

NIVEAU DEBUTANT

# Apprendre à afficher des messages sur le PC à partir de la carte Arduino et à utiliser le Terminal Série.



## Ateliers Arduino

par X. HINAULT

[www.mon-club-elec.fr](http://www.mon-club-elec.fr)



Tous droits réservés – 2012.

### Document gratuit.

Ce support PDF d'atelier Arduino vous est offert.

Pour acheter d'autres supports d'ateliers Arduino, rendez-vous ici :

[http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

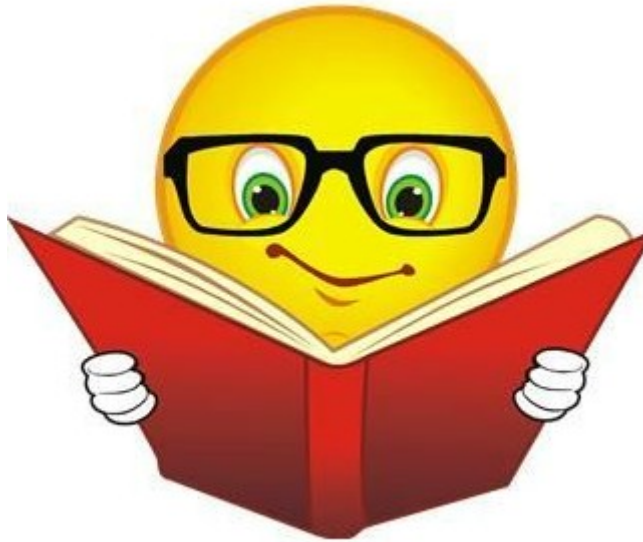
Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

## 1. Intro

L'objectif ici est :

- de comprendre le principe de communication de la carte Arduino vers le PC
- de comprendre la notion de Classe
- de découvrir la classe Serial du langage Arduino et ses principales fonctions pour envoyer des messages au PC
- d'écrire un programme envoyant un message au PC
- d'apprendre à utiliser le Terminal Série pour visualiser les messages sur le PC
- 

... afin d'être en mesure d'afficher des messages dans le Terminal Série du PC. Dans cet atelier, vous allez apprendre à afficher des messages sur le PC à partir de la carte Arduino.



**Prêt ? C'est parti !**

### **Pratique :**

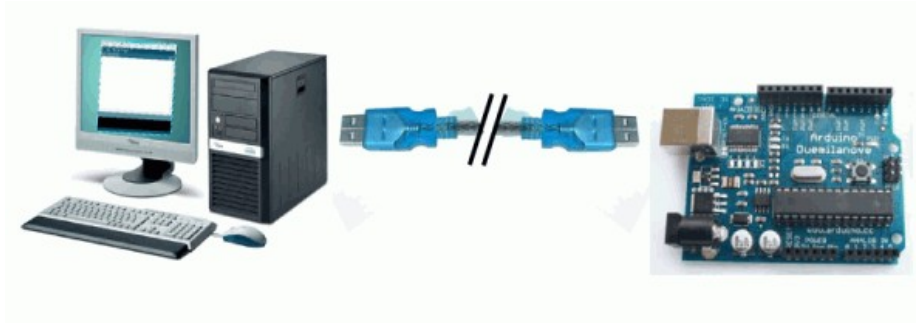
Les codes de cet atelier sont disponibles ici :

[http://www.mon-club-elec.fr/mes\\_downloads/tutos\\_arduino/1a.atelier\\_arduino\\_envoi\\_serie\\_terminal.tar.gz](http://www.mon-club-elec.fr/mes_downloads/tutos_arduino/1a.atelier_arduino_envoi_serie_terminal.tar.gz)

