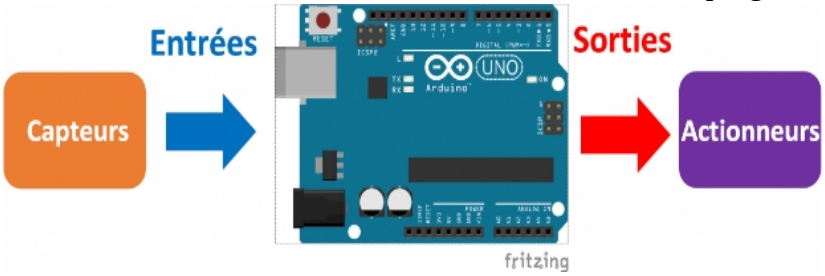
I. Carte à microcontrôleur de type ARDUINO

►Un microcontrôleur est un circuit intégré de faible dimension qui rassemble les éléments essentiels nécessaires au fonctionnement d'un ordinateur : processeur, mémoires (ROM et RAM), unités périphériques et interfaces d'entrées-sorties.

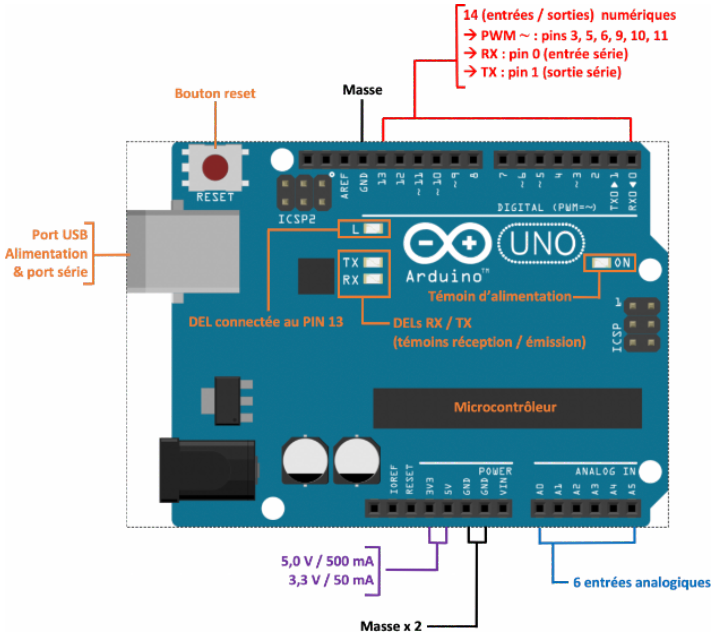
►Les microcontrôleurs sont fréquemment utilisés dans des systèmes embarqués, en robotique et en domotique. Relié à différents types de capteurs, un microcontrôleur peut être programmé afin de produire des signaux électriques ou acquérir des mesures de grandeurs physiques afin de mieux comprendre les lois régissant un phénomène. Les cartes à microcontrôleur de type ARDUINO TM sont les plus populaires en raison de leur robustesse et de leur faible coût.

A. Exemple : la carte ARDUINO - UNO

►Un micro-contrôleur permet, à partir d'événements détectés par des capteurs, de programmer et commander des actionneurs. Une carte Arduino TM est donc une interface programmable.



►L'une des cartes la plus utilisée est la carte Arduino - UNO. La conception matérielle (schémas électroniques et typons) est distribuée sous licence Creative Commons. Le code source de l'environnement de programmation et les bibliothèques embarquées sont disponibles sous licence **LGPL**. Une grande communauté d'amateurs et de passionnés contribuent à développer des applications et à les partager sur le net.



B. Entrées / sorties numériques (ports D0 à D13)

►Chacun des ports **D2 à D13** peut être configuré, via une interface de programmation, en entrée ou en sortie. Les signaux acheminés par ces connecteurs sont des signaux logiques, c'est-à-dire qu'ils ne peuvent prendre que deux états possibles : HIGH (HAUT ; 5,0 V) ou LOW (BAS ; 0 V), par rapport au connecteur de masse GND, qui est toujours par définition à 0 V.

