

Apprendre à utiliser un afficheur LCD alpha-numérique standard avec Arduino et découvrir les fonctions de la librairie Arduino LiquidCrystal.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : 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.

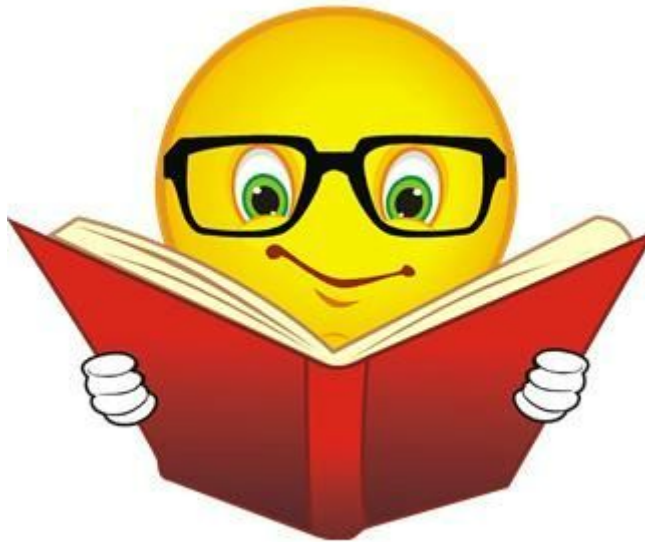
En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

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 découvrir l'afficheur LCD alpha-numérique standard (=qui affiche des lettres et des chiffres)
 - d'apprendre à préparer un afficheur LCD standard pour être utilisé simplement avec Arduino
 - de découvrir et d'apprendre à utiliser la librairie la librairie Arduino LiquidCrystal
 - de découvrir les fonctions de la librairie Arduino LiquidCrystal
- ... afin d'être en mesure de réaliser des applications utilisant un afficheur LCD pour afficher des messages

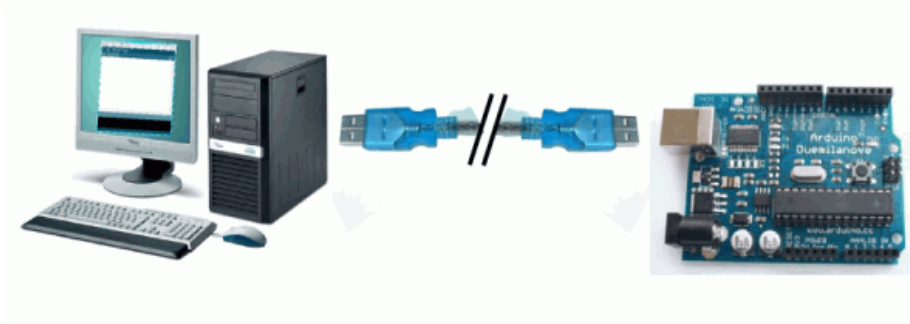


Prêt ? C'est parti !

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

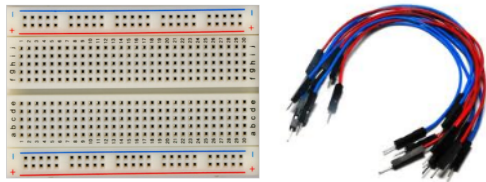


L'espace de développement Arduino associe :

- 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

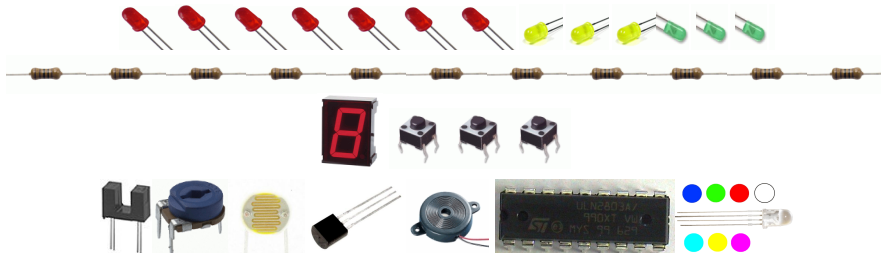


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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

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. Matériel spécifique nécessaire pour cet atelier

Pour cet atelier vous aurez besoin également :

D'un afficheur LCD alpha-numérique standard 4 lignes x 20 colonnes

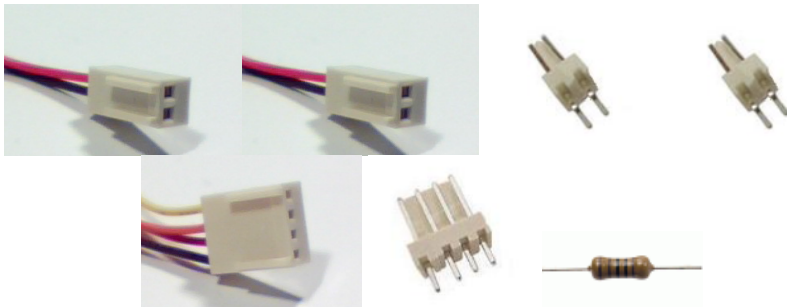


L'afficheur LCD alphanumérique standard 4 lignes x 20 colonnes est facile à utiliser avec la carte Arduino à l'aide de 6 broches numériques. Cet afficheur permet d'afficher des lettres et des chiffres sur 4 lignes et 20 colonnes. Un équipement idéal pour créer des applications autonomes réalisant des mesures, etc...

Existe en différents formats 2x8, 1x16, 2x16, etc... Le format 4x20 présente l'avantage d'être « à l'aise » pour afficher ses messages. Existe également en mode réflectif ou avec rétro-éclairage.

disponible chez : <http://www.gotronic.fr/> | De 8 à 20€ selon modèle
4x20 : code 03351 | 2x16 : code 03313

Du matériel nécessaire pour « préparer » un afficheur standard pour une utilisation simplifiée avec Arduino



Pour être utilisable facilement avec Arduino, l'afficheur LCD standard brut nécessite une petite préparation assez simple à l'aide d'éléments de connectique simples et une résistance :

- 2 connecteurs femelles sur fils – 2 contacts (code : 09825)
- 1 connecteur femelle sur fils – 4 contacts (code : 09827)
- 2 connecteurs droits – 2 contacts (code : 09815)
- 1 connecteur droit – 4 contacts (code : 09817)
- 1 résistance 1KOhms – 1/4w (code : 04036)

disponible chez : <http://www.gotronic.fr/> avec les codes indiqués

Des outils nécessaires pour réaliser des soudures



Pour réaliser des soudures, vous aurez besoin :

- d'un fer à souder à pointe fine, 25-30W et de son support
- de soudure d'étain 60% dite « électronique » - Ø 1mm
- d'un rouleau de scotch (pratique pour faire tenir les composants)
- d'une petite pince coupante
- d'une pompe à dessouder (pour rattraper le coup au besoin)

disponible chez : <http://www.gotronic.fr/> ou en magasin de bricolage

4. Fiche Technique : Afficheur LCD alpha-numérique standard

Description

Un afficheur alpha-numérique standard est un composant que vous connaissez bien et qui équipe toutes sortes de dispositifs de la vie courante. C'est un module qui permet d'afficher des messages à base de lettres et de chiffres assez simplement avec Arduino.



Selon les modèles, il existe des variantes, notamment :

- réflectif ou rétro-éclairé (consomme plus)
- avec LED de rétro-éclairage (utilisable dans l'obscurité)
- couleur de l'affichage : soit noir sur fond vert classique, soit blanc sur fond bleu, etc...

Il existe différentes tailles également :

- en 4 lignes x 20 colonnes, en 2 lignes x 8 colonnes, en 1 ligne x 16 colonnes.
- en pratique, un 4 lignes x 20 colonnes permet d'être à l'aise, et c'est celui que je conseille. Compter 20€ pièce. Les autres modèles sont moins chers, dès 8€.

Important

Toutes les variantes d'afficheurs dits « standards » (comme sur la photo) sont utilisables avec Arduino assez simplement comme nous allons le voir, jusqu'à 80 caractères (soit maximum 4 x 20) .

Ne pas confondre les afficheurs standards avec les afficheurs LCD « série » souvent vendus plus chers pour « économiser des broches » et nécessitant une librairie de communication spécifique. Nous ne traitons pas de ces afficheurs ici car ils sont particuliers à tel ou tel fabricant et pas toujours universels. Un afficheur LCD standard n'utilise que 6 broches numériques, ce qui n'est quand même pas la « mer à boire »... et ne justifie pas la différence de prix.