## 2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

### De l'espace de développement Arduino

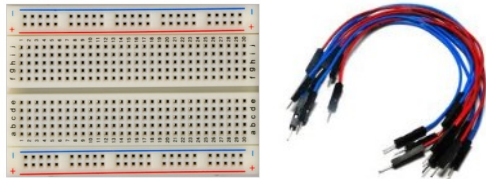


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

### Du nécessaire pour réaliser des montages sans soudure

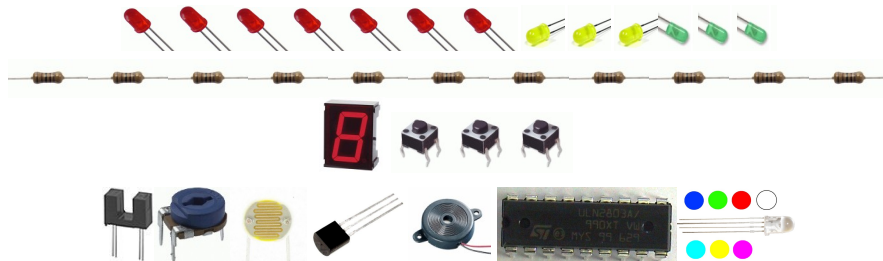


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

### De quelques composants de base



**Pour vous simplifier la vie, nous avons négocié ce kit pour vous !**

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire

<http://www.gotronic.fr/> avec le code express **701710**

**GO TRONIC**  
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

### 3. Principe de communication de l'Arduino vers le PC

Comme vous le savez déjà, la carte Arduino est (re-)programmable à volonté via le port série USB du PC : c'est ce qui fait toute la simplicité de son utilisation.

Mais la carte Arduino est également capable très simplement de communiquer avec le PC pendant l'exécution d'un programme :

- pour **envoyer des messages vers le PC** (chaîne texte, valeurs numériques) afin de les visualiser sur l'écran sous forme texte ou même sous forme de graphiques (usage avancé) !
- pour **recevoir des messages depuis le PC** (chaîne texte, valeurs numériques) ce qui permettra de contrôler la carte Arduino à partir du clavier ou de la souris par exemple (usage avancé) !

Le langage Arduino dispose de toutes les instructions nécessaires pour réaliser et programmer cette communication au sein d'un programme comme nous allons le voir ici.

La possibilité offerte par le langage Arduino d'afficher des messages sur le PC est l'une des grandes forces de ce système : il est ainsi possible de « voir » de l'intérieur comment un programme fonctionne, ce qui est très puissant pour comprendre, mais aussi mettre au point ses programmes.

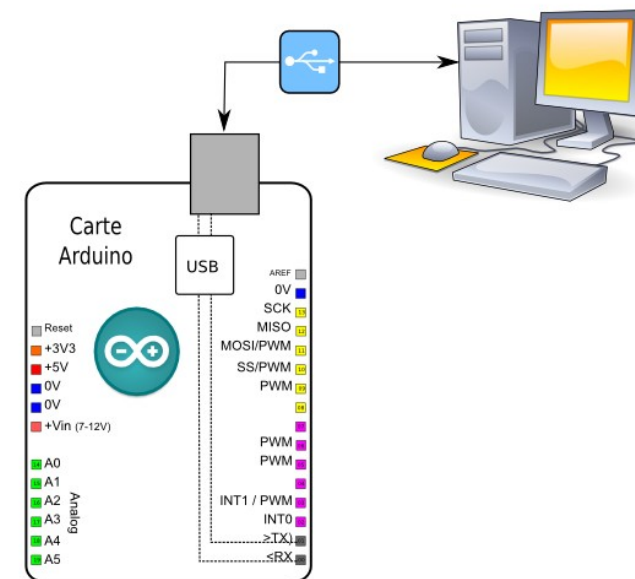
Au final, la carte Arduino programmée pourra fonctionner de 2 façons :

- soit en autonomie, déconnectée du PC une fois programmée,
- soit en communiquant avec le PC pendant l'exécution du programme, réalisant un véritable périphérique USB programmable et personnalisable à souhait !

Programmation par le port USB



Communication par le port USB pendant l'exécution du programme !



## 4. Notion de « Classe »

Dans les langages de programmation actuels, les concepteurs ont imaginé la possibilité de pouvoir rassembler des fonctions ensemble lorsqu'elles s'appliquent à une même fonctionnalité.

Par exemple, toutes les fonctions qui s'occupent des opérations mathématiques vont pouvoir être regroupées ensemble.

