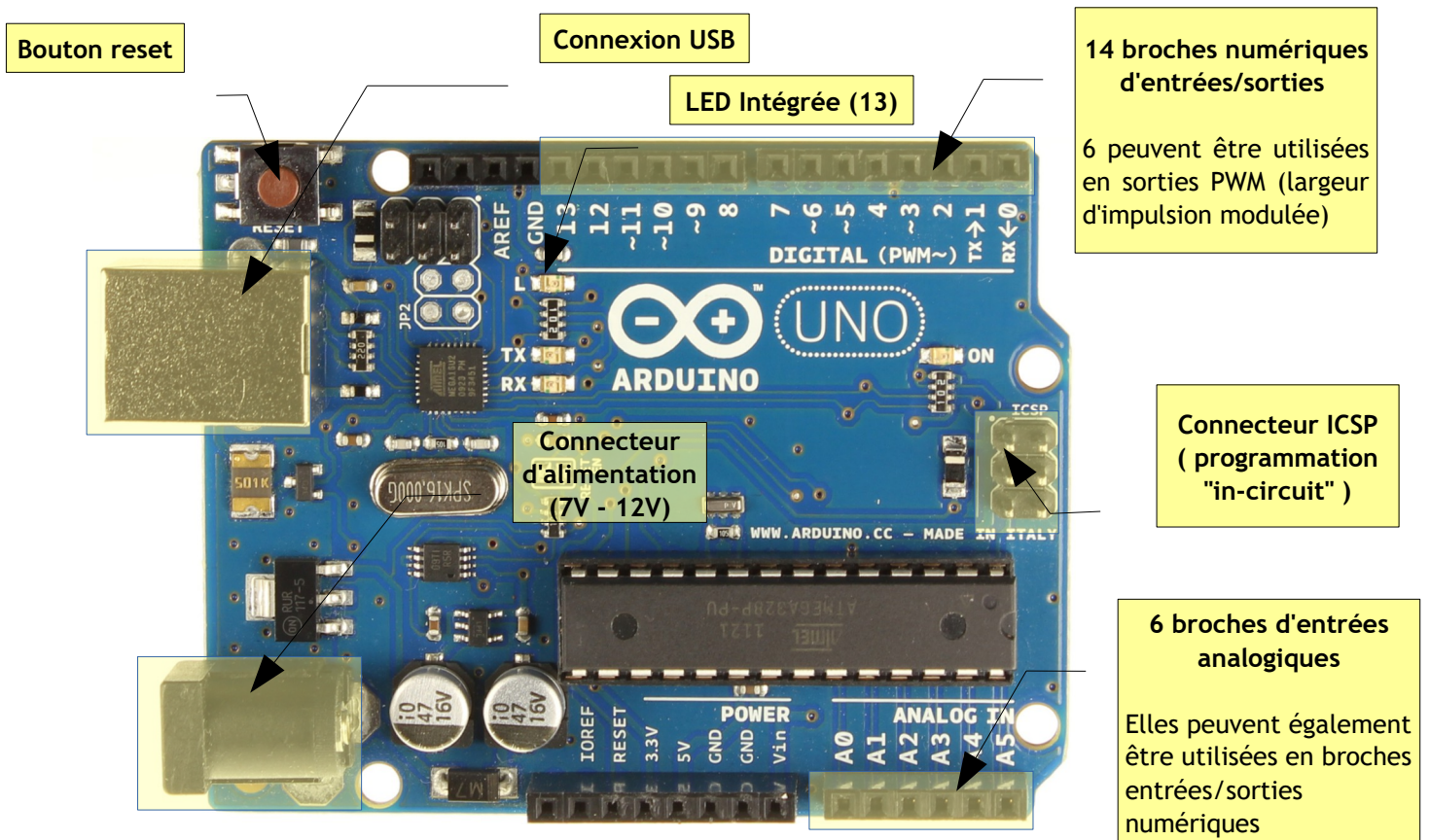


Dans ce document nous allons découvrir la carte **Arduino R3**, son environnement de programmation ainsi que les éléments de base de son langage de programmation.

Anatomie de la carte Uno R3

La carte **Arduino Uno R3** est une carte à microcontrôleur basée sur l'Atmega328. C'est le modèle de référence des plateformes Arduino. Ces dimensions (68,6 x 53,3 mm) lui permettent d'être utilisée dans de nombreux projets.

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.