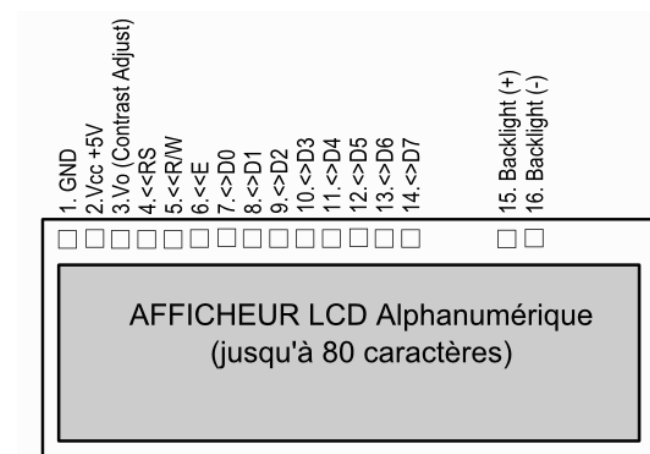
Brochage

Un afficheur LCD standard dispose typiquement :

- de 2 broches d'alimentation et d'une broche de réglage du contraste
- de 3 broches numériques de commande RS, E et RW
- de 8 broches numériques de données notées D0 à D7
- +/- de 2 broches correspondant à la LED de rétro-éclairage

Caractéristiques électriques

- Un afficheur LCD standard nécessite une tension d'**alimentation** typiquement de 5V et va consommer au plus quelques dizaines de mA : il sera donc directement utilisable et alimentable par le +5V de la carte Arduino (**pour mémoire, cette alimentation laisse 300mA dispo**).
- L'afficheur dispose par ailleurs de plusieurs **broches numériques** de contrôle qui sont de type numérique et connectables directement à la carte Arduino.
- Certains modèles enfin disposent d'une **LED interne de rétro-éclairage** qui permet d'utiliser le LCD dans l'obscurité. A utiliser comme une LED classique (càd avec une résistance en série) .



Les broches de l'afficheur LCD standard : ne vous laisser pas impressionner, c'est simple !

Mode de fonctionnement

Un afficheur LCD alpha-numérique standard peut fonctionner :

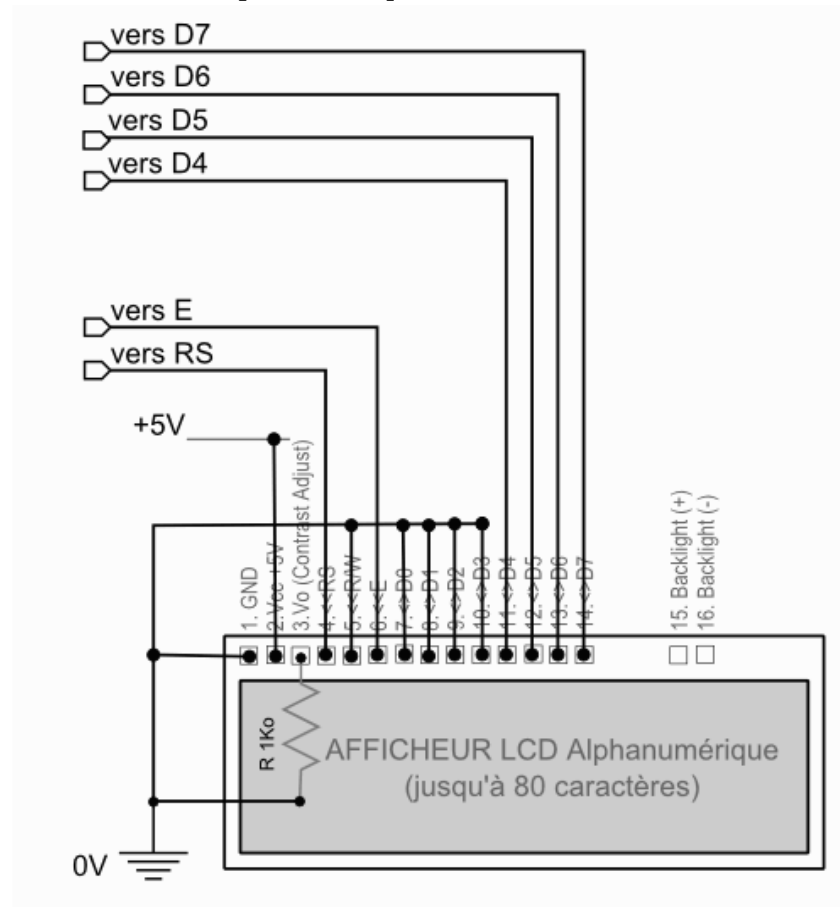
- soit en mode dits « 8 bits » et dans ce cas nécessite 8 broches de données + 3 broches de commandes soit 11 broches !
- soit en mode dits « 4bits » et dans ce cas nécessite 4 broches de données + 2 broches de commandes soit seulement **6 broches**. **C'est ce mode de fonctionnement que nous allons utiliser.**

5. Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : le schéma théorique

Comme on vient de la dire, un afficheur LCD peut être utilisé en mode "4 bits" ou "8 bits". Dans le mode simplifié, dit « 4 bits », on n'utilise que 4 lignes de données pour envoyer les données à l'afficheur. C'est ce mode qui est le plus pratique. Les connexions à réaliser sont alors les suivantes :

- broches de commande **RS** et **E** connectées à **2 broches numériques en sortie** de la carte Arduino
- broches de données **D4** à **D7** connectées à **4 broches numériques en sortie** de la carte Arduino
- Le **+** et **-** connectées au **5V** et à la **masse (0V)**
- Une résistance de réglage du contraste entre le +5V et la broche Vo (en pratique 1Kohm – 1/4w fait l'affaire). On pourrait aussi utiliser une résistance variable, mais c'est plus compliqué ici, surtout qu'il suffit d'incliner plus ou moins l'afficheur pour régler le « contraste » apparent...
- la broche **RW** et les broches **Do - D3** non utilisées et connectées à la **masse (=0V)**.
- enfin, si l'afficheur intègre une LED de rétroéclairage, on la connectera comme une LED classique (pas utilisée ici).

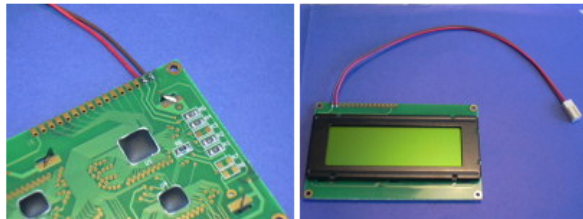
Voici le schéma de cette connexion simplifiée de l'afficheur LCD alpha-numérique :



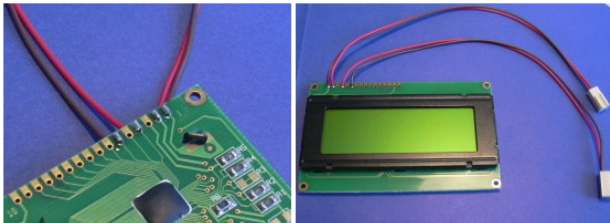
6. Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : description concrète

La préparation de l'afficheur n'est pas très compliquée à réaliser et permettra ensuite d'utiliser très facilement l'afficheur avec une carte Arduino. Voici la procédure en images.

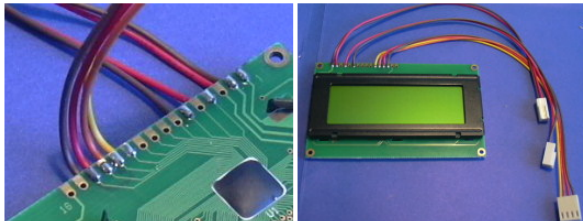
On commence par souder le connecteur 2 broches sur fils sur le + et - de l'afficheur :



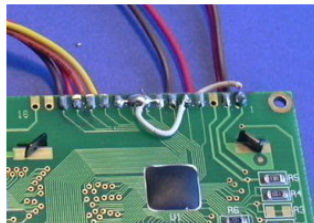
On soude ensuite le connecteur 2 broches sur fils sur les broches RS et E :



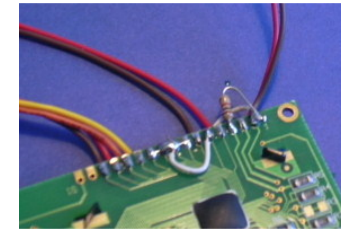
On soude ensuite le connecteur 4 broches sur fils sur les broches D4 à D7 :



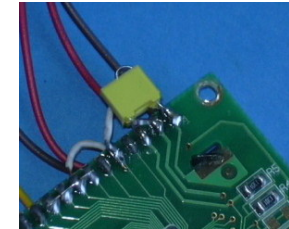
On soude ensuite entre-elles les broches D0 à D3, la broche RW et la broche 0V



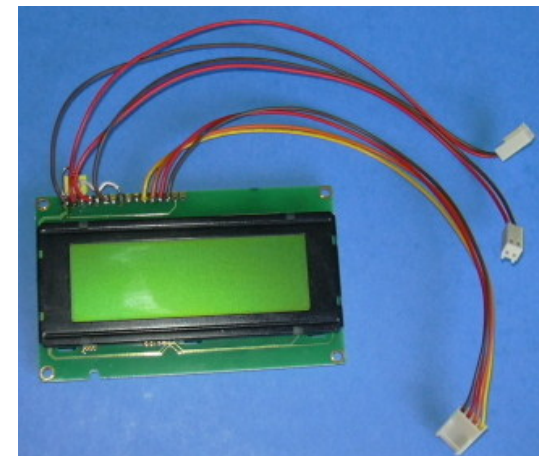
On soude la résistance de 1Ko entre le 0V et la broche Vo



On peut enfin souder un condensateur 100nF entre le + et le - (pas indispensable... limite les « parasites », c'est tout.)