**A retenir : On appelle « classe » un regroupement de fonctions.**

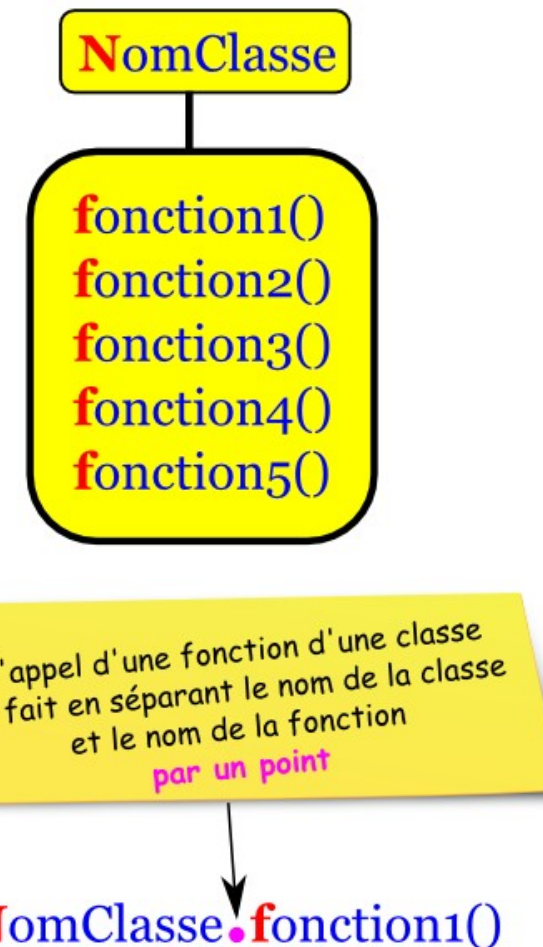
Tout comme une fonction, une classe aura un nom : pour distinguer une classe d'une fonction, **le nom d'une classe commencera par une MAJUSCULE.**

En pratique, lorsque l'on programme en langage Arduino, on n'a pas besoin de créer de classes (ouf !). Mais le langage Arduino ou ses bibliothèques comporte plusieurs classes et il faut donc comprendre ce concept :

- Ainsi, toutes les fonctions qui gèrent la communication avec le port série USB sont rassemblées dans une classe appelée **Serial** : nous allons utiliser cette classe ici.
- la classe LiquidCrystal pour la gestion d'un afficheur LCD
- la classe Servo pour la gestion d'un servomoteur
- etc...

**En pratique, pour utiliser une fonction d'une classe du langage Arduino, on utilisera le nom de la classe + un point + le nom de la fonction.**

Remarque technique : les instructions de base du langage Arduino, même si elles ne sont pas précédées par un nom de classe, appartiennent toutes à une même classe (implicite) : celle du cœur (ou core) du langage Arduino.



## 5. A la découverte de votre première classe : la classe Serial

Ainsi, comme on vient de le dire :

**On appelle « classe » un regroupement de fonctions.**

La première classe du langage Arduino que je vous propose de découvrir est celle qui rassemble toutes les fonctions utilisées pour la communication série USB : cette classe s'appelle **Serial** !

Les fonctions de la classe Serial sont au nombre d'une dizaine. Je vous propose ici de découvrir les 3 fonctions qui permettent d'écrire un programme pour afficher des messages vers le PC :

- `begin()` : fonction d'initialisation de la communication USB
- `print()` : fonction d'affichage d'un message sans saut de ligne
- `println()` : fonction d'affichage d'un message avec saut de ligne

Comme on l'a déjà dit :

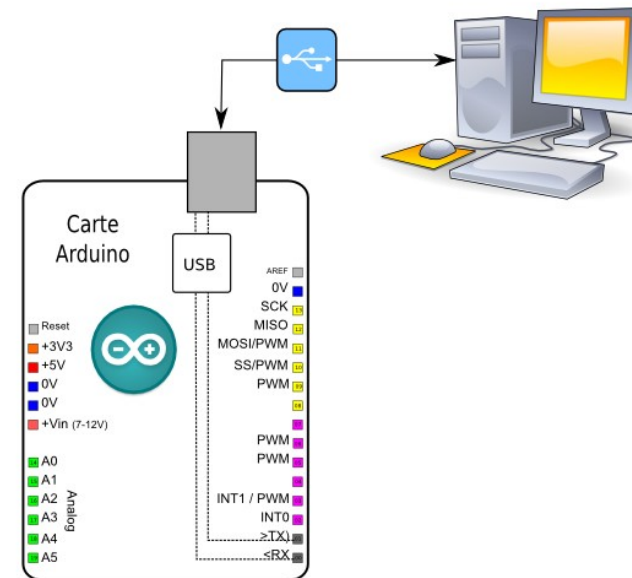
**En pratique, pour utiliser une fonction d'une classe du langage Arduino, on utilisera le nom de la classe + un point + le nom de la fonction.**