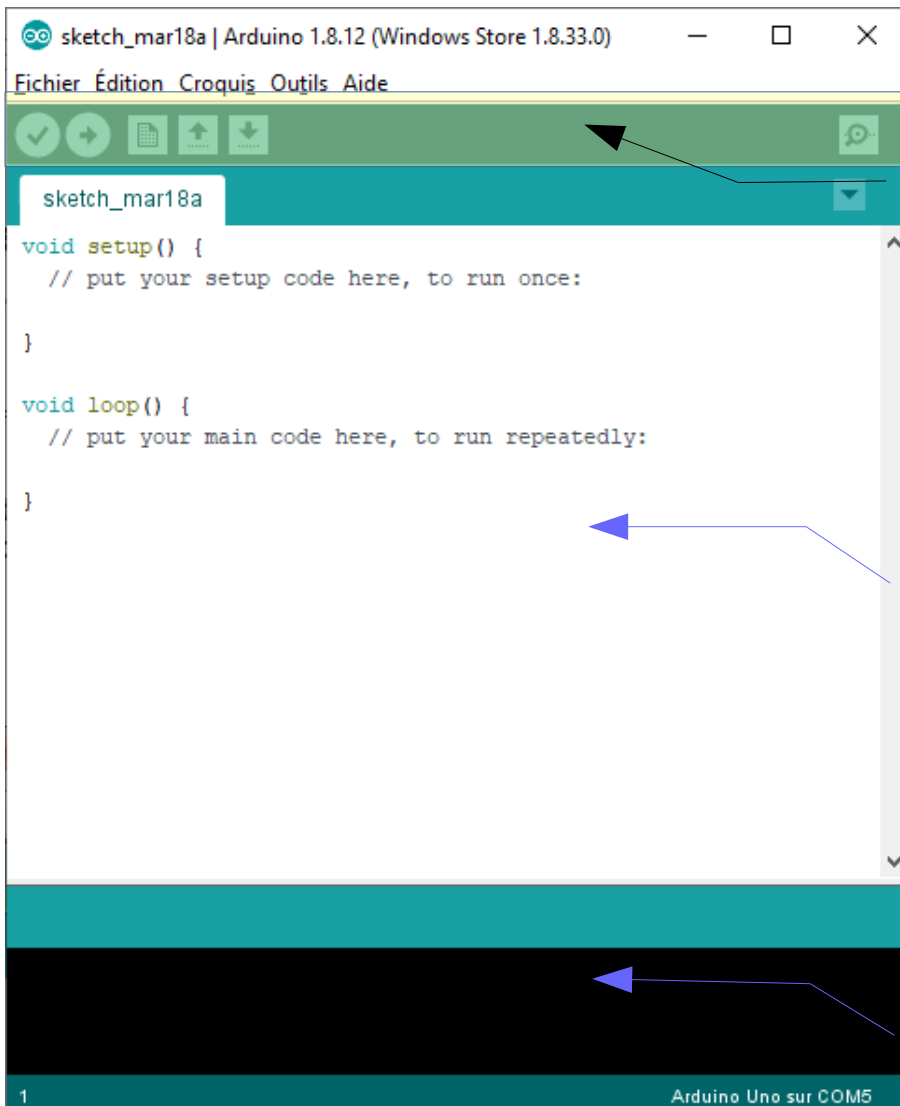
Il existe de nombreuses cartes compatibles ou équivalentes à la carte Uno R3. On peut ainsi trouver sur le web des cartes à des prix très compétitifs (env. 3€50).

L'environnement de développement

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

<https://www.arduino.cc/en/Main/Software#>

Une fois lancé, l'Environnement de Développement 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'accéder 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
- Ouvrir la fenêtre de terminal

La zone d'édition de croquis

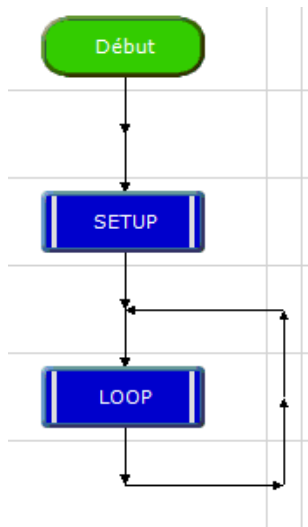
C'est l'éditeur de programme proprement dit, les deux fonctions `setup()` et `loop()` y sont par défaut

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 bibliothèques 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 rappellera qu'un programme est une suite d'instructions logiquement ordonnées, et qu'une instruction est une action élémentaire qui peut être réalisée par la machine.