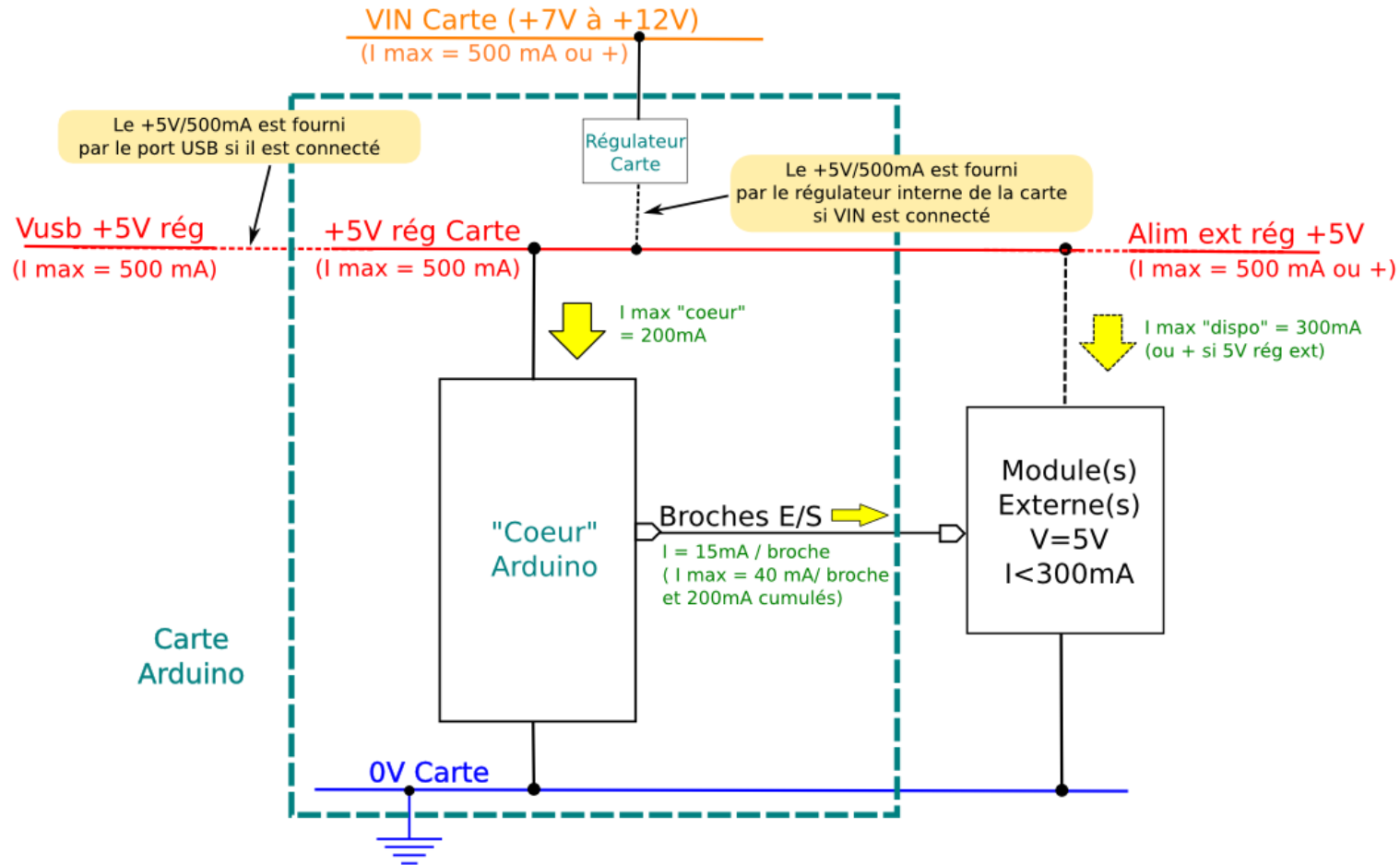
Voici à quoi ressemble le LCD préparé fini et prêt à l'emploi avec votre carte Arduino :



Rien de bien sorcier.. Cette fois, vous êtes prêts à utiliser votre LCD !

7. Schéma électrique type d'utilisation d'un afficheur LCD avec une carte Arduino

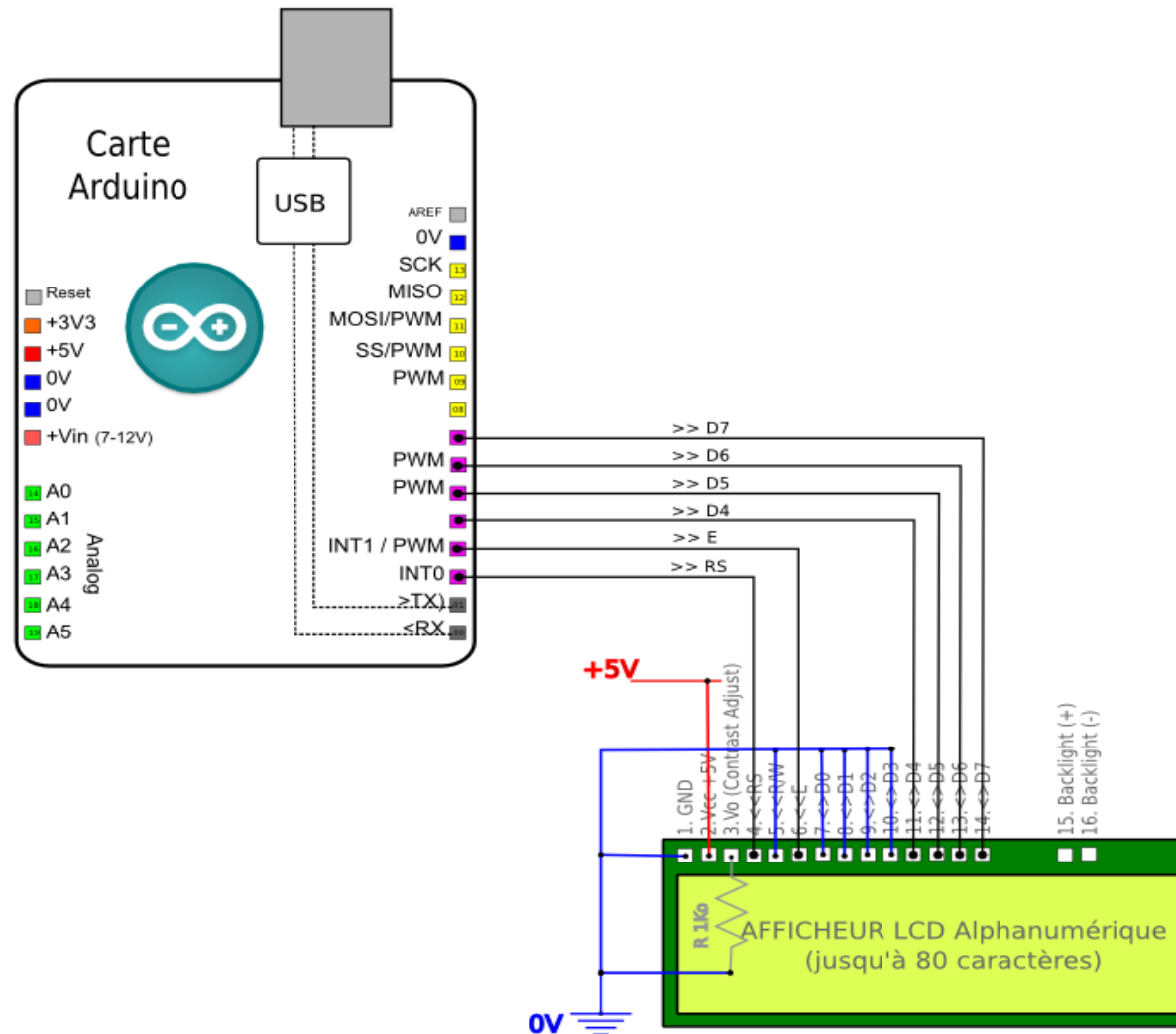
L'afficheur LCD ne va demander que quelques mA et est alimentable en 5V : on va donc pouvoir l'alimenter directement sur l'alimentation 5V de la carte Arduino (qui rappelons-le peut fournir 500mA – 200mA (conso du coeur Arduino) = 300mA disponibles environ).