Dans le cas de la classe Serial, on fera :

- `Serial.begin()` pour appeler la fonction `begin()`
- `Serial.print()` pour utiliser la fonction `print()`
- `Serial.println()` pour utiliser la fonction `println()`

A présent, nous avons tous les éléments pour envoyer des messages au PC depuis notre carte Arduino !

Remarque technique : habituellement, comme on le verra, on doit déclarer une nouvelle instance d'une classe avant de l'utiliser, sauf avec la classe Serial (tant mieux!).





## 6. Apprendre à lire la fiche technique d'une fonction Arduino

Une instruction Arduino est une fonction : sa fiche technique va donc indiquer :

- son rôle, ce qu'elle fait (= description)
- la manière de l'écrire (=syntaxe)
- le ou les paramètres qu'elle reçoit
- la valeur renvoyée

Le langage Arduino comporte :

- un trentaine de fonctions de base
- plusieurs bibliothèques utilisables en option incluant plusieurs dizaines de fonctions

**Pas de panique : on peut démarrer avec quelques instructions pour écrire un programme et l'apprentissage se fait au fur et à mesure.**

Au fur et à mesure de vos besoins, vous trouverez les fiches techniques de toutes les fonctions du langage Arduino :

- depuis le logiciel Arduino : Menu Help > Reference
- en anglais, sur le site de la référence officielle : <http://arduino.cc/en/Reference/HomePage>
- en français, sur mon site : <http://www.mon-club-elec.fr/>



**Référence Arduino français**

Accueil Pour Débuter Langage Bibliothèques Cartes Arduino FAQ Apprendre

Main  
Référence : Langage Arduino : [ Mini | Standard | **Étendue** | Maxi ] Bibliothèques : [ Vue d'ensemble | Synthèse ] Infos : [ Comparaison | Changements ]

**Référence du langage Arduino en français**

Voir la [référence étendue](#) pour davantage de fonctions avancées du langage Arduino et la page des [bibliothèques](#) pour interfaçage avec des types de matériel particuliers (afficheur LCD par exemple).

Les programmes Arduino peuvent être divisés en trois parties principales: la structure, les valeurs (variables et constantes) et les fonctions. Le langage Arduino est basé sur les langages C/C++.

Ici un [guide d'écriture](#) qui aidera ceux qui veulent écrire des programmes d'exemple.

Déjà 27001 visites sur cette page.

**NOUVEAU : ARDUINO 0018 est désormais disponible en version française !**

**Structure**

**FONCTIONS DE BASE**

Ces deux fonctions sont obligatoires dans tout programme en langage Arduino :

- void setup()
- void loop()

**STRUCTURES DE CONTRÔLE**

**Variables et constantes**

Les variables sont des expressions que vous pouvez utiliser dans les programmes pour stocker des valeurs, telles que la tension de sortie d'un capteur présente sur une broche analogique.

**CONSTANTES PRÉDÉFINIES**

Les constantes prédéfinies du

**Fonctions**

**ENTRÉES/SORTIES NUMÉRIQUES**

- pinMode(broche, mode)
- digitalWrite(broche, valeur)
- int digitalRead(broche)

**ENTRÉES ANALOGIQUES**

**Search**

Go

Accueil

**Pour débuter**

- Arduino : Mode d'emploi !
- Présentation
- Matériel Arduino
- Logiciel Arduino

**Langage Arduino**

- Référence Mini
- Référence Standard
- Référence Étendue **N**

**Bibliothèques Arduino**

- Librairie Série
- Librairie Afficheur LCD

## 7. La fonction **Serial.begin()**

### Description

Fixe le débit de communication en nombre de caractères par secondes (l'unité est le baud) pour la communication série.

### Syntaxe

```
Serial.begin(debit);
```

### Paramètres

debit: débit de communication en caractères par seconde (ou baud).