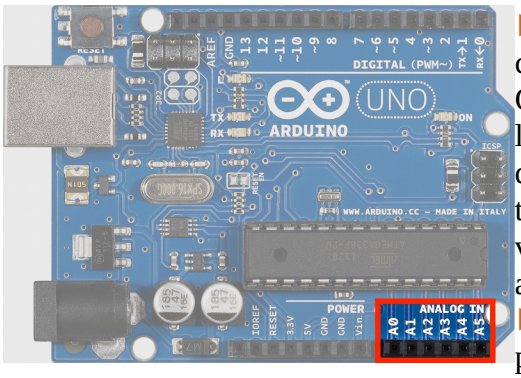
►Remarques : les ports **D0 et D1** sont **réservés** à la communication série, il ne faut donc pas les utiliser pour autre chose. Le port **D13** est relié à la LED de test « L » sur la carte : quand cette sortie est dans l'état HIGH, la LED de test s'allume, dans l'état LOW elle s'éteint.

►Attention : ces connecteurs ne peuvent pas fournir en sortie un courant supérieur à 40 mA, ce qui interdit par exemple, de brancher directement un moteur sur une sortie numérique.

►Remarque : le signe **~ sur les connecteurs 3, 5, 6, 9 10 et 11**, signifie **PWM**

►*Vocabulaire : on qualifie parfois ces entrées/sorties de numériques, de logiques ou de digitales, ces trois adjectifs sont ici considérés comme synonymes.*

C. Les entrées analogiques (ports A0 à A5)



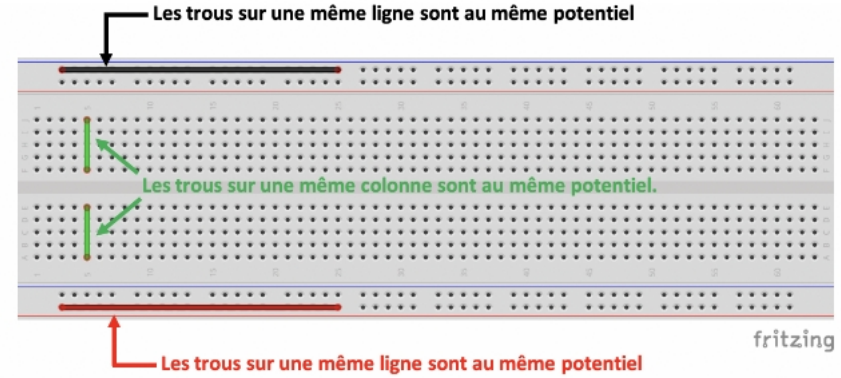
►Une entrée analogique se comporte comme un voltmètre : la carte lit la tension qui est appliquée sur le port correspondant. Cependant les entrées analogiques d'un microcontrôleur codent les tensions (5,0 V max) qui leurs sont appliquées sous la forme d'un nombre entier compris entre 0 et 1023. Le microcontrôleur transforme donc la valeur réelle de la tension appliquée en sa valeur numérique (0 à 1023). C'est le travail du convertisseur analogique/numérique (CAN).

►Contrairement aux entrées/sorties numériques qui ne peuvent prendre que deux états possibles HIGH et LOW, ces six entrées (ports A0 à A5) peuvent admettre 1024 valeurs analogiques comprises entre 0 V (0) et 5,0 V (1023).

►La précision de ces entrées a donc pour ordre de grandeur environ 5 mV (5 V / 1024).

D. "Breadboard"

►Les composants électroniques (dipôles passifs ou actifs, capteurs...) utilisés par la carte à microcontrôleur sont disposés sur une platine électronique appelée "breadboard".



« Breadboard »

►Les caractéristiques pour connecter des composants électroniques sur la "breadboard" sont appliquées ci-dessous.

II. Programmation d'une carte de type ARDUINO

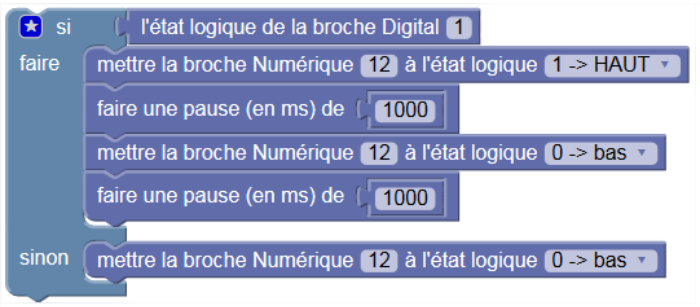
A. IDE ARDUINO

►Il est possible de programmer le comportement de la carte Arduino de deux manières différentes :

Programmation en langage C

```
void setup()
{
  pinMode(1, INPUT);
  pinMode(12, OUTPUT);
}
void loop()
{
  if (digitalRead(1)) {
    digitalWrite(12, HIGH);
    delay(1000);
    digitalWrite(12, LOW);
    delay(1000);
  } else {
    digitalWrite(12, LOW);
  }
}
```