8. Utilisation d'un afficheur LCD préparé avec une carte Arduino : le montage

Le montage consiste à connecter :

- les 2 broches de commande RS et E sur 2 broches numériques Arduino,
- les 4 broches de données D4 à D7 sur 4 broches numériques Arduino,
- le +5V et le 0V

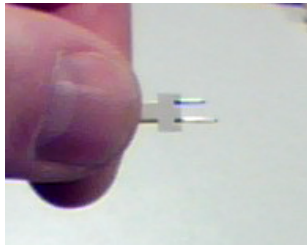


9. Utilisation d'un afficheur LCD préparé avec une carte Arduino : en images

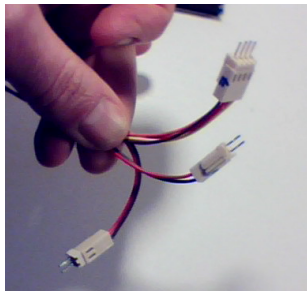
Commencer par prendre les connecteurs droits pour CI :



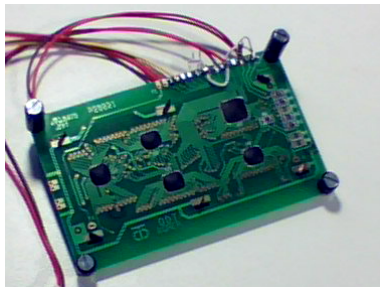
et à l'aide d'une pince, ressortir les broches droites de manière à ce qu'elles soient plus longues en partie extérieure :



Une fois fait pour tous les connecteurs droits, les mettre en place sur les connecteurs femelles sur fils de l'afficheur LCD :

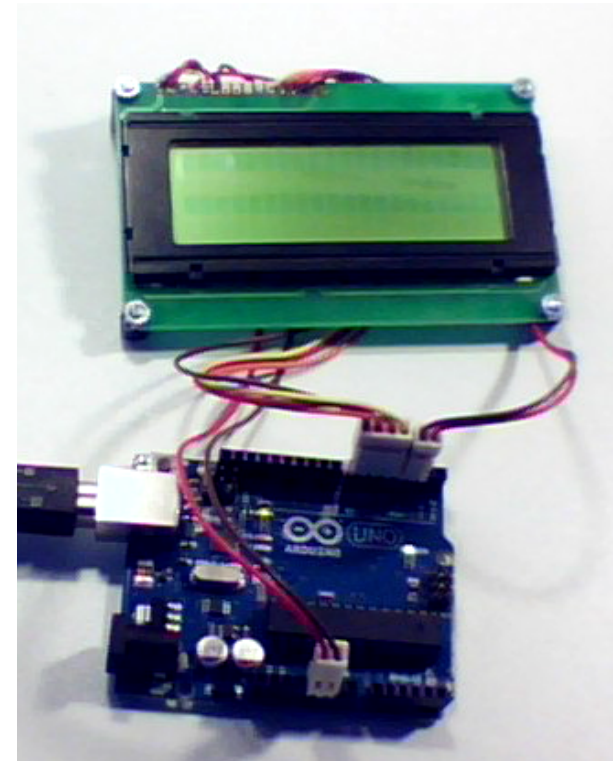


Dernière petite chose : on pourra mettre l'afficheur « sur pieds » à l'aide de 2 entretoises 5mm à l'avant et 2 entretoises 15mm à l'arrière de façon à ce qu'il soit légèrement incliné vers l'avant une fois posé sur ses pieds.



Une fois fait, il devient très facile et pratique d'utiliser l'afficheur LCD avec la carte Arduino :

- enficher le connecteur du +5V et 0V dans le connecteur 0V/+5V de la carte Arduino
- enficher le connecteur RS/E dans le connecteur des broches 2-3 de la carte Arduino,
- enficher le connecteur D4-D7 dans le connecteur des broches 4-7 de la carte Arduino.



Noter qu'un tel circuit pourra facilement être intégré dans un boîtier si l'on souhaite rendre l'application autonome.

10. Rappel : 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.

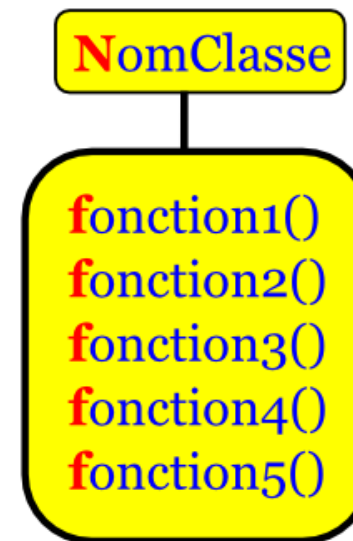
Tout comme une fonction, une classe aura un nom : pour distinguer une classe d'une fonction, **le nom d'une classe commencera par une MAJUSCULE.**

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.



L'appel d'une fonction d'une classe se fait en séparant le nom de la classe et le nom de la fonction par un point

NomClasse.fonction1()

11. Rappel : Langage Arduino : Introduction aux bibliothèques

C'est quoi une bibliothèque Arduino ?

Le langage Arduino comporte de nombreuses instructions comme vous avez pu le constater, une quarantaine en tout. Ces instructions sont intégrées dans ce que l'on appelle le « noyau » ou « cœur » (core en Anglais) du langage Arduino. Ces instructions sont « générales » et servent souvent.

Le langage Arduino peut cependant être étendu à la demande avec des instructions dédiées à certaines applications particulières : afin de ne pas surcharger inutilement le « cœur », ces instructions spécifiques ont été intégrées dans des « paquets d'instructions » appelés bibliothèques.

Comment ça marche ?

Par exemple, si on utilise un afficheur LCD, un servomoteur ou encore si l'on utilise un shield ethernet (réseau), on va intégrer dans notre programme la bibliothèque dédiée correspondante.

Principe général d'utilisation

Pour intégrer une bibliothèque dans un programme Arduino, c'est très simple : il suffit d'ajouter en début de programme une ligne de la forme :

```
#include <nombibliotheque.h> // bibliotheque pour servomoteur
```

ATTENTION : l'instruction include est un peu particulière : la ligne commence par un # et il n'y a pas de point virgule de fin de ligne !

Ensuite, dans le code, au niveau de l'entête déclarative, là où vous déclarez vos variables, il va falloir déclarer un objet (une sorte de super variable) représentant la bibliothèque. Cet objet est en fait une instance (= un exemplaire) d'une Classe (=le moule) qui regroupe les fonctions de la bibliothèque. On a :

```
ClasseObjet monObjet; // declare un objet
```

Généralement ensuite :

- au niveau de la fonction `setup()`, on initialise l'objet avec les paramètres voulus
- au niveau de la fonction `draw()`, on appelle les fonctions de la bibliothèque sous la forme que vous connaissez déjà :

```
monobjet.fonction( param, param, ..);
```

Rappel : pour utiliser une fonction d'une classe du langage Arduino, on utilise le nom de la classe + un point + le nom de la fonction.

Il peut exister des variantes selon les bibliothèques, mais grosso-modo, ça fonctionne de cette façon pour la plupart des bibliothèques Arduino.

Vous avez déjà utilisé une bibliothèque !

Si vous êtes attentifs à tout ce qu'on a déjà vu, vous me direz que ça ressemble étrangement à l'utilisation de la classe **Serial...** et vous aurez raison ! En fait, la classe `Serial` est une bibliothèque qui est intégrée implicitement lorsque vous lancez Arduino : c'est pour ça que vous n'avez pas besoin d'utiliser `#include` pour l'utiliser.

Les bibliothèques standards Arduino

Les bibliothèques Arduino disponibles sont nombreuses, et disposent chacune de quelques fonctions à plusieurs dizaines... ce qui étend considérablement la puissance du langage Arduino et qui en fait aussi tout son intérêt. Voici la liste des bibliothèques standards du langage Arduino ([le logiciel donne la liste...](#)) :

- [La bibliothèque Serial](#) - pour les communications séries entre la carte Arduino et l'ordinateur ou d'autres composants
- [La bibliothèque LCD](#) - pour l'utilisation et le contrôle d'un afficheur LCD alphanumérique standard.
- [La bibliothèque Servo](#) - pour contrôler les servomoteurs.
- [La bibliothèque Stepper](#) - pour contrôler les moteurs pas à pas (nécessite une interface de commande)
- [La bibliothèque Ethernet](#) - pour se connecter à Internet en utilisant le module Arduino Ethernet
- [La bibliothèque EEPROM](#) - référence - pour lire et écrire dans la mémoire EEPROM non volatile.
- [La bibliothèque SD](#) - référence - pour utiliser une carte mémoire SD (utiliser des fichiers, stocker des données, ...)
- [La bibliothèque SoftwareSerial \(Série Logicielle\)](#) - référence - pour communication série logicielle sur n'importe quelles broches de la carte Arduino
- [La bibliothèque Wire / I2C](#) - référence - Interface "deux fils" (TWI/I2C) pour envoyer et recevoir des données sur un réseau de modules ou capteurs.
- [La bibliothèque SPI \(Serial Peripheral Interface\)](#) - pour communication série avec des modules externes supportant le protocole SPI
- [Firmata](#) - pour communiquer avec des applications sur l'ordinateur utilisant un protocole série standard.

En jaune les plus utiles. Impressionnant non ? On les étudiera pas à pas...

Les bibliothèques de la communauté

A côté de ces bibliothèques standards, il existe toute une série de bibliothèques proposées par les uns et les autres et qui concernent des matériels spécifiques, ou autre. Par exemple :

- [La bibliothèque Keypad](#) - pour l'utilisation des claviers matriciels. ([hors référence](#))

Faites un tour ici pour voir ce qui existe : <http://arduino.cc/playground/Main/LibraryList>

12. Langage Arduino : la librairie **LiquidCrystal** pour le contrôle des afficheurs LCD standards

Présentation

- Cette librairie Arduino permet à une carte Arduino de contrôler un afficheur LCD alphanumérique standard à cristaux liquides basé sur le circuit intégré Hitachi HD44780 (ou compatible), ce qui est le cas de la plupart des afficheurs alphanumériques LCD disponibles.
- La librairie fonctionne aussi bien en mode 4 bits qu'en mode 8 bits (càd utilisant 4 ou 8 broches numériques en plus des broches de contrôle RS, Enable et RW (optionnel)). Ainsi, en mode 4 bits, 6 broches numériques de la carte Arduino suffisent pour contrôler un afficheur LCD alphanumérique.

