

# Apprendre à utiliser des boutons poussoirs avec un afficheur LCD alpha-numérique et Arduino.



## Ateliers Arduino

par X. HINAULT

[www.mon-club-elec.fr](http://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](mailto: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](mailto: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](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 !**

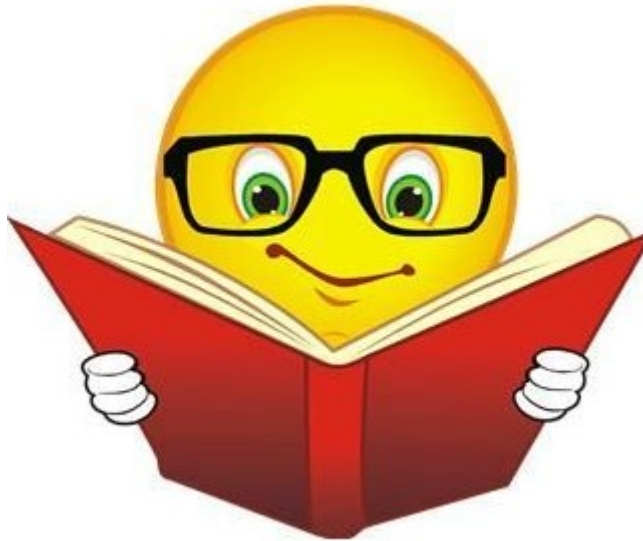
**Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !**

## 1. *Intro*

L'objectif ici est :

- d'utiliser les boutons poussoirs pour contrôler un afficheur LCD
- d'apprendre à afficher des grandeurs numériques dépendantes de l'appui sur des boutons poussoirs

... afin d'être en mesure de réaliser des applications utilisant des boutons poussoir et un afficheur LCD.

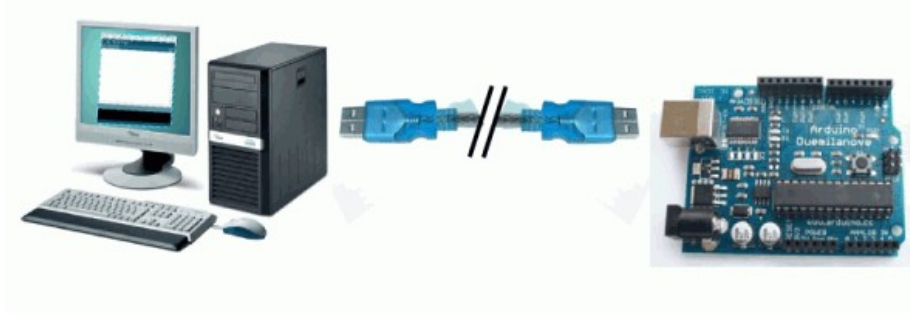


**Prêt ? C'est parti !**

## 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

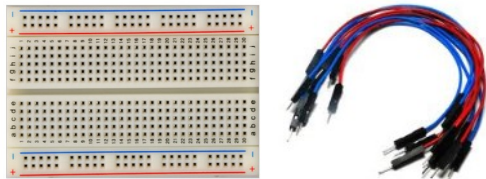


L'espace de développement Arduino associe :

- 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

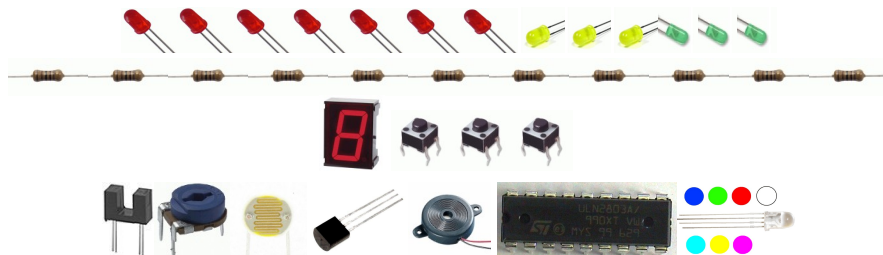


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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](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**

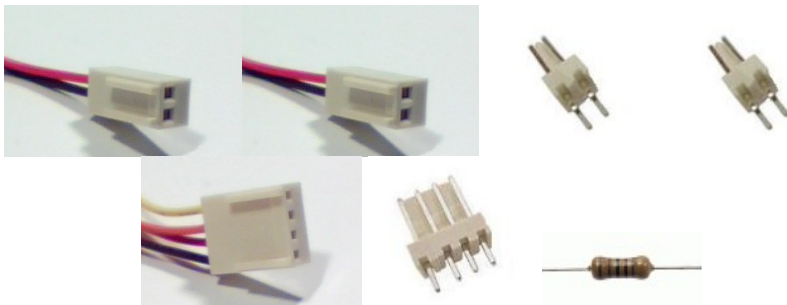


L'afficheur LCD alphanumérique standard 4 lignes x 20 colonnes est facile à utiliser avec la carte Arduino à l'aide de 6 broches numériques. Cet afficheur permet d'afficher des lettres et des chiffres sur 4 lignes et 20 colonnes. Un équipement idéal pour créer des applications autonomes réalisant des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

**La préparation de l'afficheur LCD est présentée dans l'atelier précédent consacré aux afficheurs LCD alpha-numériques standards**

## 4. Rappel : 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.

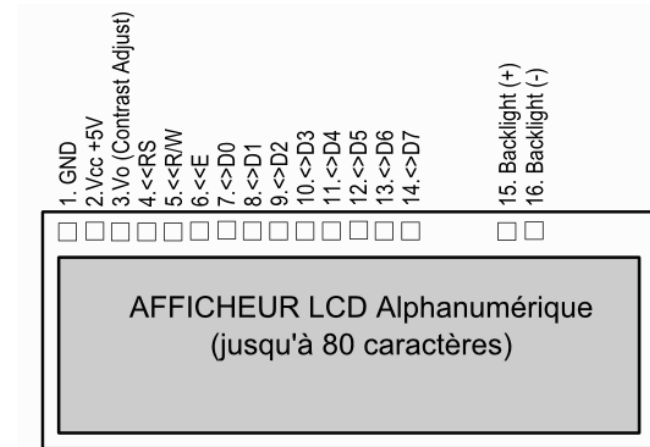
### Brochage

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 laissez pas impressionner, c'est simple !

### Mode de fonctionnement

Un afficheur LCD alpha-numérique standard peut fonctionner :

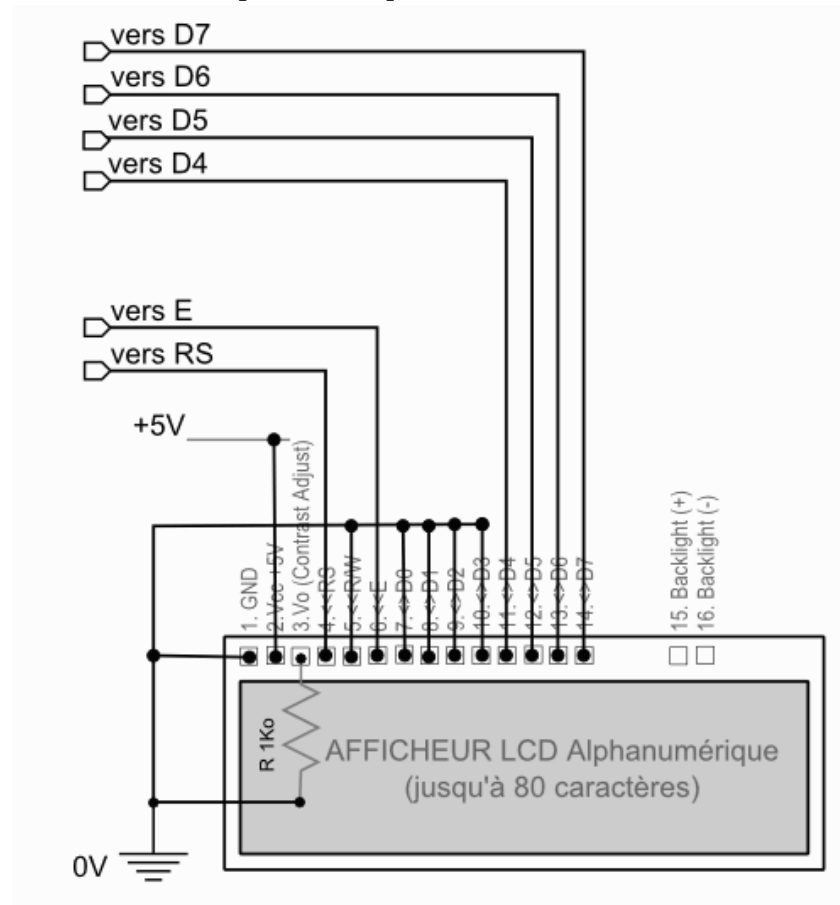
- soit en mode dits « 8 bits » et dans ce cas nécessite 8 broches de données + 3 broches de commandes soit 11 broches !
- soit en mode dits « 4bits » et dans ce cas nécessite 4 broches de données + 2 broches de commandes soit seulement **6 broches**. **C'est ce mode de fonctionnement que nous allons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

## 5. Rappel : 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 :

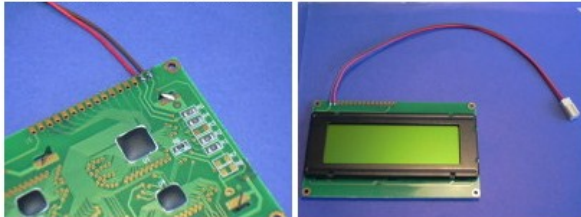




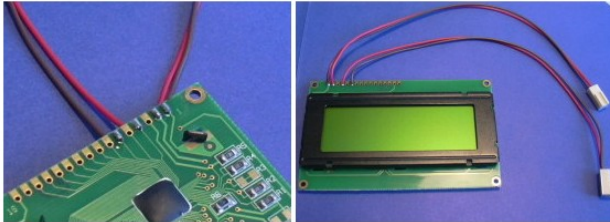
## 6. **Rappel : Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : description concrète**

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.

