2. microcontrôleur ARDUINO

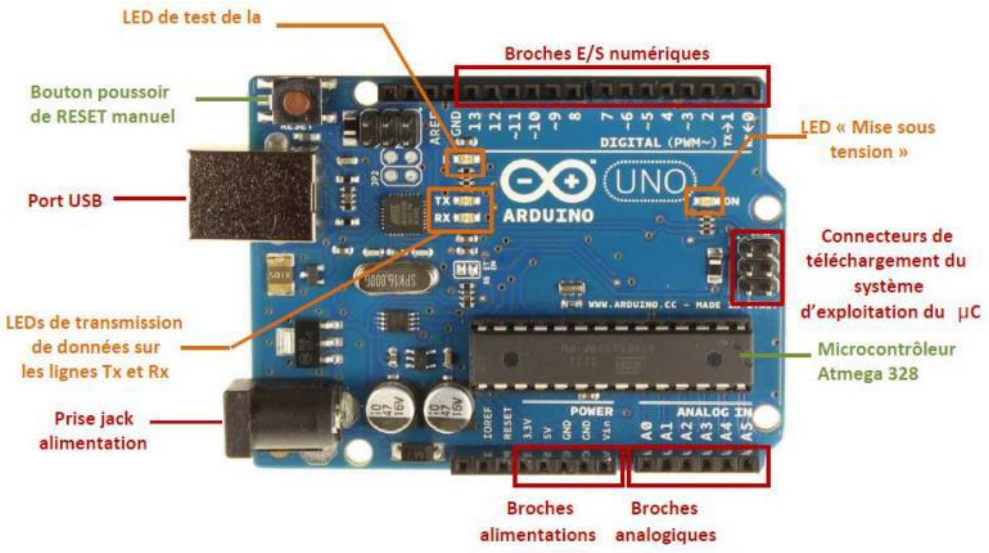
I) Présentation :

C'est une petite carte programmable, c'est à dire qu'elle peut apprendre à effectuer des tâches en fonction d'un programme écrit d'avance. Elle est au format carte bancaire.

Les réalisations autour d'Arduino peuvent être très simples : un jeu de lumière, un télémètre à sonar ou un instrument de musique ne nécessitent aucune compétence technique mais demandent un travail sur l'algorithmique et les langages de programmation.

1. La carte microcontrôleur Arduino :

Le modèle UNO de la société ARDUINO est une carte électronique dont le cœur est un microcontrôleur ATMEL de référence ATMega328. Le microcontrôleur ATMega328 est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C.
L
a force de l'Arduino est de nous proposer le microcontrôleur, **les entrées/sorties, la connectique et l'alimentation sur une seule carte**. La carte Arduino est construite autour d'un microcontrôleur Atmel AVR (pas toujours le même en fonction de la date de sortie de la carte) avec une **capacité de mémoire de 32000 octets** pour l'Arduino UNO. Soit 32 Ko,



Microcontrôleur	ATMEGA 328
Tension d'alimentation interne	5V
Tension d'alimentation (recommandée)	7 à 12V
Tension d'alimentation (limites)	6 à 20 V
Entrées/sorties numériques	14 dont 6 sorties PWM (configurables)
Entrées analogiques	6
Courant max par broches E/S	40 mA (200mA cumulé pour l'ensemble des broches)
Mémoire programme Flash	32 Ko dont 0,5 Ko sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	2 KB
Mémoire EEPROM (mémoire non volatile)	1 KB
Fréquence horloge	16 MHz
Dimensions	68.6mm x 53.3mm

La carte ARDUINO UNO peut être alimentée par le **câble USB** ou par un bloc secteur externe connecté grâce à une prise « jack » de 2,1mm ou bien par un bloc de piles.

L'alimentation extérieure doit présenter une tension comprise entre 7 à 12V.

2. La carte Arduino les entrées/sorties :

La carte « ARDUINO UNO » dispose de **14 Entrées/Sorties numériques (DIGITAL)** et de 6 **entrées analogiques (ANALOG IN)**.

Chacune des 14 broches numériques (repérées 0 à 13) peut être utilisée en entrée (input) ou en sortie (output) sous le contrôle du programme. Le sens de fonctionnement pouvant même changer de manière dynamique pendant son exécution. Elles fonctionnent en logique TTL (0V-5V ou 0-1 binaire)

II) La programmation avec Arduino :

Le logiciel Arduino a pour fonction principale de :

- Ecrire et compiler des programmes pour la carte Arduino
- Se connecter à la carte Arduino via un PC pour y transférer le programme réalisé.