Inclusion

La librairie s'intègre dans un programme avec la ligne (pas de ; !!) :

```
#include <LiquidCrystal.h> // inclut la librairie Servo
```

Le constructeur de la classe

Le constructeur de la classe existe sous 4 formes correspondant à différents brochages possibles. Avec 6 broches, on utilisera la forme suivante :

```
LiquidCrystal lcd(rs, enable, d4, d5, d6, d7); // mode 4 bits - RW non connectée (le plus simple!)
```

avec :

- rs : broche numérique connectée à la broche RS de l'afficheur
- enable : broche numérique connectée à la broche E de l'afficheur
- d4 à d7 : broches numériques connectées aux broches D4 à D7 de l'afficheur.

Les fonctions de la librairie

La librairie dispose de nombreuses fonctions, à savoir :

- Fonctions d'initialisation : [begin\(\)](#)
- Fonctions d'écriture : [print\(\)](#) | [write\(\)](#)
- Fonctions de gestion de l'écran : [clear\(\)](#) | [display\(\)](#) | [noDisplay\(\)](#) |
- Fonctions de positionnement du curseur : [home\(\)](#) | [clear\(\)](#) | [setCursor\(\)](#)
- Fonctions modifiant l'aspect du curseur : [cursor\(\)](#) | [noCursor\(\)](#) | [blink\(\)](#) | [noBlink\(\)](#)
- Fonctions de contrôle du comportement du curseur : [autoscroll\(\)](#) | [noAutoscroll\(\)](#) | [leftToRight\(\)](#) | [rightToLeft\(\)](#)
- Fonctions d'effets visuels : [scrollDisplayLeft\(\)](#) | [scrollDisplayRight\(\)](#)
- Fonction de création de caractère personnalisé : [createChar\(\)](#)

Pour le détail complet des fonctions de la librairie, voir :

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.LibrairieLCD

Principe d'utilisation

La structure type d'un programme utilisant un afficheur LCD va être la suivante :

Au niveau de l'entête déclarative

- Inclusion de la librairie **LiquidCrystal**
- Déclaration des constantes des broches utilisées avec le LCD
- Création d'un objet **LiquidCrystal** que l'on appellera lcd typiquement. On précise à ce niveau les broches utilisées en utilisant les constantes de broches déclarées précédemment.

Truc : je vous conseille de déclarer toutes les broches utilisées pour le LCD avec le nom de leur fonction RS, E, D4, D5.... De cette façon, vous pourrez appeler le constructeur sous la forme lcd(RS,E,D4,D5,D6,D7) ;

Au niveau de la fonction **setup()**

- Initialisation de l'afficheur avec la fonction lcd.begin(colonnes, lignes) où colonnes et lignes sont le nombre de ligne et de colonne de l'afficheur.
- Prendre l'habitude d'initialiser l'affichage avec l'appel de la fonction lcd.clear()
- On peut également à ce niveau afficher un rapide message d'accueil avec la fonction lcd.print(« texte ») suivi d'un effacement du LCD avec la fonction lcd.clear()
- On peut également à ce niveau réaliser l'affichage des messages fixes qui ne changeront pas ensuite (nom des valeurs par exemple)

Au niveau de la fonction **loop()**

- A ce niveau on utilisera selon les besoins toutes les fonctions utiles de la librairie pour se déplacer sur l'afficheur, afficher des caractères, effacer des messages, etc...



13. « Hello world » : afficher votre premier message sur un afficheur LCD.

Dans ce premier programme, nous allons tout simplement afficher le message « Hello World » sur l'afficheur LCD : rien de bien compliqué, mais ça sera votre premier programme utilisant LCD !

Entête déclarative

- On commence par importer la librairie **LiquidCrystal**
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet **LiquidCrystal** qui représentera le LCD dans le reste du programme.

Fonction **setup()**

- On commence par initialiser l'afficheur à l'aide de la fonction **begin**(colonnes,lignes). Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction **clear**() qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) c-à-d dans le coin supérieur gauche de l'écran.
- Puis on affiche tout simplement un message à l'aide de la fonction... **print**(« texte ») qui affiche le texte à l'emplacement du curseur !

Note : Remarquer au passage la cohérence du langage Arduino qui propose la fonction **print()** aussi bien pour la classe **Serial** que la classe **LiquidCrystal**. Cette fonction sera également disponible pour les classes gérant une carte mémoire SD ou encore le réseau ethernet. Facile et puissant !

Fonction **loop()**

- Laissée vide ici.

Fonctionnement

Une fois la carte Arduino programmée, le message s'affiche : cool !



```
// ateliers Arduino par X. HINAULT - Tous droits réservés - 2012
// licence GPL v3 - www.mon-club-elec.fr

// affiche message sur afficheur LCD standard

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

// déclaration des broches de l'afficheur
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant lcd

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    // écrire ici les instructions à exécuter au début
    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    lcd.print("Hello World"); // affiche le message

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    // écrire ici les instructions à exécuter en boucle

} // fin de la fonction loop()

// NB : les lignes précédées de // sont des commentaires
```


14. Les fonctions de gestion de l'écran : `clear()` | `display()` | `noDisplay()`

A présent, nous allons passer en revue les différentes fonctions de la librairie Arduino `LiquidCrystal` afin de savoir comment les utiliser. Rien de très compliqué. Les trois premières fonctions que nous allons voir ne reçoivent aucun paramètre et servent à gérer l'écran :

- la fonction `clear()` efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) c'est à dire dans le coin supérieur gauche de l'écran. **Mettre une courte pause** `delay(10)` **après** `clear()` **améliore le résultat et évite les affichages inattendus.**
- la fonction `display()` rallume l'écran de l'afficheur LCD, après qu'il ait été éteint avec l'instruction `noDisplay()`. Ceci réactive à l'identique le texte et le curseur de l'affichage tels qu'ils étaient.
- la fonction `noDisplay()` éteint l'écran de l'afficheur LCD, sans perdre le texte actuellement affiché (autrement dit, l'afficheur lui-même n'est pas mis hors tension, seul l'écran est éteint).

Voici un programme pour tester ces fonctions :

Entête déclarative

- On commence par importer la librairie `LiquidCrystal`
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet `LiquidCrystal` qui représentera le LCD dans le reste du programme.

Fonction `setup()`

- On commence par initialiser l'afficheur à l'aide de la fonction `begin(colonnes,lignes)`. Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction `clear()` qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) c'est à dire dans le coin supérieur gauche de l'écran.
- Puis on affiche tout simplement un message à l'aide de la fonction... `print(« Texte »)` !

Fonction `loop()`

- On éteint l'écran avec la fonction `noDisplay()` suivi d'une pause `delay()`
- On rallume l'écran avec la fonction `display()` suivi d'une pause `delay()`
- Au final, le message clignote.

Fonctionnement

- Une fois la carte Arduino programmée, le message clignote !

```
// faire clignoter affichage sur afficheur LCD

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

//-- déclaration des broches de l'afficheurs --
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant lcd

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    lcd.print("Display/noDisplay"); // affiche le message

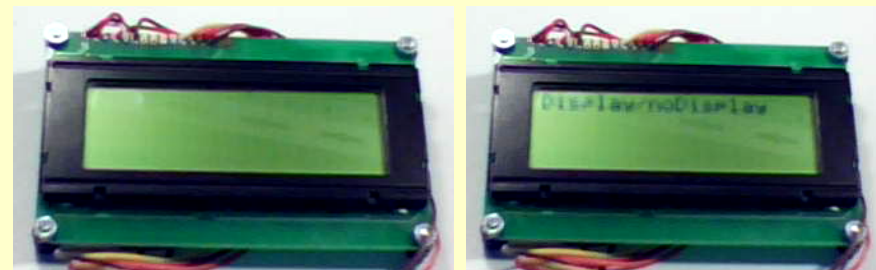
} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    lcd.display(); // active affichage LCD (sans modifier message)
    delay(1000); // pause

    lcd.noDisplay(); // désactive affichage LCD (sans modifier message)
    delay(1000); // pause

} // fin de la fonction loop()
```



15. Les fonctions modifiant l'aspect du curseur : `cursor()` | `noCursor()` | `blink()` | `noBlink()`

Le curseur correspond à la position courante sur l'afficheur LCD où sera écrit le prochain caractère. Par défaut, le curseur est invisible mais il peut être visible sous la forme d'un _ ou d'un rectangle noir clignotant. Quatre fonctions, qui ne reçoivent également aucun paramètre, permettent de gérer l'aspect du curseur :

- les fonctions `cursor()` et `noCursor()` permettent respectivement d'activer / désactiver le trait sous le curseur
- les fonctions `blink()` et `noBlink()` permettent respectivement d'activer / désactiver le rectangle clignotant à l'emplacement du curseur.
- Noter que les 2 peuvent être combinés entre eux

Voici un programme pour tester ces fonctions :

Entête déclarative

- On commence par importer la librairie `LiquidCrystal`
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet `LiquidCrystal` qui représentera le LCD dans le reste du programme.

```
// modifie l'aspect du curseur d'un afficheur LCD

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

//-- déclaration des broches de l'afficheurs --
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant
lcd
```

Fonction `setup()`

- On commence par initialiser l'afficheur à l'aide de la fonction `begin(colonnes,lignes)`. Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction `clear()` qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) càd dans le coin supérieur gauche de l'écran. A noter que `clear()` est suivie de la fonction `delay(10)`.
- Puis on affiche un message de test pendant 2 secondes avec la fonction `print(« Texte»)` suivi de la fonction `delay()`.