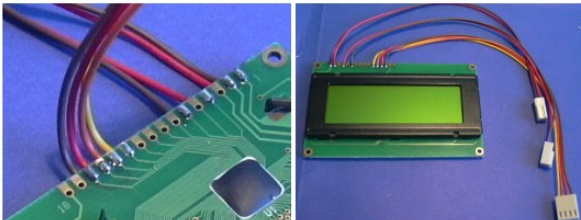
On commence par souder le connecteur 2 broches sur fils sur le + et - de l'afficheur :



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



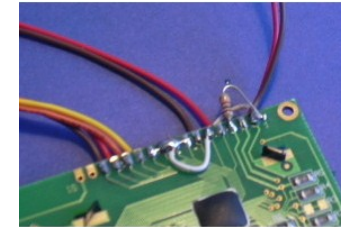
On soude ensuite le connecteur 4 broches sur fils sur les broches D4 à D7 :



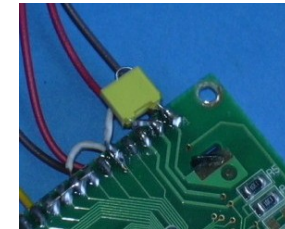
On soude ensuite entre-elles les broches D0 à D3, la broche RW et la broche 0V



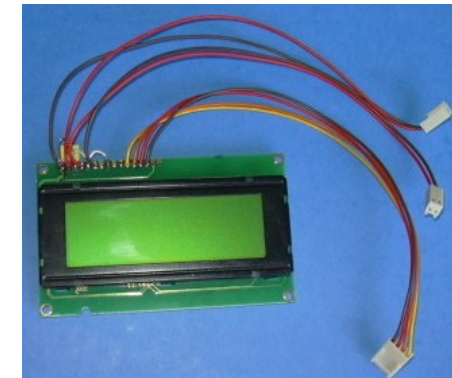
On soude la résistance de 1Ko entre le 0V et la broche Vo



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



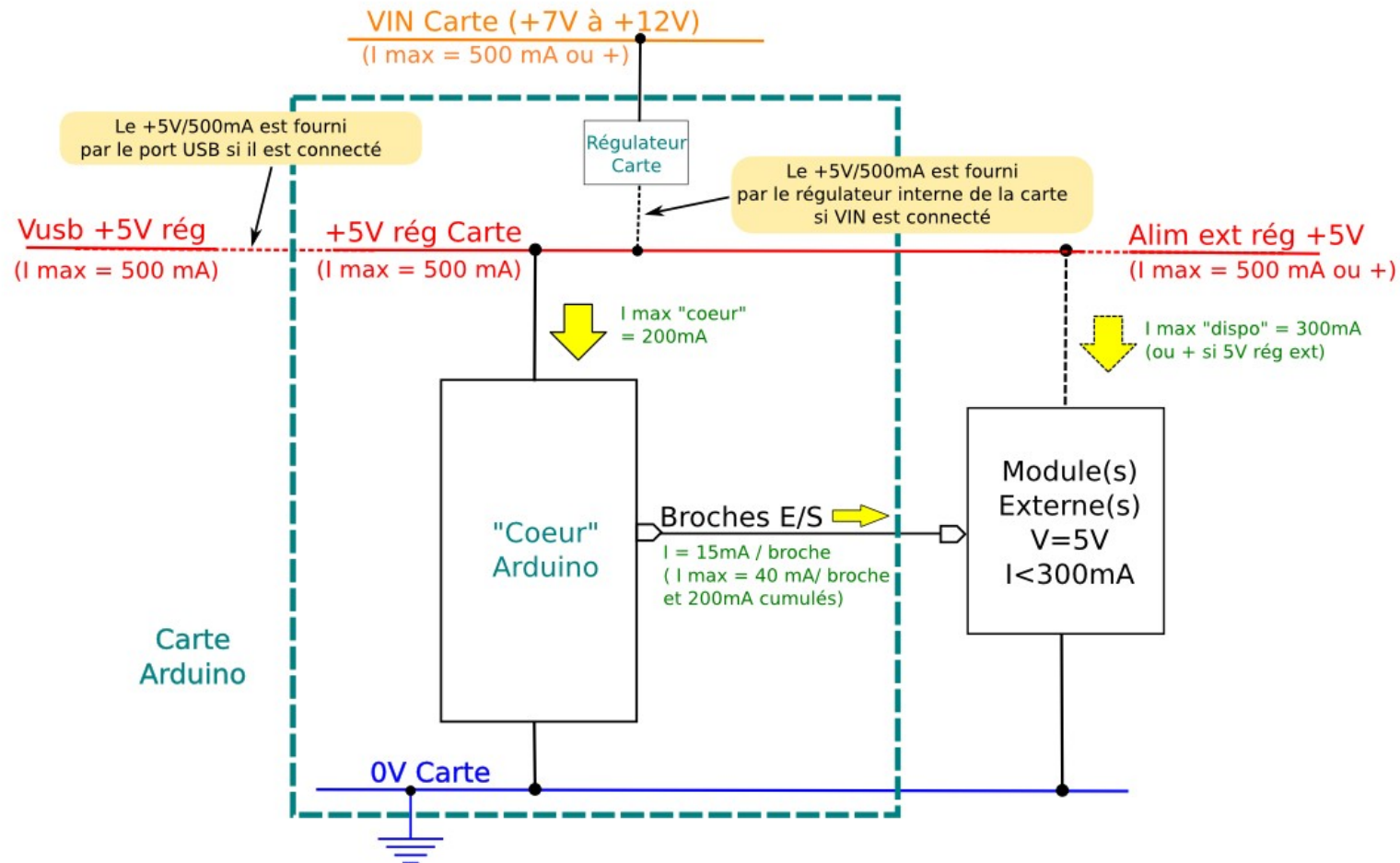
Voici à quoi ressemble le LCD préparé fini et prêt à l'emploi avec votre carte Arduino :