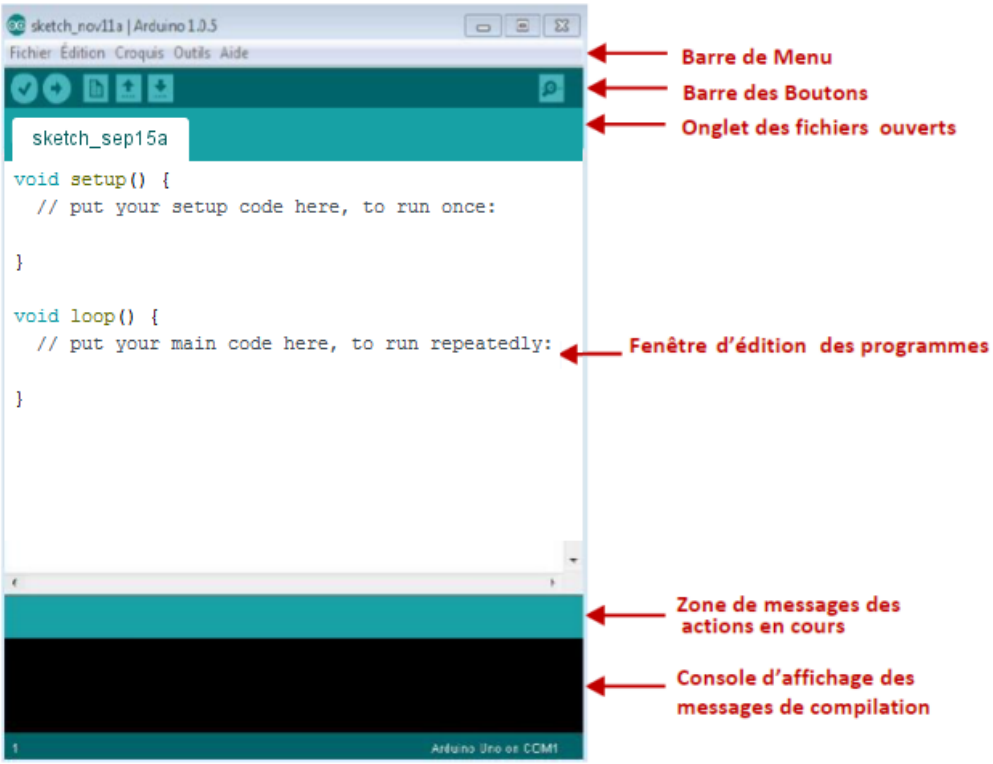
Vous devez télécharger le logiciel sur le site officiel : https://www.arduino.cc/ s'il n'est pas déjà installé sur votre PC.

Sélectionner SOFTWARE et DOWNLOAD.

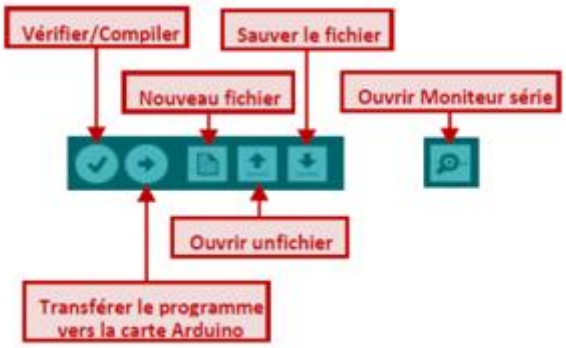
Le logiciel une fois installé, vous devez observer sur votre bureau le raccourci suivant :

Double cliquez dessus pour lancer le programme.