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    lcd.print("LCD OK !"); // affiche le message
    delay(1000); // pause

} // fin de la fonction setup()
```

Fonction loop()

- On teste les différentes possibilités avec un délai de quelques secondes entre chaque changement. Un message s'affiche pour indiquer la fonction testée.
- Chaque fonction est appelée sous la forme `lcd.fonction()` où `lcd` est un objet `LiquidDisplay` déclaré au préalable avec le constructeur.
- Noter la courte pause `delay(10)` mise après chaque appel de la fonction `clear()` pour éviter les problèmes d'affichages lors de l'enchaînement des fonctions.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin  
void loop() {
```

```
  lcd.clear(); // efface LCD + se place en 0,0  
  delay(10); // courte pause après clear()  
  lcd.noCursor(); // désactive trait de base  
  lcd.noBlink(); // désactive clignotement  
  lcd.print("noCursor+noBlink"); // affiche le message  
  delay(4000); // pause
```

```
  lcd.clear(); // efface LCD + se place en 0,0  
  delay(10); // courte pause après clear()  
  lcd.cursor(); // active trait de base  
  lcd.print("Cursor seul"); // affiche le message  
  delay(4000); // pause
```

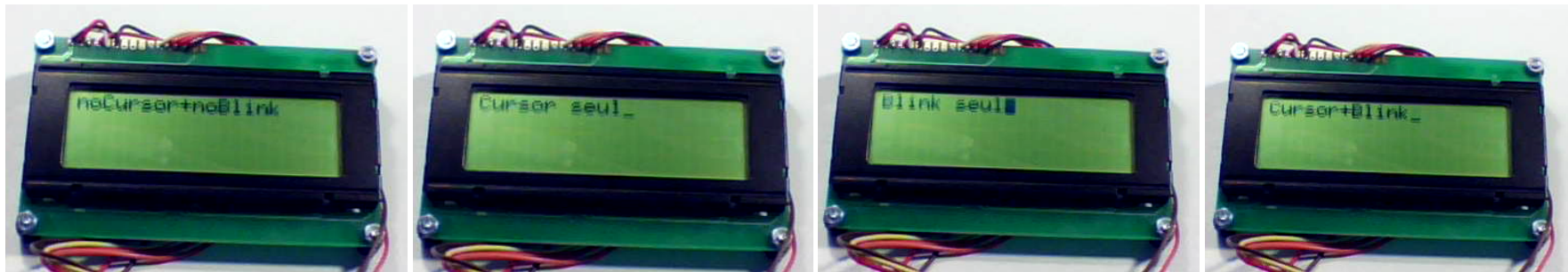
```
  lcd.clear(); // efface LCD + se place en 0,0  
  delay(10); // courte pause après clear()  
  lcd.noCursor(); // désactive trait de base  
  lcd.blink(); // active clignotement  
  lcd.print("Blink seul"); // affiche le message  
  delay(4000); // pause
```

```
  lcd.clear(); // efface LCD + se place en 0,0  
  delay(10); // courte pause après clear()  
  lcd.cursor(); // active trait de base  
  lcd.blink(); // active clignotement  
  lcd.print("Cursor+Blink"); // affiche le message  
  delay(4000); // pause
```

```
} // fin de la fonction loop()
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, les différents messages s'affichent et l'aspect du curseur se modifie en conséquence.



Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

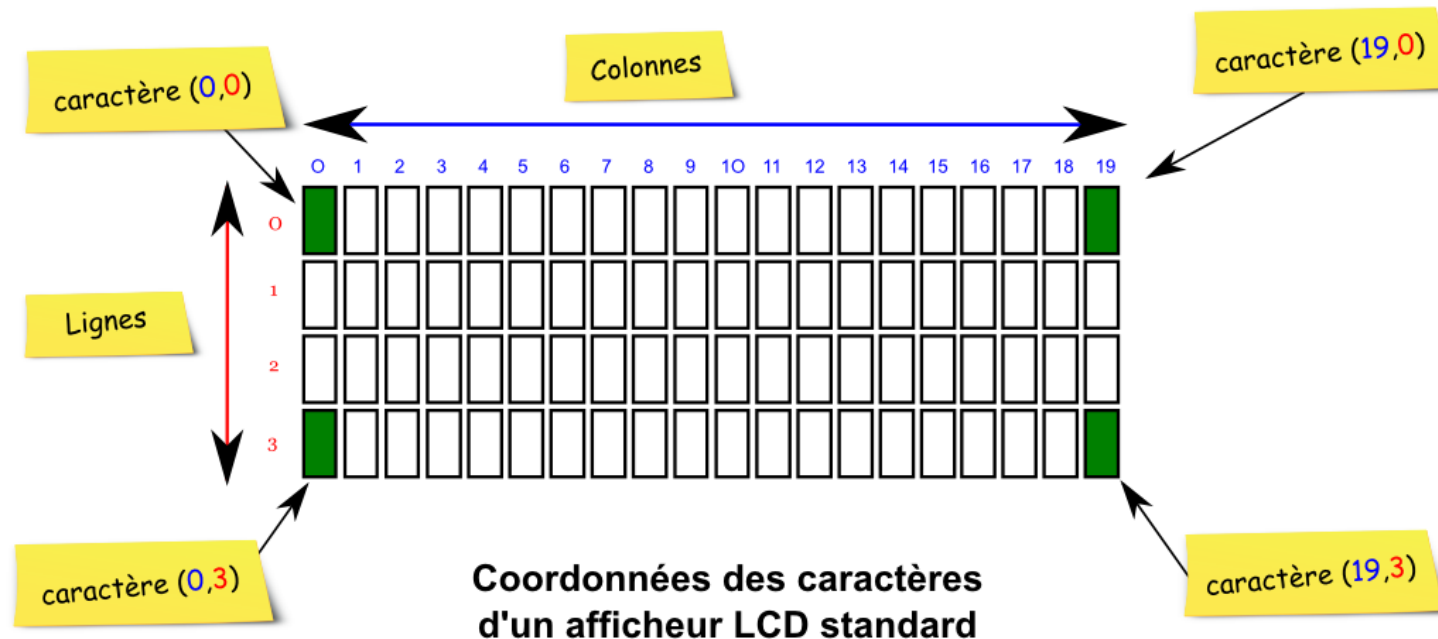
Atelier Arduino : Apprendre à utiliser un afficheur LCD alpha-numérique avec Arduino et découvrir la librairie Arduino LiquidCrystal.

16. Le principe de positionnement sur l'afficheur LCD standard

- Par la suite, nous allons voir les fonctions permettant de se positionner à un emplacement voulu sur l'afficheur. Ceci est rendu possible grâce à une numérotation des caractères sous la forme de coordonnées (colonne,ligne) un peu à la façon (x,y) des points d'un graphique ou encore d'une bataille navale (« colonne A, ligne 2 »).
- Les colonnes sont numérotées de 0 à x-1 pour un afficheur de x colonnes. et les lignes sont numérotées de 0 à y-1 pour un afficheur de y lignes. Par exemple, dans le cas d'un afficheur de 20 colonnes x 4 lignes, les colonnes seront numérotées de 0 à 19 et les lignes de 0 à 3.

Le point important : la première ligne et la première colonne ont le numéro 0

- Voici le schéma résumant ceci :



A RETENIR :

L'instruction `clear()` positionne le curseur en (0,0) c'est à dire dans le coin supérieur gauche de l'afficheur (1er caractère de la 1ère ligne)
Lorsque l'on affiche un caractère avec l'instruction `print()`, ceci a pour effet de décaler la position du curseur d'un cran vers la droite

17. Fonctions de positionnement du curseur : `home()` | `clear()` | `setCursor()`

Le curseur, comme on l'a déjà dit, correspond à la position courante sur l'afficheur LCD où sera écrit le prochain caractère. Par défaut, le curseur est invisible mais il peut être visible sous la forme d'un `_` (fonctions `cursor()` et `noCursor()`) ou d'un rectangle noir clignotant (fonctions `blink()` et `noBlink()`). Toutes ces fonctions n'ont pas d'impact sur la position courante du curseur. A présent, nous allons voir les fonctions qui ont un impact sur la position du curseur et qui permettent de contrôler cette position.

Deux fonctions qui ne reçoivent aucun paramètre, placent le curseur à « l'origine », c'est à dire en (0,0) :

- la fonction `home()` : replace le curseur en (0,0) SANS modifier l'affichage actuel
- la fonction `clear()` : que nous avons déjà vue, replace le curseur en (0,0) ET efface l'affichage actuel. Cette fonction peut prendre un certain temps et il semble nécessaire de la faire suivre d'une pause courte (`delay(10)`) pour éviter des comportements inattendus de l'afficheur.

Une fonction spécialement dédiée pour le positionnement du curseur :

- la fonction `setCursor(colonne,ligne)` : place le curseur en position (colonne,ligne). Ne pas oublier que la première colonne a le numéro 0 (et non 1), de même que la première ligne. Cette fonction est très très utile pour faire exactement ce que vous voulez avec votre afficheur LCD et devra idéalement être utilisée avant tout appel de la fonction `print(« texte »)`

A part, la fonction `print()` :

- la fonction `print(« texte »)` : cette fonction décale le curseur du nombre de caractère de la chaîne « texte » utilisée.

Bon à savoir en ce qui concerne la fonction `print()` :

une fois arrivé en fin de ligne, le caractère suivant le dernier de la première ligne est le 1er de la 3ème ligne et le caractère suivant le dernier de la 2ème ligne est le premier de la 4ème ligne. Ceci est dû à la structure interne de la mémoire de l'afficheur.

Voici un programme pour tester ces fonctions :

Entête déclarative

- On commence par importer la librairie `LiquidCrystal`
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet `LiquidCrystal` qui représentera le LCD dans le reste du programme.