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

## 7. Rappel : 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).

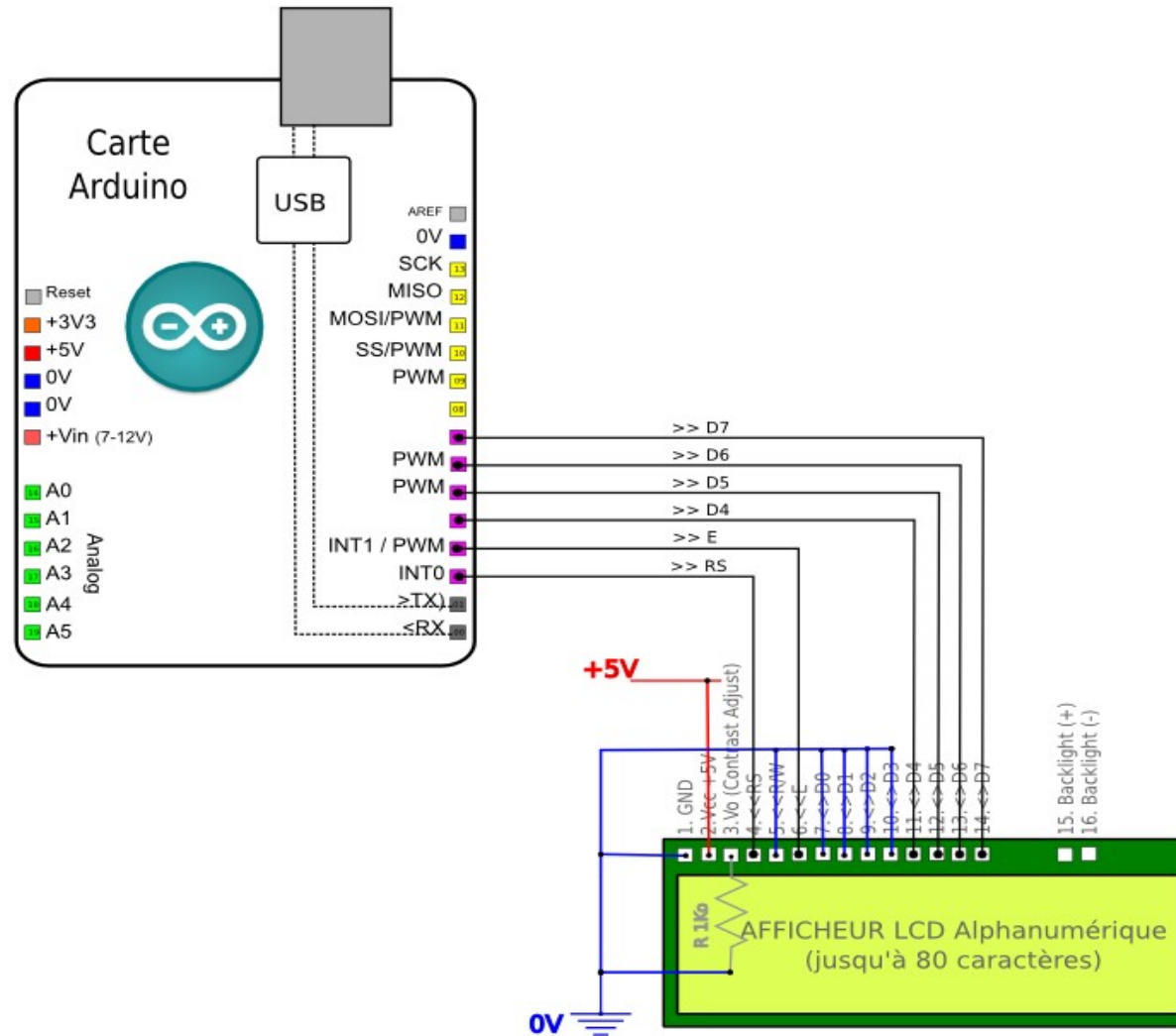




## 8. Utilisation d'un afficheur LCD « préparé » avec une carte Arduino : le montage

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

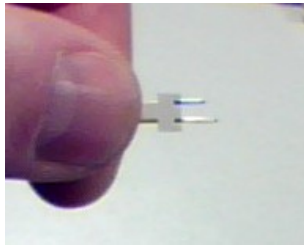


## 9. Utilisation d'un afficheur LCD préparé avec une carte Arduino : en images

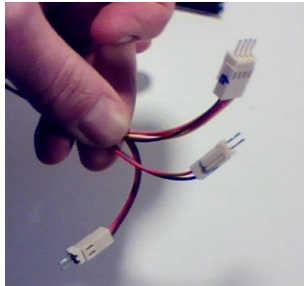
Commencer par prendre les connecteurs droits pour CI :



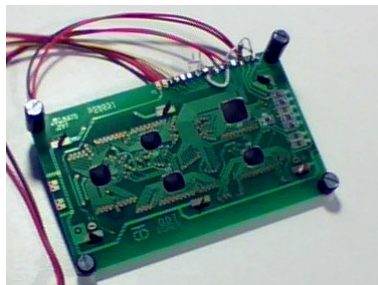
et à l'aide d'une pince, ressortir les broches droites de manière à ce qu'elles soient plus longues en partie extérieure :



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

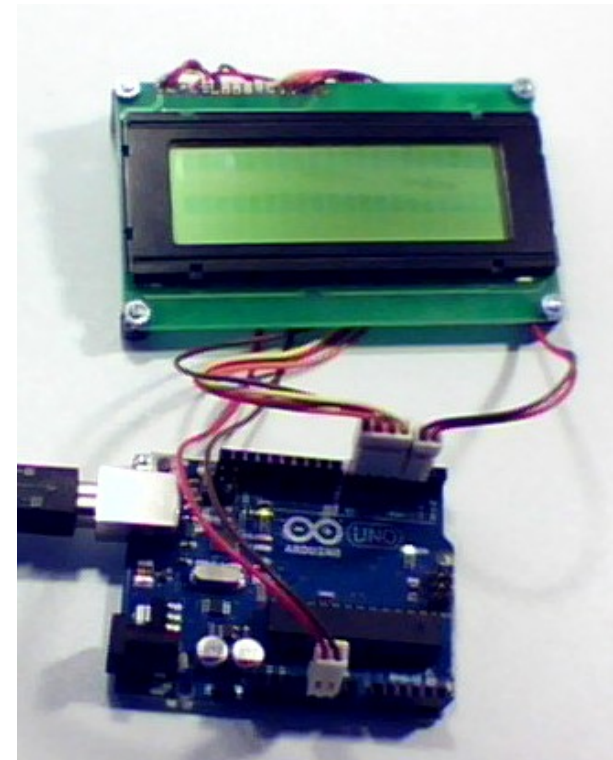


Dernière petite chose : on pourra mettre l'afficheur « sur pieds » à l'aide de 2 entretoises 5mm à l'avant et 2 entretoises 15mm à l'arrière de façon à ce qu'il soit légèrement incliné vers l'avant une fois posé sur ses pieds.



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 : 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](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...

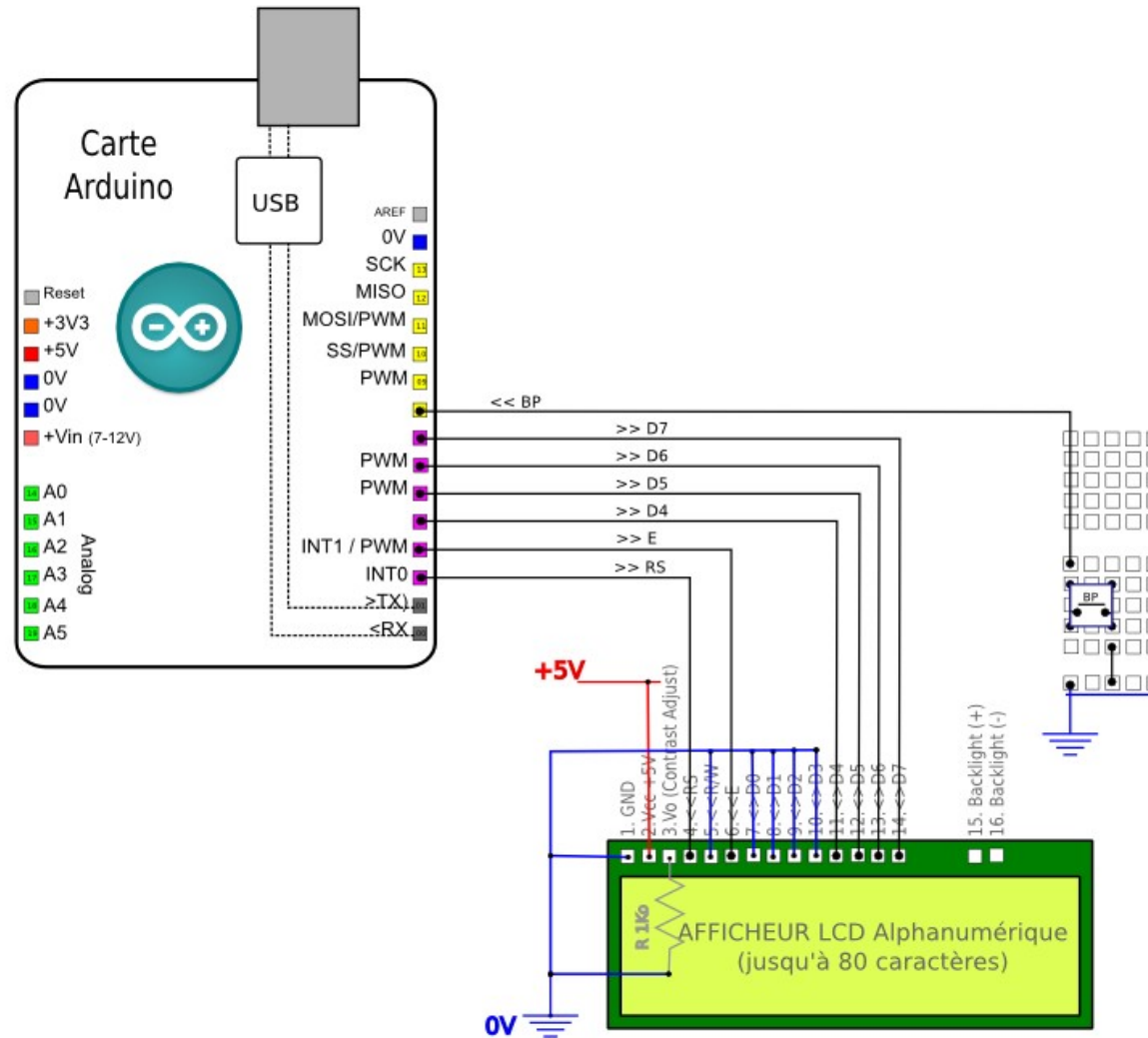


Cette librairie est présentée en détail dans l'atelier précédent dédié aux afficheurs LCD.

## 11. Afficheur LCD standard et un bouton poussoir : le montage

On connecte :

- les broches RS, E et D4-D7 de l'afficheur LCD « préparé » connectées sur 6 broches de la carte Arduino utilisées en sorties numériques,
- 1 bouton poussoir sur 1 broche de la carte Arduino utilisée en entrées numérique avec « rappel au plus » interne activé (si vous ne savez pas ce qu'est le « rappel au plus », voir l'atelier « bouton poussoir »).