Pour communiquer avec l'ordinateur, utiliser l'un de ces débits : 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. (Plus le débit est élevé et plus la communication est rapide... )

**En pratique utiliser une valeur comprise entre 9600 et 115200.  
Typiquement 115200 fonctionne très bien !**

### Valeur renvoyée

Aucune

### Exemple

```
Serial.begin(115200);
```

Le baud est une unité de mesure utilisée dans le domaine des télécommunications en général, et dans le domaine informatique en particulier. **Le baud est l'unité de mesure du nombre de symboles transmissibles par seconde.**

Le terme « baud » provient du patronyme d'Émile Baudot, l'inventeur du code Baudot utilisé en télégraphie.

Il ne faut pas confondre le baud avec le bps ou bit par seconde, ce dernier étant l'unité de mesure du nombre d'information effectivement transmise par seconde. Il est en effet souvent possible de transmettre plusieurs bits par symbole. La mesure en bps de la vitesse de transmission est alors supérieure à la mesure en baud.

source : [wikipedia](https://fr.wikipedia.org/wiki/Baud)



## 8. Les fonctions **Serial.print()** et **Serial.println()**

### Description

**Serial.println()** : Affiche les données sur le port série **suivi d'un passage à la ligne suivante**.

**Serial.print()** : Affiche les données sur le port série sans passage à la ligne suivante.

### Syntaxe

**Serial.println(data);**

**Serial.print(data);**

### Paramètres

**data** : tous types de données entières incluant les char, chaînes de caractères et floats (nombre à virgule).

Les floats (nombre à virgules) sont supportés avec une précision de 2 à plusieurs décimales.

### Valeur renvoyée

Aucune

### Exemple

**Serial.println**(« Salut! »);

Usage avancé :

- les fonctions print et println permettent l'affichage des valeurs numériques et des variables dans de nombreux formats variés, notamment binaire et hexadécimal.
- Pour stocker une chaîne en mémoire FLASH, on fera précéder la chaîne à afficher par F() selon **Serial.println(F(« montexte »))**;

```
Serial.print(78); // affiche "78"
Serial.print(1.23456); // affiche "1.23"
Serial.print(byte(78)); // affiche "N" (dont la valeur ASCII est 78)
Serial.print('N'); // affiche "N"
Serial.print("Hello world."); // affiche "Hello world."
```

```
Serial.print(78, BYTE); // affiche "N"
Serial.print(78, BIN); // affiche "1001110"
Serial.print(78, OCT); // affiche "116"
Serial.print(78, DEC); // affiche "78"
Serial.print(78, HEX); // affiche "4E"
Serial.print(1.23456, 0); // affiche "1"
Serial.print(1.23456, 2); // affiche "1.23"
Serial.print(1.23456, 4); // affiche "1.2346"
```

## 9. « Hello world ! » : Ecrire votre 1er programme Arduino envoyant un message vers le PC via le port USB

Le principe d'utilisation de la communication USB dans un programme Arduino consiste à :

- initialiser le débit (ou vitesse) de communication une fois pour toute au début du programme (dans la fonction `setup()` )
- utiliser les fonctions `Serial.println()` lorsqu'on en a besoin :
  - soit dans `setup()` pour afficher des messages une seule fois au début du programme,
  - soit dans `loop()` pour afficher des messages à intervalles réguliers ou en boucle.

Notre premier programme Série va :

- initialiser la communication à 115200 bauds avec l'instruction `Serial.begin(vitesse)`
- afficher un message toutes les secondes avec les instructions suivantes :
  - pour afficher une chaîne :

```
Serial.println(« mon message »);
```

**Attention : Une chaîne de caractères s'écrit entre « »**

- pour réaliser une pause d'une seconde (=1000 millisecondes) : l'instruction Arduino `delay(duree)` – **ne pas oublier cette pause sinon le port série va saturer car Arduino va très très vite !** :

```
delay(1000);
```

### Note technique :

les chaînes de caractères de message sont stockées en mémoire RAM par défaut, ce qui ne pose pas de problème pour une dizaine de messages. Dès que l'on va écrire beaucoup de messages, il faudra les écrire en mémoire FLASH ce qui se fait depuis Arduino 1.0 par