```
// ateliers Arduino par X. HINAULT - Tous droits réservés - 2012
// licence GPL v3 - www.mon-club-elec.fr

// positionnement du curseur d'un afficheur LCD

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

//-- déclaration des broches de l'afficheurs --
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant
lcd
```

Fonction `setup()`

- On commence par initialiser l'afficheur à l'aide de la fonction `begin(colonnes,lignes)`. Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction `clear()` qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) càd dans le coin supérieur gauche de l'écran. A noter que `clear()` est suivie de la fonction `delay(10)`.
- Puis on affiche un message de test pendant 2 secondes avec la fonction `print(« Texte »)` suivi de la fonction `delay()`.
- On initialise à nouveau l'affichage avec la fonction `clear()`, suivie de la fonction `delay(10)`.

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    lcd.print("LCD OK !"); // affiche le message
    delay(1000); // pause

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

} // fin de la fonction setup()
```


Fonction loop()

- On teste les différentes possibilités avec un délai de quelques secondes entre chaque changement. Un message s'affiche pour indiquer la fonction testée.
- Chaque fonction est appelée sous la forme `lcd.fonction()` où `lcd` est un objet **LiquidDisplay** déclaré au préalable avec le constructeur.
- On teste ici successivement :
 - le décalage du curseur avec la fonction `print(« texte »)`
 - le retour en (0,0) avec la fonction `home()`
 - le positionnement à l'emplacement voulu avec `setCursor(colonne, ligne)`
 - l'effacement et le retour en (0,0) avec la fonction `clear()`

Truc pratique :

Si vous n'êtes pas à l'aise avec la numérotation des lignes/colonnes à partir de 0, vous pouvez utiliser la fonction `setCursor()` sous la forme :

`setCursor(x-1, y-1);`

où x et y représente la x-ième colonne et y-ième ligne.

Ainsi, pour se positionner en 5ème colonne, 3ème ligne on fera :

`setCursor(5-1, 3-1);`

C'est plus intuitif. Une autre solution passe par la création d'une fonction.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    lcd.blink(); // active clignotement

    lcd.print("decalage texte"); // affichage chaine décale le curseur
    delay(4000); // pause

    lcd.home(); // retour en 0,0 sans modifier affichage
    lcd.setCursor(1-1, 2-1); // se place en 1ère colonne 2ème ligne (-1
pour corriger indice)
    lcd.print("home");
    lcd.home(); // retour en 0,0 sans modifier affichage
    delay(4000); // pause

    lcd.setCursor(10, 1); // se place en 11ème colonne (indice=10), 2ème
ligne (indice=1)
    lcd.print("ici");
    delay(4000);

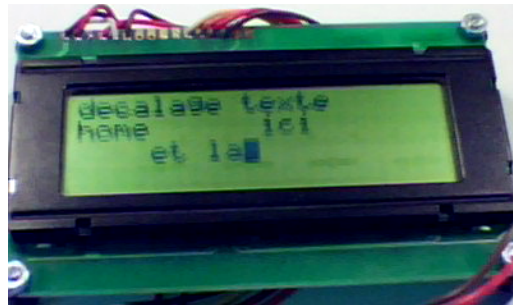
    lcd.setCursor(5-1, 3-1); // se place en 5ème colonne 3ème ligne (-1
pour corriger indice)
    lcd.print("et la");
    delay(4000);

    lcd.clear(); // efface écran et position en 0,0
    delay(4000);

} // fin de la fonction loop()
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, les différents messages s'affichent aux positions voulues :



18. Fonctions de contrôle du comportement du curseur : `autoscroll()` | `noAutoscroll()` | `leftToRight()` | `rightToLeft()`

A présent, nous allons aborder les fonctions de la librairie Arduino LiquidCrystal qui permettent de contrôler le comportement du curseur. Plusieurs fonctions, qui ne reçoivent aucun paramètre, permettent de contrôler le comportement du curseur lors de l'affichage de texte. **Par défaut, au démarrage, le texte s'affiche de la gauche vers la droite et chaque nouveau caractère entraîne un décalage du curseur d'un cran vers la droite.** Ce comportement de base peut être modifié à loisir avec les fonctions suivantes (pas forcément très utile, mais comme ça vous savez que ça existe...) :

- la fonction `autoscroll()` active le **décalage du texte** (le curseur ne bouge pas)
- la fonction `noAutoscroll()` **décalage du curseur** d'un caractère lors de l'écriture d'un nouveau caractère (le texte reste fixe) (**Mode par défaut**),
- la fonction `leftToRight()` active le décalage de gauche à droite (c'est le décalage actif par défaut) (**Mode par défaut**),
- la fonction `rightToLeft()` active le décalage de droite à gauche.

Voici un programme pour tester ces fonctions :

Entête déclarative

- On commence par importer la librairie `LiquidCrystal`
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet `LiquidCrystal` qui représentera le LCD dans le reste du programme.

```
// ateliers Arduino par X. HINAULT - Tous droits réservés - 2012
// licence GPL v3 - www.mon-club-elec.fr

// controler le comportement du curseur d'un afficheur LCD

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

//-- déclaration des broches de l'afficheurs --
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant
lcd
```

Fonction `setup()`

- On commence par initialiser l'afficheur à l'aide de la fonction `begin(colonnes,lignes)`. Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction `clear()` qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) càd dans le coin supérieur gauche de l'écran. A noter que `clear()` est suivie de la fonction `delay(10)`.
- Puis on affiche un message de test pendant 1 seconde avec la fonction `print(« Texte »)` suivi de la fonction `delay()`.
- On initialise à nouveau l'affichage avec la fonction `clear()`, suivie de la fonction `delay(10)`.
- On active le trait sous le curseur avec la fonction `cursor()`

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    lcd.print("LCD OK !"); // affiche le message
    delay(1000); // pause

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    lcd.cursor(); // active curseur
} // fin de la fonction setup()
```

Fonction `loop()`

- On teste les différentes possibilités avec un délai de quelques secondes entre chaque changement. Un message s'affiche pour indiquer la fonction testée.
- Chaque fonction est appelée sous la forme `lcd.fonction()` où `lcd` est un objet **LiquidDisplay** déclaré au préalable avec le constructeur.
- On teste ici successivement :
 - le comportement par défaut (gauche vers droite et texte fixe)
 - le comportement droite vers gauche avec texte fixe
 - le comportement gauche vers droite avec curseur fixe.
- A noter que l'on appelle ici la fonction `defileChaine(« texte »)` qui permet d'afficher les caractères 1 à 1 pour bien visualiser l'effet.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()
    lcd.leftToRight(); // active le décalage gauche vers droite - par
défaut
    lcd.noAutoscroll(); // active le décalage du curseur (texte reste
fixe) - mode par défaut

    lcd.setCursor(10,0); // positionne curseur en (index colonne , index
ligne)
    defileChaine("Par défaut");

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()
    lcd.rightToLeft(); // active le décalage droit vers gauche
    lcd.setCursor(18,0); // positionne curseur en (index colonne , index
ligne)
    defileChaine("rightToLeft");

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()
    lcd.leftToRight(); // active le décalage gauche vers droite - mode
par défaut
    lcd.autoscroll(); // active le décalage du texte (curseur reste
fixe)
    lcd.setCursor(15,0); // positionne curseur en (index colonne , index
ligne)
    defileChaine("autoscroll");

} // fin de la fonction loop()
```

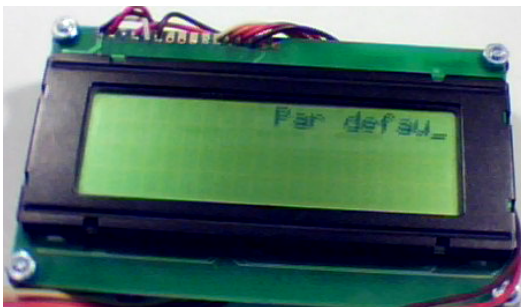
Fonction defileChaine()