. 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 d'instructions

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

// (double slash) : Commentaire de fin de ligne

```
// Ceci est un commentaire de fin de ligne
```

. 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 opérands (= entre deux termes). Cette opération est réalisée en utilisant le type des données des opérands.

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

. 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)

Ne pas utiliser accidentellement le signe '=' (égal unique) à la place de '==' (double égal)

Ex : if (varX = 10)

Le signe égal unique est l'opérateur d'affectation (attribution d'une valeur).

Dans cet exemple, il fixe la valeur de la variable varX à 10. Autrement dit, on met la valeur 10 dans la variable varX. La valeur testée est celle de varX et donc la condition est vraie puisque varX est non nulle.

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).

. 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 représenter 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 est 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.

```
pinMode(13, OUTPUT);
```

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

. Les fonctions de base

- **Attente**

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...!)

```
delay(250); // attend pendant 250 ms
```

- **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**)

```
pinMode(13, OUTPUT); // déclare en la broche 13 en sortie
```

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.

```
digitalWrite(13, HIGH); // active la sortie 13
digitalWrite(13, LOW);  // désactive la sortie 13
```

digitalRead(broche) : Retourne l'état (le niveau logique) de la broche numérique précisée en paramètre, et renvoie la valeur **HIGH** ou la valeur **LOW**. Cette fonction s'utilise avec une variable ou dans une condition

```
varD13=digitalRead(13); // affecte à la variable varD13 le
                        // résultat lecture de l'état
```

Si une tension supérieure à 3,5V est présente sur la broche 13 le résultat sera **HIGH**

• Entrées analogiques

analogRead(broche) : Retourne la valeur lue sur une *broche* analogique (A0 à A5) et la code sur 10 bits. La tension comprise entre 0 et 5V sera codée par une valeur de l'intervalle [0;1023]

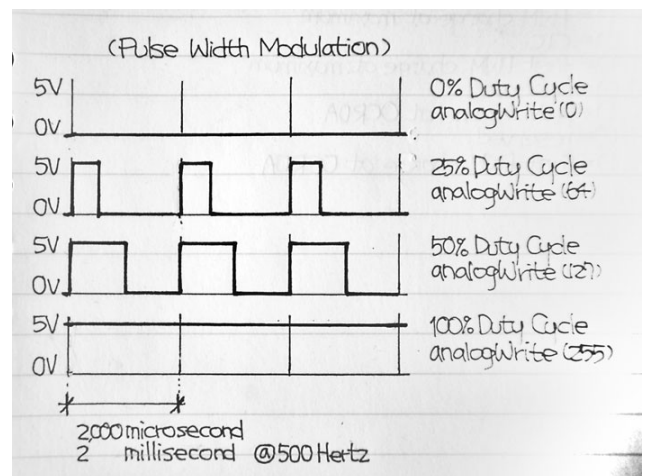
```
varA0=digitalRead(A0); // affecte à la variable varA0 le
                       // résultat la lecture de l'entrée A0
```

Si une tension de 3V est présente sur la broche A0 le résultat sera de $614 = 3 * \frac{5}{1024}$

• Sorties PWM

analogWrite(broche, valeur sortie) : Génère sur une *broche* [3,5,6,9,10,11] un signal pseudo-analogique par modulation de largeur d'impulsion -PWM-. *Valeur de sortie* est une valeur de l'intervalle [0;255] et correspond à la proportion de temps de l'intervalle pendant laquelle la broche est active :