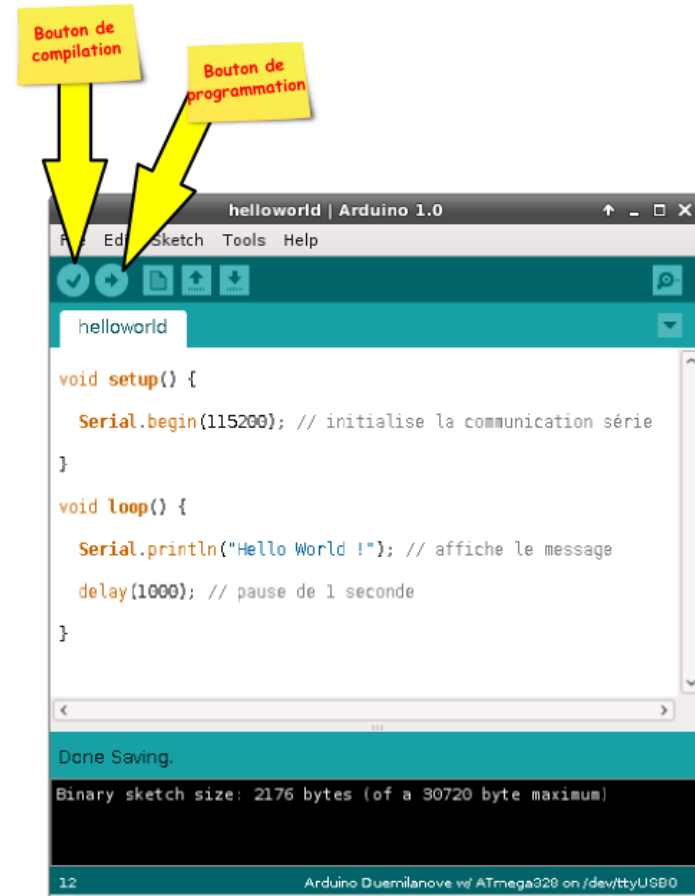
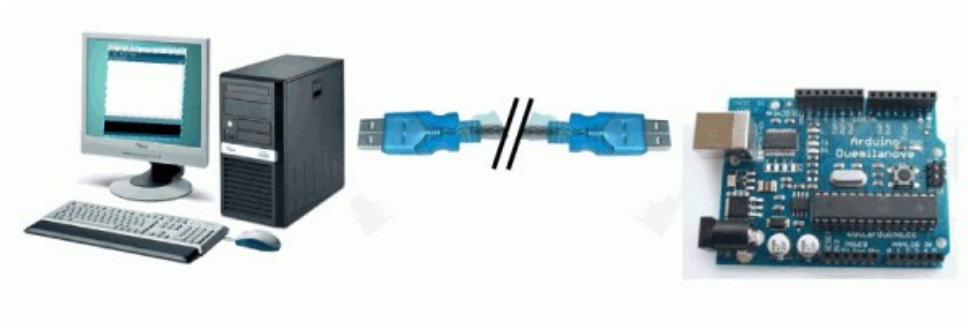
```
Serial.print(F(« message »));
```

```
void setup() {  
  
    Serial.begin(115200); // initialise la communication  
    série  
  
}  
  
void loop() {  
  
    Serial.println("Hello World !"); // affiche le message  
  
    delay(1000); // pause de 1 seconde  
  
}
```

## 10. « Hello world ! » : Programmer votre programme dans la carte Arduino

Une fois le programme écrit :

- connecter (si ce n'est déjà fait) votre carte Arduino à votre ordinateur
- le compiler pour vérifier l'absence d'erreur en cliquant sur le bouton de compilation : vous devez obtenir le message « Done compiling » dans la barre d'état du logiciel Arduino. Si un message d'erreur apparaît, corriger votre code et ré-essayer.
- **vérifier :**
  - **la carte utilisée (Tools>Board)**
  - **et le port (Tools>Serial Port) utilisé**
- cliquer sur le bouton programmer :
  - le programme est transféré dans la carte Arduino : vous devez voir 2 LEDs de la carte Arduino clignoter rapidement
  - une fois le transfert terminé, le message « Done uploading » s'affiche dans la barre d'état du logiciel Arduino



## 11. Lancer et paramétrer le Terminal Série pour afficher sur le PC les messages envoyés par Arduino

Une fois que la carte Arduino est programmée, elle envoie à intervalle régulier des messages vers le PC. Concrètement, vous ne voyez rien se passer à ce stade : **pour voir les messages envoyés par la carte Arduino au PC, vous allez devoir utiliser un logiciel de visualisation.**