## 12. Afficheur LCD et un bouton-poussoir : la « pompe à essence »

Ce que nous allons faire ici :



- Pour pousser un petit peu plus loin dans l'utilisation de l'affichage des calculs sur afficheur LCD, je vous propose ici de simuler la « pompe à essence ».
- Le principe, que vous connaissez bien je pense, est le suivant :
  - le prix du litre est fixé,
  - les litres défilent (ici par appui sur un bouton poussoir) et le total correspondant à la multiplication du prix par le nombre de litre s'affiche en direct.
- Vous ne pourrez pas ouvrir une station service pour autant avec ce code, mais cela vous montre le principe d'affichage de calcul en direct sur afficheur LCD. Y'a plus qu'à... !

### Entête déclarative

#### Déclarations pour l'afficheur LCD

- 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.
- On déclare 2 variables pour aider la gestion du positionnement sur l'afficheur.

#### Déclaration pour les boutons poussoirs

- On déclare également la broche utilisée avec le bouton poussoir.
- On déclare une constante APPUI définissant le niveau actif ddu bouton poussoir : ici **LOW**.

#### Déclaration des variables pour le calcul du prix de l'essence

- On déclare logiquement 3 variables de type **float** correspondant au prix du litre, au nombre de litres et au prix à payer.

```
// Simulation d'affichage de "pompe à essence" sur afficheur LCD
// Le défilement des litres se fait par appui sur un BP

//--- 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

//--- constante de broches
const int BP=8; // broche BP

const int APPUI=LOW; // BP actif sur niveau BAS

// --- Déclaration des variables globales ---

int ligne=1, colonne=1; // variables de positionnement sur l'afficheur

float prixf=1.43; // variable à virgule pour le prix du litre
float totalf=0.0; // variable à virgule pour le prix à payer = prix du
litre x nombre de litres
float nombreLitresf=0.0; // variable à virgule pour le nombre de litre
```



## Fonction **setup()**

### Initialisation du bouton poussoir

- On configure en entrée les 3 boutons poussoirs
- On active le rappel au plus interne.

### Initialisation de l'afficheur LCD

- 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()** suivie d'une rapide pause.
- Puis on affiche un message de test pendant 1 seconde avec la fonction **print**(« Texte») suivi de la fonction **delay()**.
- Le mode **cursor()** est activé au démarrage (mode = o)

### Affichage initial

- Afin de faciliter l'affichage, on réalise un affichage initial à l'aide d'un jeu de combinaison des fonctions d'affichage de façon à afficher les 3 variables et leur titre sur 3 lignes.

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

    //----- initialisation des boutons poussoirs
    pinMode(BP,INPUT); // met la broche en entrée

    digitalWrite(BP,HIGH); // active le rappel au plus sur la broche

    //----- initialisation afficheur LCD ---
    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()

    //----- affichage initial
    ligne=1, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print("Prix  =");

    ligne=2, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print("Litres =");

    ligne=3, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print("Total  =");

    //---- affiche le prix ----
    ligne=1, colonne=10;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(prixf,2); // affiche float avec 2 décimales

    //---- affiche le nombre de litres ----
    ligne=2, colonne=10;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(nombreLitresf,2); // affiche float avec 2 décimales

    totalf=nombreLitresf * prixf; // calcul du total = prix du litre x nombre de
    litres

    //---- affiche le nombre de total ----
    ligne=3, colonne=10;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(totalf,2); // affiche float avec 2 décimales

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

## Fonction **loop()**

- Au sein d'une première condition on teste le bouton BP : si le bouton est appuyé :
  - on incrémente la valeur du nombre de litres,
  - le nouveau total est calculé
- Ensuite, on réalise l'affichage de la nouvelle valeur du nombre de litre et du total à payer.

Remarquer qu'ici, on se contente uniquement de mettre à jour les valeurs des variables qui sont modifiées : les messages fixes (titres) ont été affichés une fois pour toute au niveau de la fonction **setup()**

Retenez cette façon de faire pour obtenir des affichages plus propres (pas de « clignotement ») et une exécution du programme plus rapide (moins d'instructions à exécuter)

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

    if (digitalRead(BP)==APPUI) { // si appui sur BP OK = changement de
mode

        nombreLitresf=nombreLitresf+0.25; // incrémente le nombre de litre par
250 ml
        if (nombreLitresf>100) nombreLitresf=0; // Remise à zéro du nombre de
litres à 100

        totalf=nombreLitresf * prixf; // calcul du total = prix du litre x
nombre de litres

        //---- affiche le nombre de litres ----
        ligne=2, colonne=10;
        lcd.setCursor (colonne-1, ligne-1);
        lcd.print(nombreLitresf,2); // affiche float avec 2 décimales

        //---- affiche le nombre de total ----
        ligne=3, colonne=10;
        lcd.setCursor (colonne-1, ligne-1);
        lcd.print(totalf,2); // affiche float avec 2 décimales

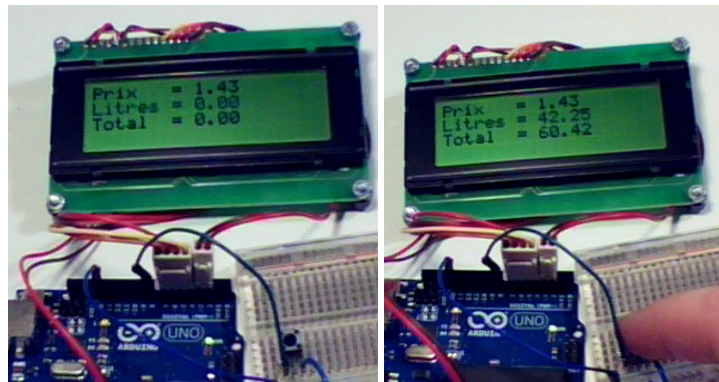
        delay (100); // pause entre deux prises en compte de l'appui

    } // fin if BP appuyé

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

## Fonctionnement du programme

- Une fois la carte Arduino programmée l'appui sur le bouton poussoir fait défiler les litres et modifie l'affichage du total à payer :



### 13. Afficheur LCD et un bouton-poussoir : un chronomètre précis au millième de seconde : le programme.

Ce que nous allons faire ici :



- Avec les Jeux Olympiques, j'ai « l'humeur Olympique » ! Tous ces temps qui s'affichent avec une précision du centième ou du millième de seconde, ça m'a donné quelques idées...
- Ici, je vous propose la réalisation toute simple d'un chronomètre de précision au millième de seconde près. Le principe est simple :
  - un premier appui sur le BP lance le chrono et le temps en millisecondes s'affiche sur le LCD
  - un second appui sur le BP stoppe le chrono
  - un troisième appui le remet à zéro
- A vos marques ! Prêt.. partez ! (je sais, c'est facile... mais je ne pouvais pas la rater celle-là...)



## Entête déclarative

### Déclarations pour l'afficheur LCD

- 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.
- On déclare 2 variables pour aider la gestion du positionnement sur l'afficheur.

### Déclaration pour les boutons poussoirs

- On déclare également la broche utilisée avec le bouton poussoir.
- On déclare une constante APPUI définissant le niveau actif ddu bouton poussoir : ici **LOW**.

### Déclaration des variables pour le comptage du temps

- On déclare plusieurs variables utiles pour la gestion de l'écoulement du temps, de type **long**,
- ainsi que plusieurs variables pour le calcul des minutes, secondes, millisecondes.

```
// Chronomètre au millième de seconde sur afficheur LCD
// Déclenchement, stop et RAZ par appui sur un BP

//--- 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

//--- constante de broches
const int BP=8; // broche BP

const int APPUI=LOW; // BP actif sur niveau BAS

// --- Déclaration des variables globales ---

int ligne=1, colonne=1; // variables de positionnement sur l'afficheur
int comptAppui=0; // variable de comptage du nombre d'appui

long millisTop=0; // variable de mémorisation millis() top chrono
long millis0=0; // variable de mémorisation millis()
long deltaMillis=0; // variable de mémorisation delta ms

int millisecondes=0; // variable de comptage des millisecondes
int secondes=0; // variable de comptage des secondes
int minutes=0; // variable de comptage des minutes
```

## Fonction **setup()**

### **Initialisation du bouton poussoir**

- On configure en entrée les 3 boutons poussoirs
- On active le rappel au plus interne.

### **Initialisation de l'afficheur LCD**

- 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()** suivie d'une rapide pause.
- Puis on affiche un message de test pendant 1 seconde avec la fonction **print**(« Texte») suivi de la fonction **delay()**.
- Le mode **cursor()** est activé au démarrage (mode = o)

### **Affichage initial**

- Afin de faciliter l'affichage, on réalise un affichage initial à l'aide d'un jeu de combinaison des fonctions d'affichage de façon à afficher les titres sur 2 lignes.
- Pour l'affichage des chiffres, on appelle une fonction dédiée pour éviter les répétitions dans le code.