Programmation par blocs



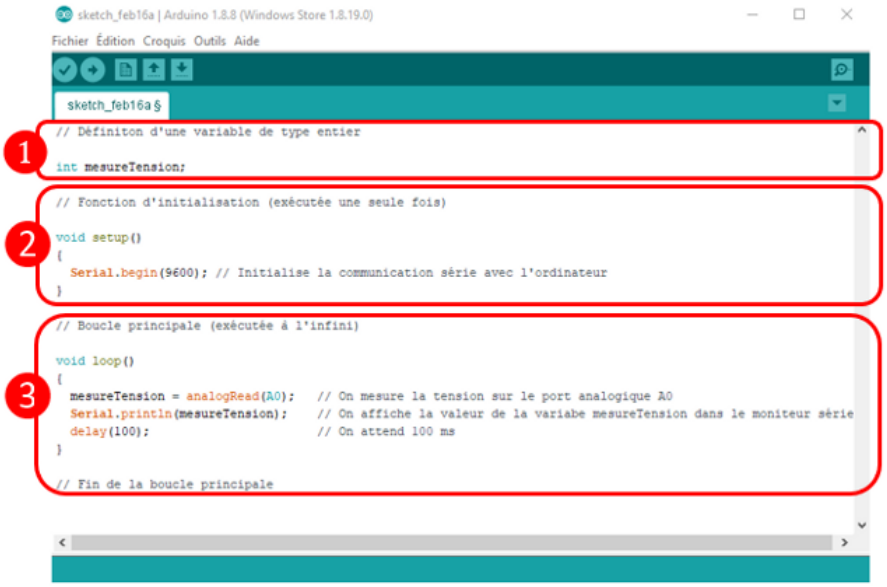
►Sur la **page d'accueil** officielle du projet ARDUINO, vous trouverez toutes les informations utiles au codage de ce type de carte. Vous pouvez télécharger sur votre machine depuis :

https://www.arduino.cc/en/Main/Software

►L'environnement de développement intégré (IDE) qui vous permettra de créer vos programmes afin de piloter votre carte ARDUINO. Le site propose également un IDE en ligne ne nécessitant aucune installation sur votre machine.

B. Structure de base d'un programme ARDUINO :

►Le langage ARDUINO s'appuie sur le langage de programmation **C++**. La structure minimale de tout programme ARDUINO a la forme rappelée ci-dessous :



(1) : déclaration des variables (optionnelle)
(2) : initialisation / configuration des entrées sorties via la fonction setup()
(3) : programme principal qui s'exécute en boucle via la fonction loop()

open source hardware

C. Les variables :

►En raison de la faible quantité de mémoire embarquée sur ce type de carte à microcontrôleur, la gestion du type associé à chaque variable (ou constante) doit être rigoureuse.

►Chaque variable (ou constante) devra être déclarée préalablement dans le programme ARDUINO avec le type compatible avec les valeurs qu'elle est susceptible de prendre.

Type de variable	Taille en bits	Taille en octets	Valeurs stockées
boolean	1	1	True ou False
unsigned char	8	1	0 à 255
char	8	1	-128 à 127
unsigned int	16	2	0 à 65535
int	16	2	-32768 à 37767
float	32	4	-3.4E-38 à 3.4E38

D. Les fonctions

►Une fonction 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 natives, par exemple analogRead(), digitalWrite() ou delay(). Il est possible de déclarer ses propres fonctions dans un programme.

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

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

►Pour exécuter cette fonction dans le programme, il suffira de taper la commande : clignote();

►On peut également faire intervenir un ou des paramètres d'entrée dans une fonction : clignote(5,1)

E. 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.

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

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

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

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

```
void loop()
{
  if (digitalRead(7) == HIGH)
  {
    Serial.println('5') ;
  }
  else
  {
    Serial.println('0') ;
  }
delay(100) ;
}
```

```
compteur = 0;

while(compteur < 200)
{
  compteur++;
}
```

```
// Fait un choix parmi parmi plusieurs messages reçus

switch (message)
{
  case 0 : // si le message est "0"
  // Allume la sortie 3
  digitalwrite(3,HIGH);
  digitalwrite(4,LOW);
  digitalwrite(5,LOW);
  break;
  case 1 : // si le message est "1"
  // Allume la sortie 4
  digitalwrite(3,LOW);
  digitalwrite(4,HIGH);
  digitalwrite(5,LOW);
  break;
  case 2 : // si le message est "2"
  // Allume la sortie 5
  digitalwrite(3,LOW);
  digitalwrite(4,LOW);
  digitalwrite(5,HIGH);
  break;
}
```

open source hardware