Heureusement pour nous, le logiciel Arduino (qui est vraiment très pratique !) dispose d'un tel outil de visualisation : **c'est le Terminal Série**. Pour le lancer :

- soit Menu Tool > Serial Monitor
- soit clic sur le bouton Terminal Serie

Une fois la fenêtre du Terminal Série ouverte :

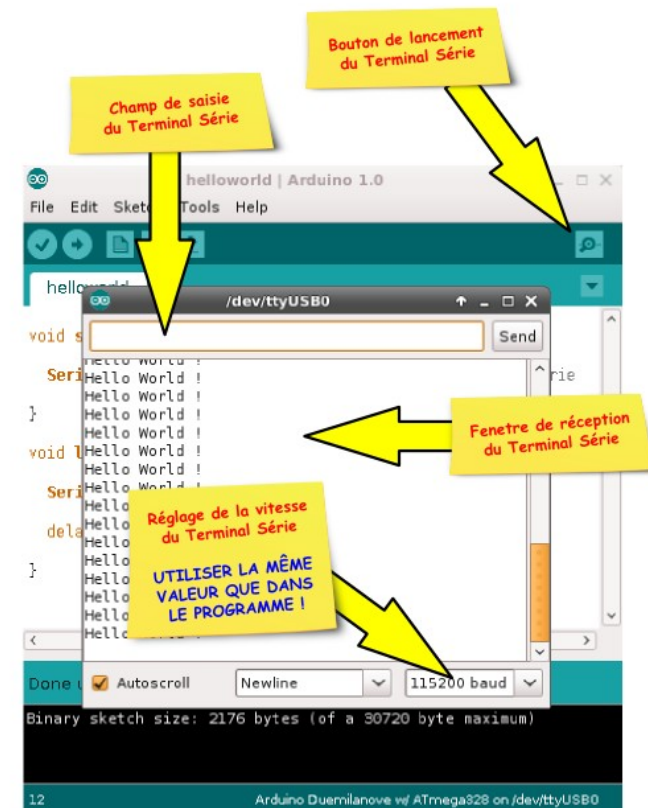
- vérifier que la vitesse de communication est la même que celle que vous avez fixé dans le programme Arduino
- une fois fait vous devriez voir les messages s'afficher dans la fenêtre.

**Info : le Terminal Série dispose d'un champ de saisie pour envoyer des caractères vers Arduino mais nous ne l'utiliserons pas pour le moment.**