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

    //----- initialisation des boutons poussoirs
    pinMode(BP,INPUT); // met la broche en entrée

    digitalWrite(BP,HIGH); // active le rappel au plus sur la broche

    //----- initialisation afficheur LCD ---
    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()

    //----- affichage initial

    //-- titre
    ligne=1, colonne=1; // positionnement 1er caractère 1ère colonne
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(" *** Chrono *** ");

    //-- titre time
    ligne=2, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print("Time = ");

    //--- affiche chiffres
    afficheTime();

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



## Fonction `loop()` (1)

- La première partie de la fonction `loop()` consiste à gérer l'effet de l'appui sur le BP :
  - au premier appui, déclenche le comptage
  - au 2ème appui, stoppe le comptage
  - au 3ème appui : ré-initialise le comptage

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

    if (digitalRead(BP)==APPUI) { // si appui sur BP

        if (comptAppui==0) { // 1er appui

            millisTop=millis(); // mémorise millis
            millis0=millis(); // mémorise millis
            comptAppui=1;

            delay (250); // pause entre deux prises en compte de l'appui

        } // fin if

        else if (comptAppui==1) { // 2ème appui

            millisTop=0; // RAZ millisTop
            comptAppui=2;

            delay (250); // pause entre deux prises en compte de l'appui

        } // fin else if

        else if (comptAppui==2) { // 3ème appui

            millisecondes=0, secondes=0, minutes=0;

            millisTop=0; // RAZ millisTop
            comptAppui=0; // RAZ comptAppui

            afficheTime();

            delay (250); // pause entre deux prises en compte de l'appui

        } // fin else if

    } // fin if BP appuyé

}
```

### Fonction **loop()** (suite)

- La seconde partie de la fonction **loop()** gère les variables d'écoulement du temps par un système de conditions imbriquées, somme toute assez logique.
- L'affichage des valeurs est réalisé en appelant la fonction dédiée pour l'affichage des chiffres.

```
// --- à chaque passage loop ---  
deltaMillis=millis()-millis0;  
  
if ((deltaMillis>1) && (millisTop!=0) ) { // si plus de 1  
milliseconde écoulée  
  
    millis0=millis(); // mémorise millis()  
  
    millisecondes=millisecondes+deltaMillis;  
  
    //--- gestion des valeurs --  
    if (millisecondes>999) {  
        secondes=secondes+1;  
        millisecondes=0;  
    }  
  
    if (secondes>59) {  
        minutes=minutes+1;  
        secondes=0;  
    }  
  
    if (minutes>59) {  
        minutes=0;  
    }  
  
    //--- affiche chiffres  
    afficheTime();  
  
} // fin passage toutes les millisecondes  
  
} // fin de la fonction loop()
```

## Fonction afficheTime()

- Cette fonction contient le code utile pour afficher les valeurs au format **00'00"000ms**
- Bien noter au passage comment le type des variables numériques est modifié (ou « casté ») en **String** pour que ces variables soient bien considérées comme une chaîne de caractère.

```
//----- fonction d'affichage du temps au format 00'00"000ms
void afficheTime() {

    ligne=2, colonne=8;
    lcd.setCursor (colonne-1, ligne-1);

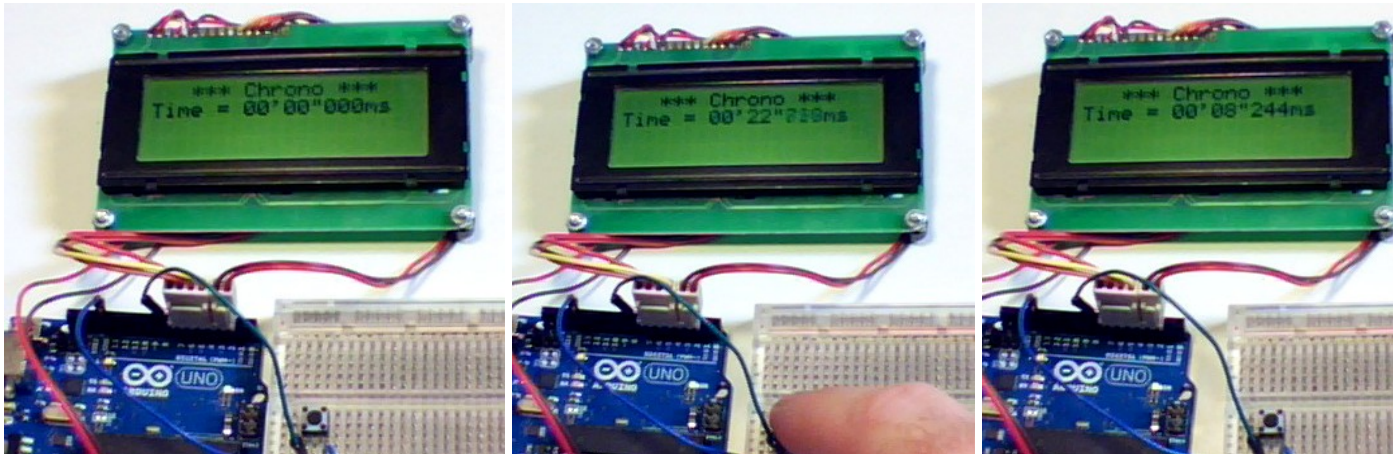
    if (minutes<10) lcd.print("0"+(String)minutes+""); else
    lcd.print((String)minutes+""); // minutes
    if (secondes<10) lcd.print("0"+(String)secondes+"\"); else
    lcd.print((String)secondes+"\"); // secondes

    if ( (millisecondes<1000) && (millisecondes>=100) )
    lcd.print((String)millisecondes+"ms"); // millisecondes
    else if ((millisecondes<100) && (millisecondes>=10)) lcd.print("0"+
    (String)millisecondes+"ms");
    else lcd.print("00"+(String)millisecondes+"ms");

} // fin afficheTime
```

## Fonctionnement du programme

- Une fois la carte Arduino programmée :
  - l'appui sur le bouton poussoir lance le comptage
  - un nouvel appui stoppe le comptage
  - un autre appui réinitialise l'affichage et ainsi de suite.



**Bravo : vous avez réalisé votre premier chronomètre « fait maison » !**

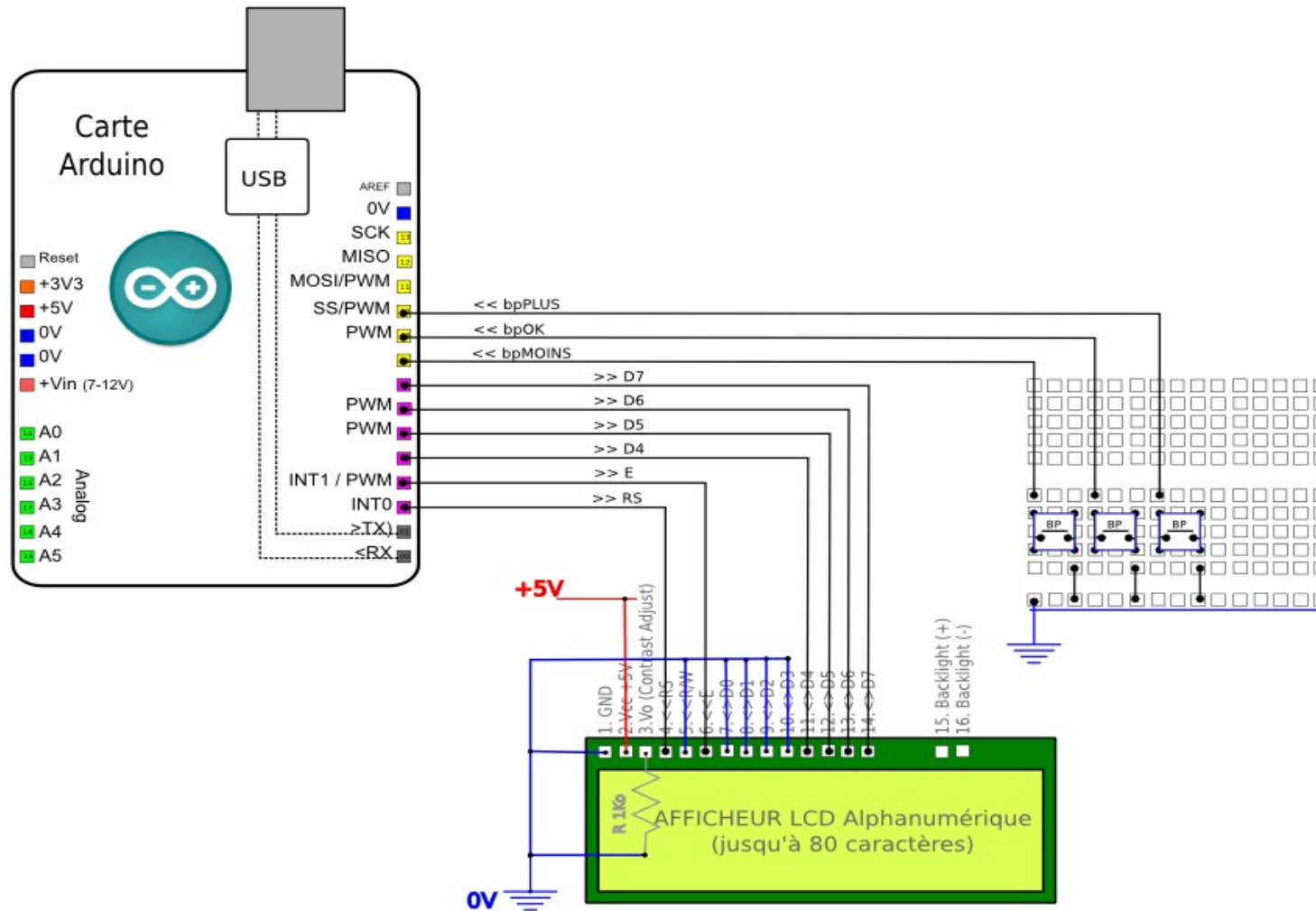
Les « puristes » me diront que la précision laisse potentiellement à désirer et ils auront raison... Il sera cependant possible d'être très précis (à la microseconde près!) à l'aide des interruptions externes, comme nous le verrons dans un autre atelier. On pourra de cette façon chronométrer très précisément la chute d'un objet par exemple... et même (re-)calculer la constante de la gravité !

Voir ici, par exemple, cette page sur mon site : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.SCIENCEGravitometre](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.SCIENCEGravitometre)

## 14. Afficheur LCD standard et trois boutons poussoirs : le montage

On connecte :

- les broches RS, E et D4-D7 de l'afficheur LCD « préparé » connectées sur 6 broches de la carte Arduino utilisées en sorties numériques,
- 3 boutons poussoirs sur 3 broches de la carte Arduino utilisées en entrées numériques avec « rappel au plus » interne activé (si vous ne savez pas ce qu'est le « rappel au plus », voir l'atelier « bouton poussoir »).



## 15. Afficheur LCD : Afficher la détection des appuis sur l'afficheur LCD : le programme

### Ce que nous allons faire ici :

Ce programme ne fait rien de bien sorcier, mais il permet de poser les bases d'un programme utilisant afficheur LCD et boutons poussoirs. Ici, seul l'appui sur chaque bouton poussoir est affiché sous forme d'un message. A partir de cette base, vous serez en mesure d'écrire et d'élaborer vos propres programmes utilisant boutons poussoirs et LCD standard alpha-numérique. A ce stade, vous devriez être capable d'écrire ce programme tout seul. Allez, on se lance....

### Entête déclarative

#### Déclarations pour l'afficheur LCD

- 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.

#### Déclaration pour les boutons poussoirs

- On déclare également les 3 broches utilisées avec les 3 boutons poussoirs.
- On déclare une constante APPUI définissant le niveau actif des boutons poussoirs : ici **LOW**.

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

// Tester l'appui de 3 boutons poussoirs avec 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

//--- constante de broches des BP
const int bpMOINS=8; // broche BP MOINS
const int bpOK=9; // broche BP OK
const int bpPLUS=10; // broche BP +

const int APPUI=LOW; // BP actif sur niveau BAS
```