L'interface de programmation est la suivante :



La barre des boutons donne un accès direct aux fonctions essentielles :

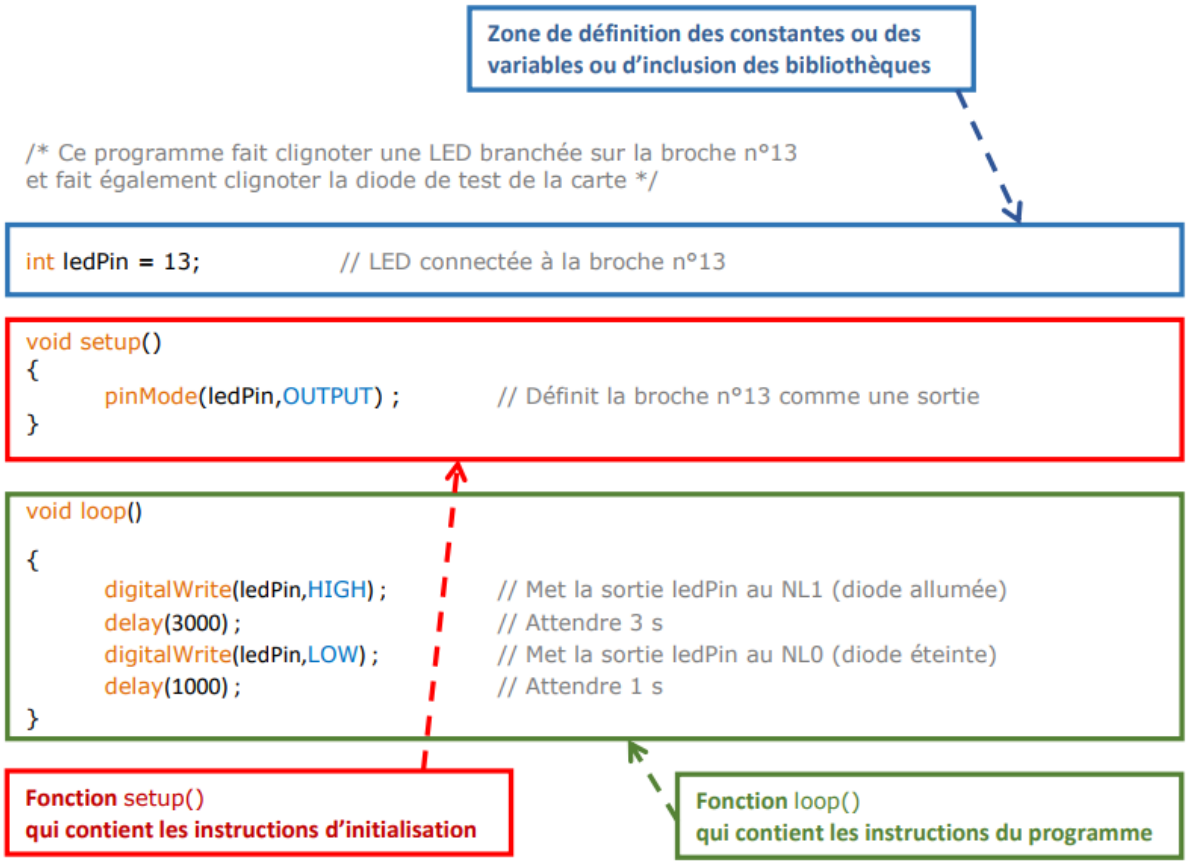


Le logiciel attend que vous réalisiez un programme dans la fenêtre d'édition des programmes. On remarque que des « instructions » sont déjà implémentées dans cette fenêtre. Ces instructions servent à structurer le programme pour que le compilateur puisse le traiter.

La structure d'un programme sous le logiciel Arduino se présente toujours sous la même forme. Il y a un ordre d'exécution des instructions car la carte Arduino va **COMPILER** le programme dans l'ordre ou celui-ci est écrit.

a) Structure générale d'un programme :

Un programme destiné à une carte ARDUINO est constitué de **3 parties** :



La première partie permet de **définir les constantes et les variables** en déclarant leur type. Elle permet également l'inclusion des bibliothèques utilisées dans le programme au moyen de #include.

La fonction setup() contient **les instructions d'initialisation ou de configuration des ressources** de la carte comme par exemple, la configuration en entrée ou sorties des broches d'E/S numériques, la définition de la vitesse de communication de l'interface série, etc.. Cette fonction **n'est exécutée qu'une seule fois** juste après le lancement du programme.

La fonction loop() contient **les instructions du programme** à proprement parlé. Cette fonction sera **répétée indéfiniment** tant que la carte ARDUINO restera sous tension.