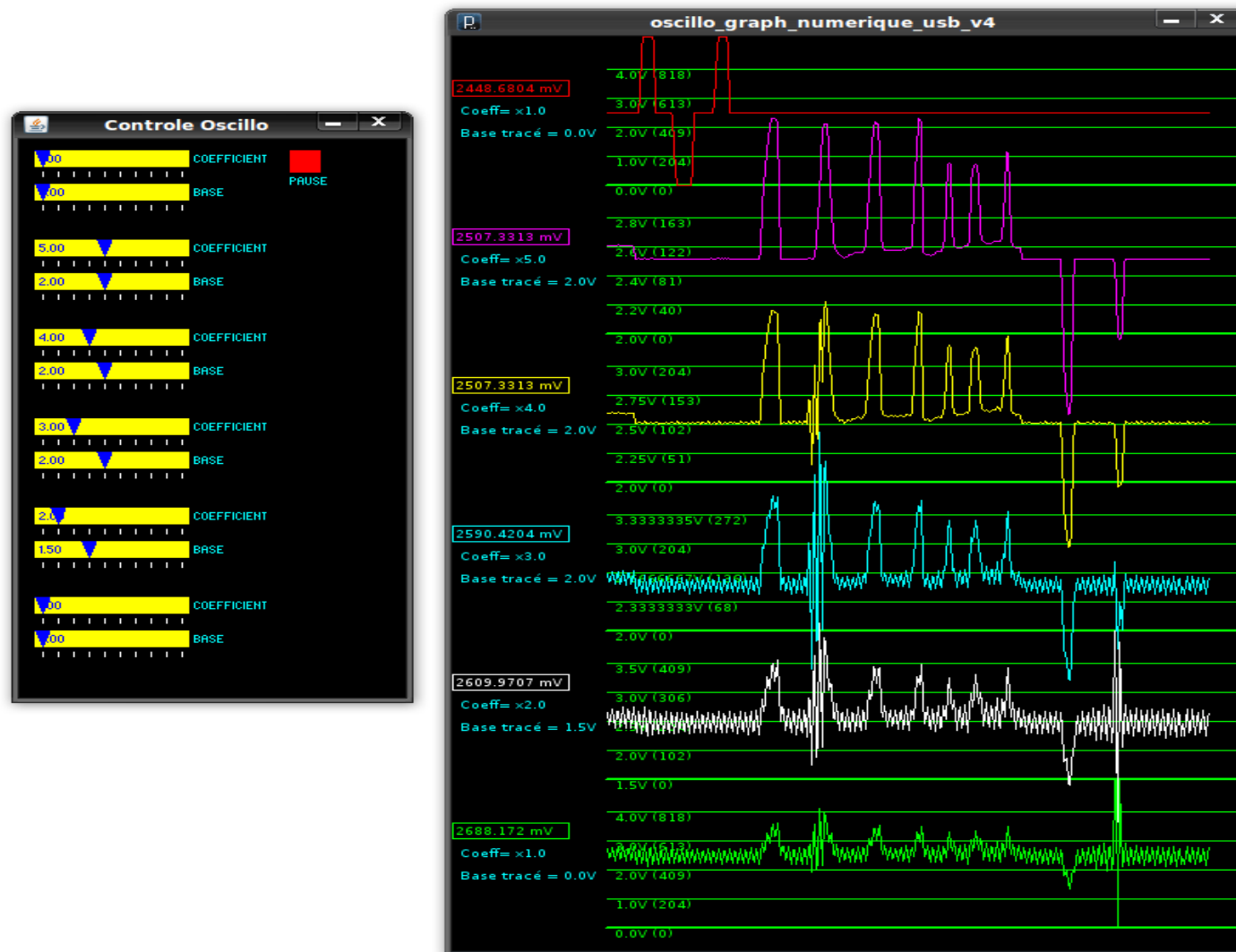
Cette fois, ça y est : vous savez afficher des messages en provenance de la carte Arduino sur votre PC ! A présent, vous pouvez vous amuser à :

- varier les messages,
- tester l'affichage de valeurs numériques entières ou à virgule.

**Truc : Pour relancer l'affichage de vos messages au début, faites un appui bref sur le bouton **reset** de votre carte Arduino.**



## 12. Exemple avancé de communication USB de l'Arduino vers le PC : un mini-oscilloscope 6 voies !



Exemple d'interface graphique Processing recevant des données en provenance d'une carte Arduino. Code sur [www.mon-club-elec.fr](http://www.mon-club-elec.fr)

### 13. Les éléments du langage Arduino étudiés dans cet atelier

Structure

Variables et constantes

Fonctions

Classe Serial

- [begin\(\)](#)
- [print\(\)](#)
- [println\(\)](#)

La documentation complète du langage Arduino en français est disponible ici :  
[http://www.mon-club-elec.fr/pmwiki\\_reference\\_arduino/pmwiki.php?n=Main.ReferenceMaxi](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi)

#### **14. A présent, vous devriez être capable :**

- d'écrire un programme simple permettant d'afficher des messages sur le PC
- d'ouvrir et paramétrer le Terminal Série pour afficher des messages dans le logiciel Arduino.



# Table des matières

Intro |

Principe de communication de l'Arduino vers le PC |

Notion de « Classe » |

A la découverte de votre première classe : la classe Serial |

Apprendre à lire la fiche technique d'une fonction Arduino |

La fonction Serial.begin() |

Les fonctions Serial.print() et Serial.println() |

« Hello world ! » : Ecrire votre 1er programme Arduino envoyant un message vers le PC via le port USB |

« Hello world ! » : Programmer votre programme dans la carte Arduino |

Lancer et paramétrer le Terminal Série pour afficher sur le PC les messages envoyés par Arduino |

Exemple avancé de communication USB de l'Arduino vers le PC : un mini-oscilloscope 6 voies ! |

A présent, vous devriez être capable : |

**Bravo !**  
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

[http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)