## Fonction **setup()**

### **Initialisation des boutons poussoirs**

- On configure en entrée les 3 boutons poussoirs
- On active le rappel au plus interne.

### **Initialisation de l'afficheur LCD**

- 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()** suivie d'une rapide pause.
- Puis on affiche un message de test pendant 1 seconde avec la fonction **print**(« Texte») suivi de la fonction **delay()**.
- Le mode **cursor()** est activé au démarrage (mode = 0)

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

    //----- initialisation des boutons poussoirs
    pinMode(bpPLUS,INPUT); // met la broche en sortie
    pinMode(bpOK,INPUT); // met la broche en sortie
    pinMode(bpMOINS,INPUT); // met la broche en sortie

    digitalWrite(bpPLUS,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpOK,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpMOINS,HIGH); // active le rappel au plus sur la broche

    //----- initialisation afficheur LCD ---
    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.blink(); // curseur clignotant

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

## Fonction `loop()`

- Au sein d'une première condition on teste le bouton OK : si le bouton est appuyé, on affiche un rapide message en 0,0 avant d'effacer l'écran.
- Ensuite, on teste l'appui sur le bouton PLUS et on affiche également un message si il est appuyé.
- Ensuite, on teste l'appui sur le bouton MOINS, et de même, on affiche un message si il est appuyé.
- A chaque appui détecté, la pause utilisée pour l'affichage aura également le rôle de pause « anti-rebond ».

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

    if (digitalRead(bpOK)==APPUI) { // si appui sur BP OK = changement de
mode

        lcd.print("APPUI BP OK");
        delay(2000); // pause affichage + anti-rebond
        lcd.clear();
        delay(10);

    } // fin if bpOK

    if (digitalRead(bpPLUS)==APPUI) { // si appui bpPLUS

        lcd.print("APPUI BP PLUS");
        delay(2000); // pause affichage + anti-rebond
        lcd.clear();
        delay(10);

    } // fin si bpPLUS

    if (digitalRead(bpMOINS)==APPUI) { // si appui bpMOINS

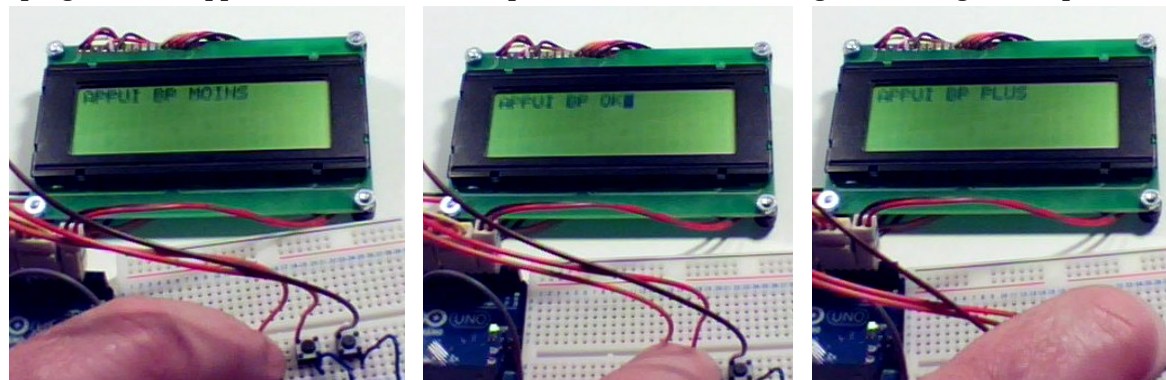
        lcd.print("APPUI BP MOINS");
        delay(2000); // pause affichage + anti-rebond
        lcd.clear();
        delay(10);

    } // fin si bpMOINS

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

## Fonctionnement du programme

- Une fois la carte Arduino programmée l'appui sur l'un des boutons poussoir entraîne l'affichage du message correspondant.



Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Apprendre à utiliser des boutons poussoirs avec un afficheur LCD alpha-numérique et Arduino.

## 16. Afficheur LCD : Déplacer et modifier le curseur à l'aide de boutons poussoirs : le programme

### Rappels

- Je rappelle ici que le curseur est invisible par défaut mais il peut être visible sous la forme d'un \_ ou d'un rectangle noir clignotant. Quatre fonctions 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.
- De la même façon nous avons vu la fonction `setCursor(colonne,ligne)` qui 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 »)`
- Ici nous allons utiliser 3 boutons poussoirs : 2 pour déplacer le curseur et un pour modifier l'état du curseur.

### Entête déclarative

#### Déclarations pour l'afficheur LCD

- 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.
- On déclare 2 constantes pour le nombre de lignes et de colonnes : Valeurs à adapter à votre situation.
- On déclare également 2 variables pour le positionnement du curseur sur l'afficheur. Origine ici en 1,1 (plus spontané que en 0,0).

#### Déclaration pour les boutons poussoirs

- On déclare également les 3 broches utilisées avec les 3 boutons poussoirs.
- On déclare une constante APPUI définissant le niveau actif des boutons poussoirs : ici `LOW`.

#### Variables globales

- On déclare une variable utilisée pour fixer le mode de fonctionnement du programme :
  - mode = 0 : déplacement du curseur Droite/Gauche
  - mode=1 : déplacement du curseur Haut/Bas

```
// modifier positionnement et l'aspect du curseur d'un afficheur LCD
// à l'aide de 3 boutons poussoirs

//--- 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

const int nombreColonnes=20; // nombre de colonnes de l'afficheurs
const int nombreLignes=4; // nombre de lignes de l'afficheur

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

int ligne=1; // variable de ligne
int colonne=1; // variable de colonne

//--- constante de broches des BP
const int bpMOINS=8; // broche BP MOINS
const int bpOK=9; // broche BP OK
const int bpPLUS=10; // broche BP +
const int APPUI=LOW; // BP actif sur niveau BAS

//--- variables globales utiles
int mode=0; // variable pour changement de mode de fonctionnement
// mode = 0 : droit/gauche + cursor mode=1 : haut/bas + blink
```

## Fonction **setup()**

### **Initialisation des boutons poussoirs**

- On configure en entrée les 3 boutons poussoirs
- On active le rappel au plus interne.