Remarque :

Les éléments en fin de chaque ligne de programme sont appelés **COMMENTAIRES** :

- Pour placer des commentaires sur une ligne unique ou en fin de ligne, il faut utiliser la syntaxe suivante :
  `// Cette ligne est un commentaire sur UNE SEULE ligne`
- Pour placer des commentaires sur plusieurs lignes :
  `/* Commentaire, sur PLUSIEURS lignes qui sera ignoré par le programme, mais pas par celui qui lit le code */`

Il est fortement conseillé de laisser des commentaires lors de la réalisation du programme pour pouvoir le relire et le comprendre ultérieurement.

b) Syntaxe du langage Arduino :

La cinquantaine d'éléments de la syntaxe ARDUINO est visible ici
http://www.arduino.cc/en/Reference/HomePage
ainsi qu'à partir du document "index.html" (dans le dossier "Référence" que vous avez téléchargé avec ARDUINO), également accessible dans le menu "Aide" du logiciel.

Différents **types et syntaxes** pour commencer avec Arduino :

Types de variables :

Nom	Description
char	pour stocker des caractères (taille : un byte).
byte	pour stocker un chiffre compris entre 0 et 255.
int	pour stocker un chiffre compris entre 2^{-15} et 2^{15}-1, c'est-à-dire, entre -32,768 et 32,767 (taille : 2 bytes (16 bits)).
unsigned int	n'a pas de signe donc peut stocker des valeurs comprises entre 0 et 2^{16}-1, c'est-à-dire entre 0 et 65,535 (taille : 2 bytes).
unsigned long.	pour des valeurs comprises entre -2,147,483,648 et 2,147,483,647 (taille : 32 bits (4 bytes)).
float	pour nombres décimaux compris entre -3.4028235E+38 et +3.4028235E+38 (taille : 32 bits (4 bytes)).
double	pour nombres décimaux aussi (taille : 8 bytes (64 bits)).

Syntaxe :

Nom	Description
true	Donnée binaire égale à 1 ou donnée numérique différente de 0
false	Donnée binaire ou numérique égale à 0
HIGH	Génération d'un niveau de tension haut sur une des sorties numériques
LOW	Génération d'un niveau de tension bas sur une des sorties numériques
INPUT	Configuration d'une broche numérique en entrée
OUTPUT	Configuration d'une broche numérique en sortie

Les operateurs arithmétiques :

Symbole	Description	Exemple
+	Addition	c = a + b ;
-	Soustraction	c = a - b ;
*	Multiplication	c = a * b ;
/	Division	c = a / b ;
%	Modulo (reste de la division entière)	c = a % b ;

Les operateurs de comparaisons :

Symbole	Description	Exemple
&&	ET logique	if (a && b)
||	OU logique	if (a || b)
==	Égal à	if (a == b)
!=	Différent de	if (a != b)
>	Supérieur à	if (a > b)
<	Inférieur à	if (a < b)
>=	Supérieur ou égal à	if (a >= b)
<=	Inférieur ou égal à	if (a <= b)

Les bibliothèques de fonctions

Une bibliothèque est un ensemble de fonctions utilitaires mises à disposition des utilisateurs de l'environnement Arduino. Les fonctions sont regroupées en fonction de leur appartenance à un même domaine conceptuel (mathématique, graphique, tris, etc).
Quelques exemples qui vont nous servir pour l'initiation au microcontrôleur :

La fonction « delay »

```
delay(tempo) ;
```

Cette fonction génère une pause dont la durée est égale à tempo x 1 ms

La variable tempo peut être de type unsigned long pour autoriser des tempo qui peuvent être très longues.

Exemple :

```
delay(1000) ;       // pause de 1 seconde
```

La fonction « digitalWrite»

```
digitalWrite(numéro de pin, niveau logique) ;
```

Cette fonction permet d'imposer un niveau logique haut **HIGH** (5 volt) ou bas **LOW** (0 volt) sur la sortie numérique de pin sélectionné.

Exemple :

```
digitalWrite (12,HIGH) ;       // La sortie 12 est placée au niveau logique haut
```

La fonction « digitalRead»

```
digitalRead(numéro de pin) ;
```

Cette fonction permet de faire un niveau logique haut **HIGH** (5 volt) ou bas **LOW** (0 volt) sur l'entrée numérique sélectionnée.

Exemple :

```
digitalRead (12) ;       // Le pin 12 est une entrée dont la valeur logique va être lue
```

Vous pouvez visualiser les fonctions déjà définies sous Arduino sur le site constructeur en vous rendant à cette adresse :

https://www.arduino.cc/reference/en/