- Cette fonction affiche 1 à 1 les caractères d'une chaîne et repose sur l'objet Arduino **String** déjà présenté par ailleurs.
- Cette fonction reçoit un objet **String** en paramètre : la chaîne de caractères à afficher
- Cette fonction ne renvoie rien = type **void**
- Cette fonction réalise une boucle **for** pour défile les caractères de l'objet String :
 - on extrait chaque caractère avec la fonction **charAt(index)** de la classe String
 - on affiche le caractère avec la fonction **print()**
- On réalise une pause entre chaque affichage de caractère avec la fonction **delay()**

```
void defileChaine(String chaineIn) {  
    for(int i=0; i<chaineIn.length(); i++) { // défile les caractères de la chaîne  
        lcd.print(chaineIn.charAt(i)); // affiche le caractère i de la chaîne  
        delay(1000); // 1 seconde entre chaque  
    } // fin for  
} // fin fonction defile chaîne
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, les différents messages s'affichent avec des comportements du curseur modifiés en conséquence :



Cette fois, vous êtes en passe de devenir maître dans l'art d'utiliser un afficheur LCD !
Il nous reste encore quelques fonctions à voir, et vous serez capable de faire tout ce que vous voulez avec votre afficheur !

19. Fonctions d'effets visuels : `scrollDisplayLeft()` | `scrollDisplayRight()`

La librairie Arduino LiquidCrystal offre également 2 fonctions (qui ne reçoivent aucun paramètre) qui permettent de décaler intégralement l'ensemble de l'affichage courant, ce qui permet certains effets visuels potentiellement intéressants :

- la fonction `scrollDisplayLeft()` qui décale l'ensemble de l'affichage vers la gauche
- la fonction `scrollDisplayRight()` qui défile l'ensemble de l'affichage vers la droite

Voici un programme pour tester ces fonctions :

Entête déclarative

- On commence par importer la librairie `LiquidCrystal`
- Ensuite on déclare l'ensemble des 6 broches utilisées pour le contrôle de l'afficheur LCD
- On déclare un objet `LiquidCrystal` qui représentera le LCD dans le reste du programme.

```
// ateliers Arduino par X. HINAULT - Tous droits réservés - 2012
// licence GPL v3 - www.mon-club-elec.fr

// décalage latéral de l'affichage entier d'un afficheur LCD

//--- entete déclarative ---
#include <LiquidCrystal.h> // inclusion de la librairie LCD

//-- déclaration des broches de l'afficheurs --
const int RS=2; // broche RS
const int E=3; // broche E
const int D4=4; // broche D4
const int D5=5; // broche D5
const int D6=6; // broche D6
const int D7=7; // broche D7

LiquidCrystal lcd(RS,E,D4,D5,D6,D7); // déclaration objet représentant
lcd
```


Fonction `setup()`

- On commence par initialiser l'afficheur à l'aide de la fonction `begin(colonnes,lignes)`. Ici, afficheur 20 colonnes x 4 lignes. A adapter à votre situation le cas échéant.
- On initialise l'affichage avec la fonction `clear()` qui efface l'écran et positionne le curseur (invisible par défaut) en 0,0 (colonne 0, ligne 0) càd dans le coin supérieur gauche de l'écran. A noter que `clear()` est suivie de la fonction `delay(10)`.
- Puis on affiche un message de test pendant 1 seconde avec la fonction `print(« Texte »)` suivi de la fonction `delay()`.
- On initialise à nouveau l'affichage avec la fonction `clear()`, suivie de la fonction `delay(10)`.
- On affiche un message central sur 2 lignes.

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    lcd.begin(20,4); // initialise LCD colonnes x lignes

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    lcd.print("LCD OK !"); // affiche le message
    delay(1000); // pause

    lcd.clear(); // efface LCD + se place en 0,0
    delay(10); // courte pause après clear()

    //----- message initial ----
    lcd.setCursor(4,0); // positionne curseur
    lcd.print("**Ateliers**"); // affiche message
    lcd.setCursor(4,1); // positionne curseur
    lcd.print("**Arduino**"); // affiche message

} // fin de la fonction setup()
```

Fonction loop()

- On teste les différentes possibilités avec un délai de quelques secondes entre chaque changement. Un message s'affiche pour indiquer la fonction testée.
- Chaque fonction est appelée sous la forme `lcd.fonction()` où `lcd` est un objet `LiquidDisplay` déclaré au préalable avec le constructeur.
- On réalise un défilement du message de droite à gauche jusqu'aux bords du LCD à l'aide de boucles `for()` et des fonctions `scrollDisplayLeft()` et `scrollDisplayRight()` ce qui réalise un effet de va-et-vient latéral. La vitesse est modifiable à l'aide de la pause `delay()`.

Ce type d'effet sera typiquement utilisé pour des messages de veille ou d'attente de votre future application utilisant un LCD. Comme un pro !



« Faire les choses sérieusement sans se prendre au sérieux ! »

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    for (int i=0; i<4; i++) {
        lcd.scrollDisplayLeft(); // décalage affichage entier vers la
gauche
        delay(200);
    }

    for (int i=0; i<8; i++) {
        lcd.scrollDisplayRight(); // décalage affichage entier vers la
gauche
        delay(200);
    }

    for (int i=0; i<4; i++) {
        lcd.scrollDisplayLeft(); // décalage affichage entier vers la
gauche
        delay(200);
    }

} // fin de la fonction loop()
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, le message se déplace de droite à gauche et vice-versa :



20. Pour info : caractères spéciaux et personnalisés

La librairie LiquidCrystal propose par ailleurs 2 fonctions potentiellement utiles pour afficher des caractères spéciaux et créer des caractères personnalisés :

- Tout d'abord, la fonction `write(codeChar)` qui permet d'afficher le caractère correspondant à `codeChar`
- Ensuite la fonction `createChar(codeChar, arrayPixels)` qui permet de créer un caractère personnalisé :
 - ayant le numéro `codeChar`
 - et dont l'aspect est défini par un tableau de pixels défini sous la forme :

```
//définition d'un caractère sous la forme d'un tableau de 5 colonnes x 8 lignes de pixels
byte coeur[8] = { // tableau de byte = valeur non signée 0-255 sur 8 bits
  B00000, // format binaire : chaque pixel est représenté par 0 ou 1
  B01010, // si pixel =1 : il sera allumé si pixel=0 : sera laissé éteint
  B11111,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000 // dernière ligne laissée vide
};
```



Exemple de caractères 5x7

Chaque pixel mis à 1 est « allumé » et les pixels laissés à 0 restent éteint

L'usage de ces fonctions est cependant plus ou moins capricieux, aussi nous n'en dirons pas plus ici.
Pour plus de détails, se reporter à l'exemple fourni avec le logiciel Arduino : [Exemples > LiquidCrystal > CustomCharacter](#)



Cette fois, ça y est : vous savez (presque) tout ce qu'il faut savoir pour utiliser un afficheur LCD standard avec Arduino ! Cool non ?

21. Les éléments du langage Arduino étudiés dans cet atelier

Fonctions de la librairie **LiquidCrystal**

Dans cet atelier, les fonctions de la librairie **LiquidCrystal** ont été abordées :

- Fonctions d'initialisation : [begin\(\)](#)
- Fonctions d'écriture : [print\(\)](#) | [write\(\)](#)
- Fonctions de gestion de l'écran : [clear\(\)](#) | [display\(\)](#) | [noDisplay\(\)](#) |
- Fonctions de positionnement du curseur : [home\(\)](#) | [clear\(\)](#) | [setCursor\(\)](#)
- Fonctions modifiant l'aspect du curseur : [cursor\(\)](#) | [noCursor\(\)](#) | [blink\(\)](#) | [noBlink\(\)](#)
- Fonctions de contrôle du comportement du curseur : [autoscroll\(\)](#) | [noAutoscroll\(\)](#) | [leftToRight\(\)](#) | [rightToLeft\(\)](#)
- Fonctions d'effets visuels : [scrollDisplayLeft\(\)](#) | [scrollDisplayRight\(\)](#)
- Fonction de création de caractère personnalisé : [createChar\(\)](#)

La documentation de la librairie **LiquidCrystal** en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.LibrairieLCD

22. *A présent, vous devriez être capable :*

- d'écrire un programme utilisant un afficheur LCD alpha-numérique standard et d'afficher des messages.

Table des matières

Intro |
Matériel nécessaire pour les ateliers Arduino |
Matériel spécifique nécessaire pour cet atelier |
Fiche Technique : Afficheur LCD alpha-numérique standard |
Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : le schéma théorique |
Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : description concrète |
Schéma électrique type d'utilisation d'un afficheur LCD avec une carte Arduino |
Utilisation d'un afficheur LCD préparé avec une carte Arduino : le montage |
Utilisation d'un afficheur LCD préparé avec une carte Arduino : en images |
Rappel : Notion de « Classe » |
Rappel : Langage Arduino : Introduction aux librairies |
Langage Arduino : la librairie LiquidCrystal pour le contrôle des afficheurs LCD standards |
« Hello world » : afficher votre premier message sur un afficheur LCD. |
Les fonctions de gestion de l'écran : clear() | display() | noDisplay() |
Les fonctions modifiant l'aspect du curseur : cursor() | noCursor() | blink() | noBlink() |
Le principe de positionnement sur l'afficheur LCD standard |
Fonctions de positionnement du curseur : home() | clear() | setCursor() |
Fonctions de contrôle du comportement du curseur : autoscroll() | noAutoscroll() | leftToRight() | rightToLeft() |
Fonctions d'effets visuels : scrollDisplayLeft() | scrollDisplayRight() |
Pour info : caractères spéciaux et personnalisés |
Les éléments du langage Arduino étudiés dans cet atelier |
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