### **Initialisation de l'afficheur LCD**

- 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()** suivie d'une rapide pause.
- Puis on affiche un message de test pendant 1 seconde avec la fonction **print(« Texte»)** suivi de la fonction **delay()**.
- Le mode **cursor()** est activé au démarrage (mode = o)

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

    //----- initialisation des boutons poussoirs
    pinMode(bpPLUS,INPUT); // met la broche en sortie
    pinMode(bpOK,INPUT); // met la broche en sortie
    pinMode(bpMOINS,INPUT); // met la broche en sortie

    digitalWrite(bpPLUS,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpOK,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpMOINS,HIGH); // active le rappel au plus sur la broche

    //----- initialisation afficheur LCD ---
    lcd.begin(nombreColonnes,nombreLignes); // 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(); // curseur actif par défaut

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

## Fonction **loop()**

- Au sein d'une première condition on teste le bouton OK qui sert à modifier le mode :
  - soit en mode 0 (sens droit/gauche), soit en mode 1 (sens haut/bas),
  - le curseur est modifié également : cursor() si mode=0, blink() si mode=1
- Ensuite, on teste l'appui sur le bouton PLUS :
  - si mode 0 : on incrémente colonne
  - si mode 1 : on incrémente ligne
  - des conditions limite la valeur à la taille du LCD
  - le curseur est positionné en conséquence
- Ensuite, on teste l'appui sur le bouton MOINS:
  - si mode 0 : on décrémente colonne
  - si mode 1 : on décrémente ligne
  - des conditions limitent la valeur à 0
  - le curseur est positionné en conséquence

Une fonction **loop()** un peu étoffée, mais rien de bien sorcier.  
Ce code est un bon exercice d'imbrication de conditions.

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

    if (digitalRead(bpOK)==APPUI) { // si appui sur BP OK = changement de
mode
        if (mode==0) {
            mode=1;
            lcd.noCursor(); // change l'aspect du curseur
            lcd.blink();
        } // fin if
        else {
            mode=0;
            lcd.noBlink(); // change l'aspect du curseur
            lcd.cursor();
        } // fin else

        delay(250); // pause anti-rebond
    } // fin if bpOK

    if (digitalRead(bpPLUS)==APPUI) { // si appui bpPLUS
        if (mode==0) { //si Droite/Gauche
            colonne=colonne+1;
            if (colonne>nombreColonnes) colonne=nombreColonnes; // limite
écran
        } // fin mode = 0

        if (mode==1) { // si Haut/Bas
            ligne=ligne+1;
            if (ligne>nombreLignes) ligne=nombreLignes; // limite écran
        } // fin mode = 1

        lcd.setCursor(colonne-1, ligne-1);
        delay(250); // pause anti-rebond
    } // fin si bpPLUS

    if (digitalRead(bpMOINS)==APPUI) { // si appui bpMOINS
        if (mode==0) { //si Droite/Gauche
            colonne=colonne-1;
            if (colonne<1) colonne=1; // limite écran
        } // fin mode = 0

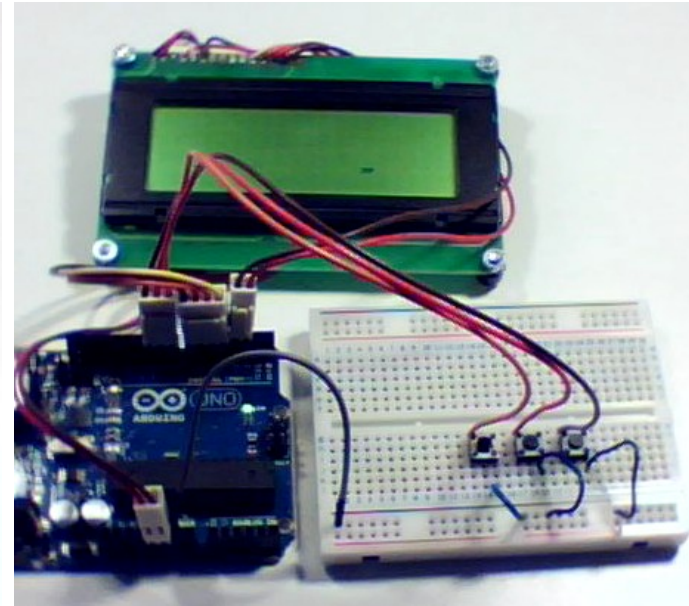
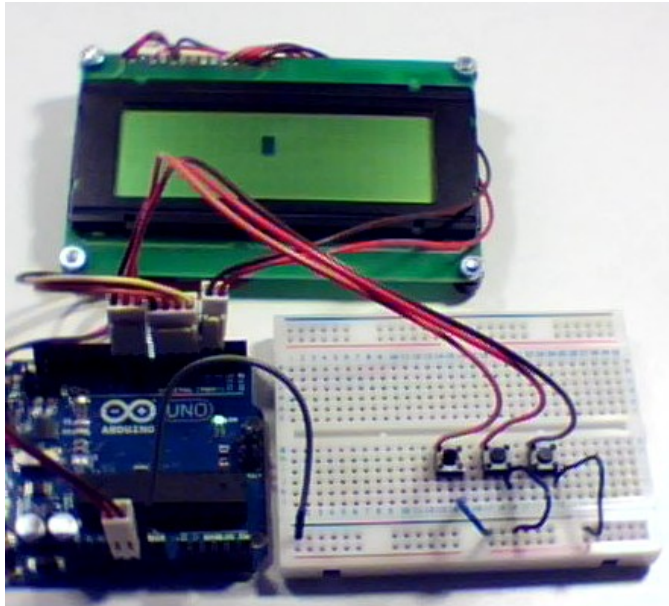
        if (mode==1) { // si Haut/Bas
            ligne=ligne-1;
            if (ligne<1) ligne=1; // limite écran
        } // fin mode = 1

        lcd.setCursor(colonne-1, ligne-1);
        delay(250); // pause anti-rebond
    } // fin si bpMOINS
} // fin de la fonction loop()
```



## Fonctionnement du programme

- Une fois la carte Arduino programmée :
  - l'appui sur le BP OK entraîne la modification du curseur correspondant au changement de mode 0/1
  - l'appui sur le BP PLUS incrémente la ligne ou la colonne selon le mode
  - l'appui sur le BP MOINS décrémente la ligne ou la colonne selon le mode.



## 17. Afficheur LCD : Modifier la valeur de variables à l'aide de boutons poussoirs : le programme

### Ce que nous allons faire ici

- Comme dans le programme précédent, ici nous allons utiliser 3 boutons poussoirs : 2 pour déplacer le curseur et un pour modifier l'état du curseur, à la différence qu'ici lorsque le blink sera activé, les boutons poussoirs auront pour effet de régler la valeur de la variable affichée sur la ligne.
- Ce programme pose les bases pour toutes sortes de montages avec réglage de valeurs de variables par boutons poussoirs. On se lance !

### Entête déclarative

#### Déclarations pour l'afficheur LCD

- 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.
- On déclare 2 constantes pour le nombre de lignes et de colonnes : Valeurs à adapter à votre situation.
- On déclare également 2 variables pour le positionnement du curseur sur l'afficheur. Origine ici en 1,1 (plus spontané que en 0,0).

#### Déclaration pour les boutons poussoirs

- On déclare également les 3 broches utilisées avec les 3 boutons poussoirs.
- On déclare une constante APPUI définissant le niveau actif des boutons poussoirs : ici **LOW**.

#### Variables globales

- On déclare une variable utilisée pour fixer le mode de fonctionnement du programme :
  - mode = 0 : déplacement du curseur réglage de la valeur
  - mode=1 : déplacement du curseur Haut/Bas
- On déclare également 4 variables numériques de type **long** correspondant aux 4 variables à régler. Ces variables sont initialisées à 0.

```
// sélectionner et régler la valeur de variables sur un afficheur LCD
// à l'aide de 3 boutons poussoirs

//--- 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

const int nombreColonnes=20; // nombre de colonnes de l'afficheurs
const int nombreLignes=4; // nombre de lignes de l'afficheur