- 0 --> 0%
- 63 --> 25%
- 127 --> 50%
- 255 --> 100%

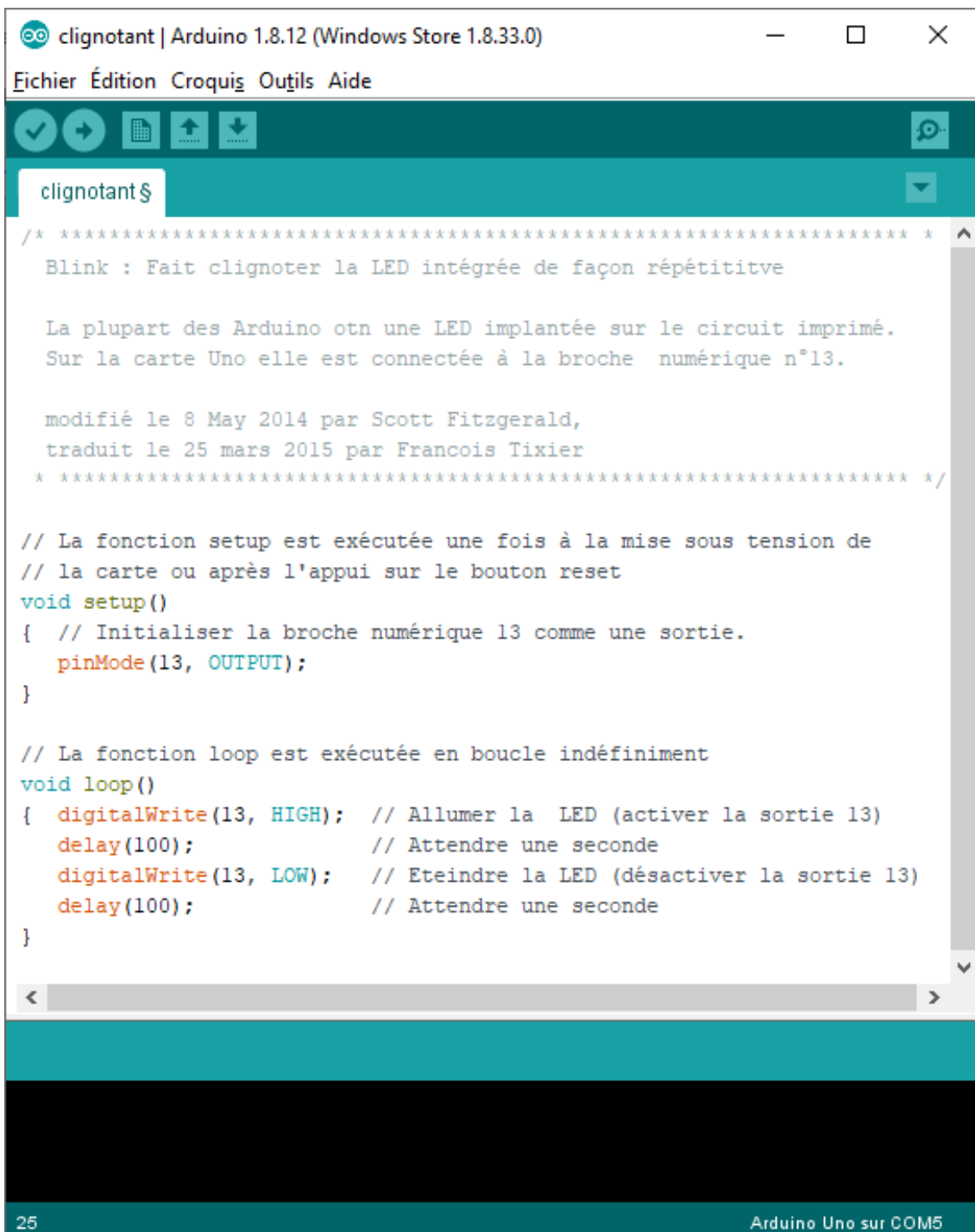


```
analogWrite(3, 127); // fixe le rapport cyclique de la sortie 3 à 50 %
```

Blink : le premier programme

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

Dans le cas présent, 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.



```
clignotant | Arduino 1.8.12 (Windows Store 1.8.33.0)
Fichier Édition Croquis Outils Aide
clignotant$
/* *****
Blink : Fait clignoter la LED intégrée de façon répétitive

La plupart des Arduino ont une LED implantée sur le circuit imprimé.
Sur la carte Uno elle est connectée à la broche numérique n°13.

modifié le 8 May 2014 par Scott Fitzgerald,
traduit le 25 mars 2015 par Francois Tixier
* ***** */

// La fonction setup est exécutée une fois à la mise sous tension de
// la carte ou après l'appui sur le bouton reset
void setup()
{ // Initialiser la broche numérique 13 comme une sortie.
  pinMode(13, OUTPUT);
}

// La fonction loop est exécutée en boucle indéfiniment
void loop()
{ digitalWrite(13, HIGH); // Allumer la LED (activer la sortie 13)
  delay(100); // Attendre une seconde
  digitalWrite(13, LOW); // Eteindre la LED (désactiver la sortie 13)
  delay(100); // Attendre une seconde
}

25 Arduino Uno sur COM5
```