1. 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 :

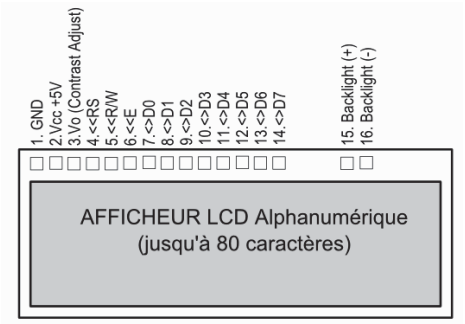
- réflectif ou rétro-éclairé (consomme plus)
- avec LED de rétro-éclairage (utilisable dans l'obscurité)
- couleur de l'affichage : soit noir sur fond vert classique, soit blanc sur fond bleu, etc...

Il existe différentes tailles également :

- en 4 lignes x 20 colonnes, en 2 lignes x 8 colonnes, en 1 lignes x 16 colonnes.

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 !

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

Mode de fonctionnement

Un afficheur LCD alpha-numérique standard peut fonctionner :

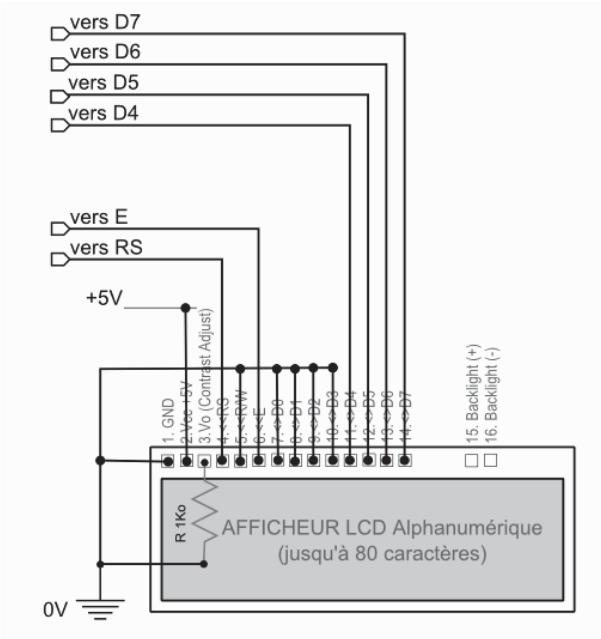
- soit en mode dits « 8 bits » et dans ce cas nécessite 8 broches de données + 3 broches de commandes soit 11 broches !
- soit en mode dits « 4bits » et dans ce cas nécessite 4 broches de données + 2 broches de commandes soit seulement **6 broches**. **C'est ce mode de fonctionnement que nous allons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

2. 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).

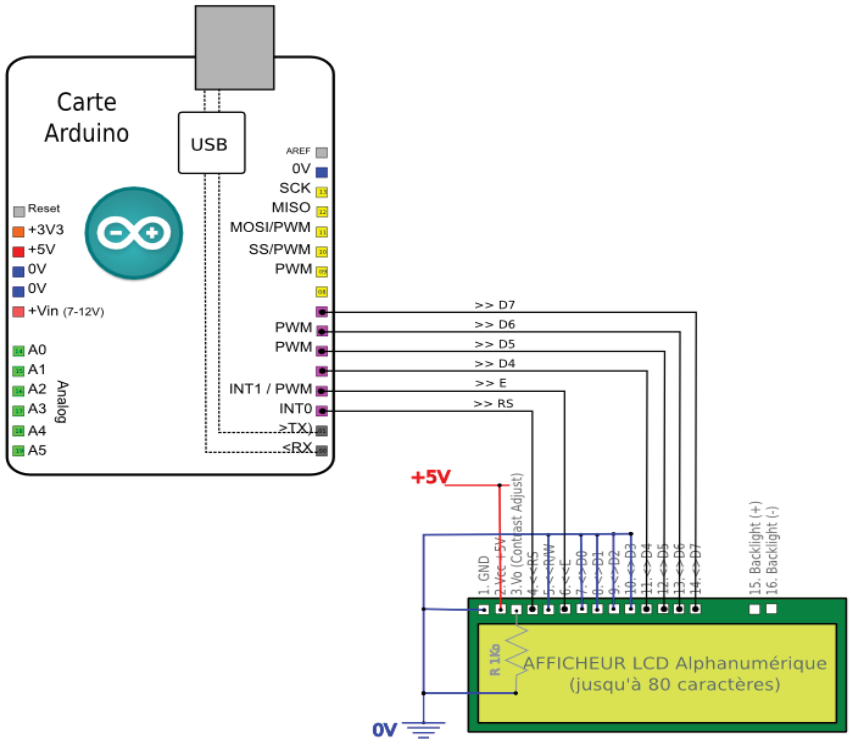
Voici le schéma de cette connexion simplifiée de l'afficheur LCD alpha-numérique :



3. 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



4. 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.

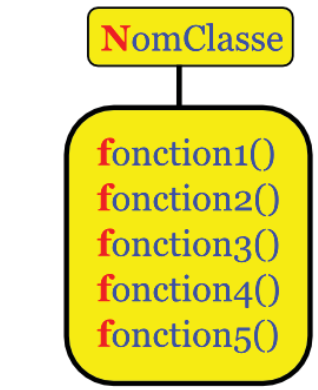
Tout comme une fonction, une classe aura un nom : pour distinguer une classe d'une fonction, **le nom d'une classe commencera par une MAJUSCULE**.

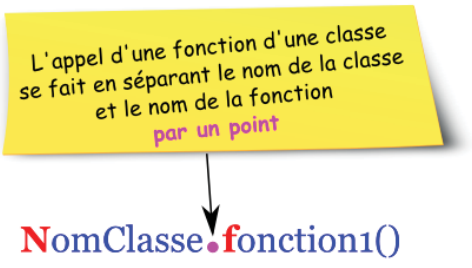
En pratique, lorsque l'on programme en langage Arduino, on n'a pas besoin de créer de classes (ouf !). Mais le langage Arduino ou ses librairies comporte plusieurs classes et il faut donc comprendre ce concept :

- 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 coeur (ou core) du langage Arduino.





5. Rappel : Langage Arduino : Introduction aux librairies

C'est quoi une librairie 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 « coeur » (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 « coeur », ces instructions spécifiques ont été intégrées dans des « paquets d'instructions » appelés librairies.

Comment çà 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 librairie dédiée correspondante.

Principe général d'utilisation

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

```
#include <nomlibrairie.h> // librairie 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 librairie. Cet objet est en fait une instance (= un exemplaire) d'une Classe (=le moule) qui regroupe les fonctions de la librairie. On a :

```
ClasseObjet monObjet; // déclare 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 librairie 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 librairies, mais grosso-modo, çà fonctionne de cette façon pour la plupart des librairies Arduino.

6. 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 connectée aux broches D4àD7 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 librairies pour se déplacer sur l'afficheur, afficher des caractères, effacer des messages, etc...

7. « 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 çà 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) càd 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
```

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()
```