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

int ligne=1; // variable de ligne
int colonne=1; // variable de colonne

//--- constante de broches des BP
const int bpMOINS=8; // broche BP MOINS
const int bpOK=9; // broche BP OK
const int bpPLUS=10; // broche BP +

const int APPUI=LOW; // BP actif sur niveau BAS

//--- variables globales utiles
int mode=0; // variable pour changement de mode de fonctionnement
// mode = 0 : droit/gauche + cursor mode=1 : haut/bas + blink

long valeur1=0; // variable 1
long valeur2=0; // variable 1
long valeur3=0; // variable 1
long valeur4=0; // variable 1
```

## Fonction **setup()**

### Initialisation des boutons poussoirs

- On configure en entrée les 3 boutons poussoirs
- On active le rappel au plus interne.

### Initialisation de l'afficheur LCD

- 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()** suivie d'une rapide pause.
- Puis on affiche un message de test pendant 1 seconde avec la fonction **print**(« Texte») suivi de la fonction **delay()**.
- Le mode **noCursor()** est activé au démarrage (mode = 0). ce mode correspond ici au mode de sélection de la ligne.

### Affichage initial

- A l'aide d'une combinaison des fonctions **setCursor** et **print**, on réalise l'affichage des titres et des 4 valeurs sur 4 lignes.
- On se positionne sur la 1ère ligne par défaut.

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

    //----- initialisation des boutons poussoirs
    pinMode(bpPLUS,INPUT); // met la broche en sortie
    pinMode(bpOK,INPUT); // met la broche en sortie
    pinMode(bpMOINS,INPUT); // met la broche en sortie

    digitalWrite(bpPLUS,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpOK,HIGH); // active le rappel au plus sur la broche
    digitalWrite(bpMOINS,HIGH); // active le rappel au plus sur la broche

    //----- initialisation afficheur LCD ---
    lcd.begin(nombreColonnes,nombreLignes); // 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.noCursor(); // curseur inactif par défaut

    //--- affichage initial---

    ligne=1, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(" Valeur1 =");

    ligne=2, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(" Valeur2 =");

    ligne=3, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(" Valeur3 =");

    ligne=4, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(" Valeur4 =");

    //----- valeurs initiales
    lcd.setCursor(11, 1-1), lcd.print(valeur1);
    lcd.setCursor(11, 2-1), lcd.print(valeur2);
    lcd.setCursor(11, 3-1), lcd.print(valeur3);
    lcd.setCursor(11, 4-1), lcd.print(valeur4);

    //----- sélecteur initial -----
    ligne=1, colonne=1;
    lcd.setCursor (colonne-1, ligne-1);
    lcd.print(">");

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

## Fonction **loop()** (1)

- Au sein d'une première condition on teste le bouton OK qui sert à modifier le mode :
  - soit en mode 0 (sens droit/gauche), soit en mode 1 (sens haut/bas),
  - le curseur est modifié également : cursor() si mode=0, blink() si mode=1

Une fonction **loop()** un peu étoffée, mais rien de bien sorcier.  
Ce code est un bon exercice d'imbrication de conditions.

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

    if (digitalRead(bpOK)==APPUI) { // si appui sur BP OK = changement de
mode

        if (mode==0) { // passe en mode réglage
            mode=1;
            lcd.noCursor(); // change l'aspect du curseur
            lcd.blink();

            if (ligne==1) lcd.setCursor(11, ligne-1), lcd.print(valeur1);
            if (ligne==2) lcd.setCursor(11, ligne-1), lcd.print(valeur2);
            if (ligne==3) lcd.setCursor(11, ligne-1), lcd.print(valeur3);
            if (ligne==4) lcd.setCursor(11, ligne-1), lcd.print(valeur4);

        } // fin if
    else { // passe en mode sélection de ligne
        mode=0;
        lcd.noBlink(); // change l'aspect du curseur
        lcd.noCursor();

    } // fin else

    delay(250); // pause anti-rebond

} // fin if bpOK
```

## Fonction **loop()** (2)

- Ensuite, on teste l'appui sur le bouton PLUS :
  - si mode 0 : on incrémente ligne
  - si mode 1 : on incrémente la variable de la ligne courante
  - à chaque fois, le curseur est positionné en conséquence

```
if (digitalRead(bpPLUS)==APPUI) { // si appui bpPLUS

  if (mode==0) { //si Haut / bas

    lcd.setCursor(colonne-1, ligne-1);
    lcd.print(" ");

    ligne=ligne+1;
    if (ligne>nombreLignes) ligne=nombreLignes; // limite écran

    lcd.setCursor(colonne-1, ligne-1);
    lcd.print(">");

    delay(250); // pause anti-rebond

  } // fin mode = 0

  if (mode==1) { // si réglage valeur

    if (ligne==1) valeur1=valeur1+1, lcd.setCursor(11, ligne-1),
    lcd.print(valeur1);
    if (ligne==2) valeur2=valeur2+1, lcd.setCursor(11, ligne-1),
    lcd.print(valeur2);
    if (ligne==3) valeur3=valeur3+1, lcd.setCursor(11, ligne-1),
    lcd.print(valeur3);
    if (ligne==4) valeur4=valeur4+1, lcd.setCursor(11, ligne-1),
    lcd.print(valeur4);

    delay(100); // pause anti-rebond

  } // fin mode = 1

} // fin si bpPLUS
```

### Fonction `loop()` (3)

- Ensuite, on teste l'appui sur le bouton MOINS:
  - si mode 0 : on décrémente ligne
  - si mode 1 : on décrémente la variable de la ligne courante
  - à chaque fois le curseur est positionné en conséquence

```
if (digitalRead(bpMOINS)==APPUI) { // si appui bpMOINS

    if (mode==0) { //si Haut/Bas

        lcd.setCursor(colonne-1, ligne-1);
        lcd.print(" ");

        ligne=ligne-1;
        if (ligne<1) ligne=1; // limite écran

        lcd.setCursor(colonne-1, ligne-1);
        lcd.print(">");

        delay(250); // pause anti-rebond

    } // fin mode = 0

    if (mode==1) { // si réglage valeur

        if (ligne==1) valeur1=valeur1-1, lcd.setCursor(11, ligne-1),
        lcd.print(valeur1);
        if (ligne==2) valeur2=valeur2-1, lcd.setCursor(11, ligne-1),
        lcd.print(valeur2);
        if (ligne==3) valeur3=valeur3-1, lcd.setCursor(11, ligne-1),
        lcd.print(valeur3);
        if (ligne==4) valeur4=valeur4-1, lcd.setCursor(11, ligne-1),
        lcd.print(valeur4);

        delay(100); // pause anti-rebond

    } // fin mode = 1

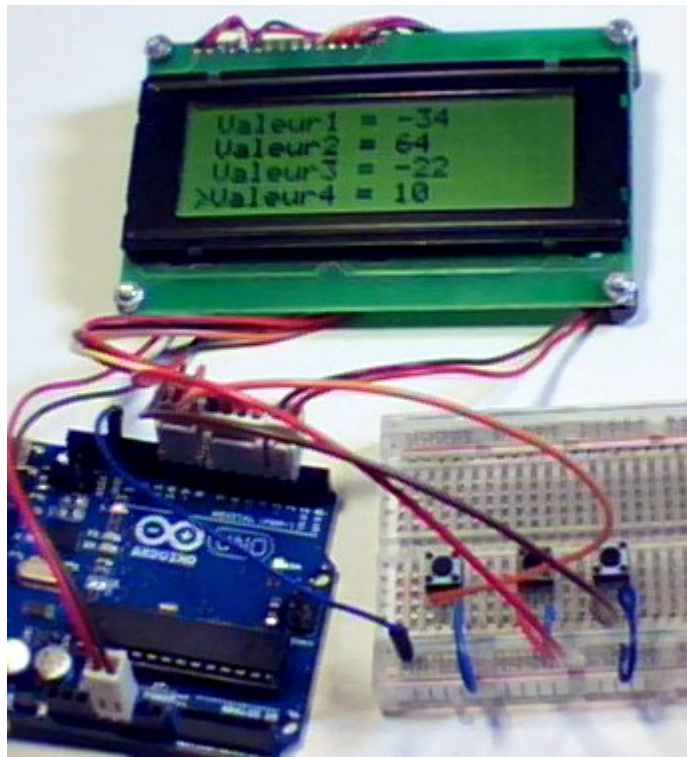
} // fin si bpMOINS

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



## Fonctionnement du programme

- Une fois la carte Arduino programmée :
  - l'appui sur le BP OK entraîne la modification du curseur correspondant au changement de mode 0/1
  - en mode 0 (=sélection de ligne ) :
    - l'appui sur le BP PLUS incrémente la ligne
    - l'appui sur le BP MOINS décrémente la ligne
  - en mode 1 (=réglage de la valeur des variables ) :
    - l'appui sur le BP PLUS incrémente la valeur de la variable de la ligne courante
    - l'appui sur le BP MOINS décrémente la valeur de la variable de la ligne courante



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

Les fonctions usuelles de la librairie **LiquidCrystal**, 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\(\)](#)

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)

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

- d'écrire un programme utilisant un afficheur LCD et des boutons poussoirs
- d'écrire un programme utilisant les boutons poussoirs pour contrôler l'affichage de valeurs numériques

# Table des matières

Intro |  
Matériel nécessaire pour les ateliers Arduino |  
Matériel spécifique nécessaire pour cet atelier |  
Rappel : Fiche Technique : Afficheur LCD alpha-numérique standard |  
Rappel : Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : le schéma théorique |  
Rappel : Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : description concrète |  
Rappel : 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 : Langage Arduino : la librairie LiquidCrystal pour le contrôle des afficheurs LCD standards |  
Afficheur LCD standard et un bouton poussoir : le montage |  
Afficheur LCD et un bouton-poussoir : la « pompe à essence » |  
Afficheur LCD et un bouton-poussoir : un chronomètre précis au millième de seconde : le programme. |  
Afficheur LCD standard et trois boutons poussoirs : le montage |  
Afficheur LCD : Afficher la détection des appuis sur l'afficheur LCD : le programme |  
Afficheur LCD : Déplacer et modifier le curseur à l'aide de boutons poussoirs : le programme |  
Afficheur LCD : Modifier la valeur de variables à l'aide de boutons poussoirs : le programme |  
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](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)