

# Utiliser un afficheur LCD alpha-numérique et un clavier matriciel avec Arduino : les bases et manipulation de valeurs entières.



## 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 !**

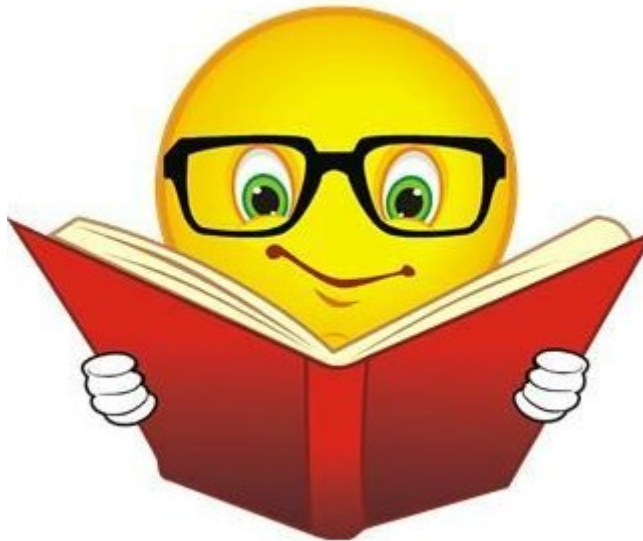
**Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !**

## 1. Intro

L'objectif ici est :

- de rappeler le principe de fonctionnement d'un clavier matriciel et d'un afficheur LCD alphanumérique,
- de rappeler comment utiliser les bibliothèques **LiquidCrystal** et **Keypad** du langage Arduino,
- d'écrire des programmes permettant :
  - de lire les touches appuyées sur un clavier numérique
  - de mémoriser une séquence de touches saisies
  - de saisir une valeur numérique entière
- de coder des applications utilisant un clavier matriciel, notamment :
  - un décodeur de codes secrets
  - un convertisseur décimal / hexadécimal / binaire

... afin d'être en mesure d'utiliser un clavier matriciel et un afficheur LCD alpha-numérique avec Arduino en fonction de ses besoins.

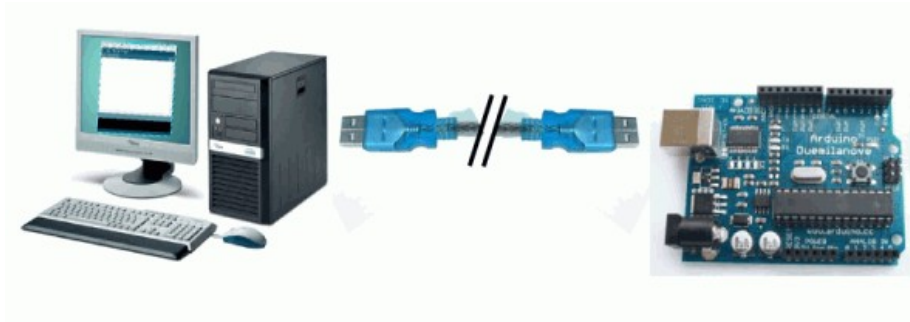


**Prêt ? C'est parti !**

## 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

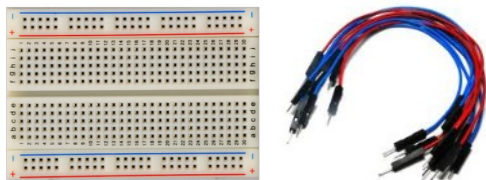


L'espace de développement Arduino associe :

- 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

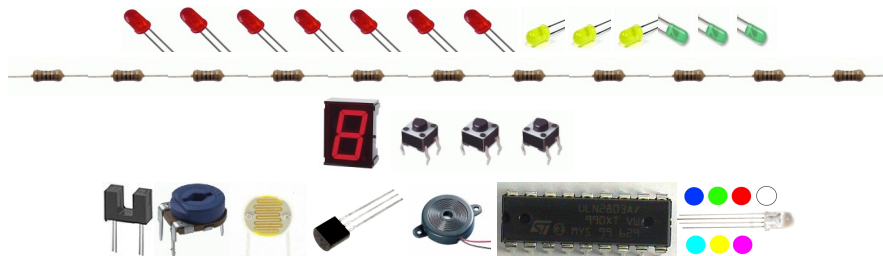


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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 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)

**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)

### 3. Matériel spécifique nécessaire pour utiliser un afficheur LCD dans cet atelier

Pour cet atelier vous aurez besoin également :

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

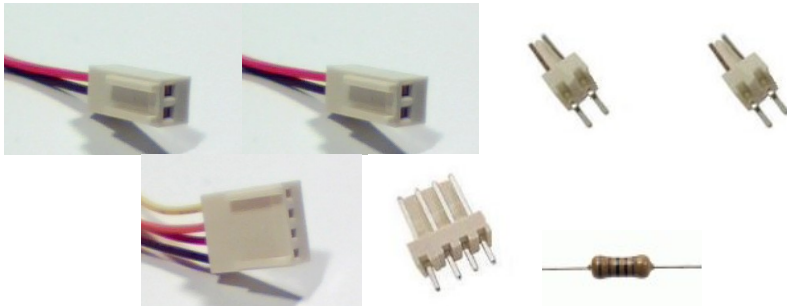


L'afficheur LCD alphanumérique standard 4 lignes x 20 colonnes est facile à utiliser avec la carte Arduino à l'aide de 6 broches numériques. Cet afficheur permet d'afficher des lettres et des chiffres sur 4 lignes et 20 colonnes. Un équipement idéal pour créer des applications autonomes réalisant des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

Pour cet atelier vous aurez besoin également :

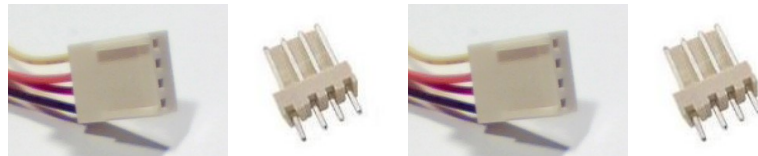
**D'un afficheur clavier matriciel 16 touches (4 lignes x 4 colonnes)**



Clavier 16 touches permettant la saisie de valeurs numériques, facile à utiliser avec une carte Arduino en utilisant des connecteurs 4 broches sur fils.

disponible chez : <http://www.gotronic.fr/> | Code : 07276

#### Du matériel nécessaire pour « préparer » un clavier matriciel pour une utilisation simplifiée avec Arduino



Pour être utilisable facilement avec Arduino, un clavier matriciel 4x4 brut nécessite une petite préparation assez simple à l'aide d'éléments de connectique simples et une résistance :

- 2 connecteur femelle sur fils – 4 contacts (code : 09827 )
- 2 connecteur droit – 4 contacts (code : 09817)

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

## 5. 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.

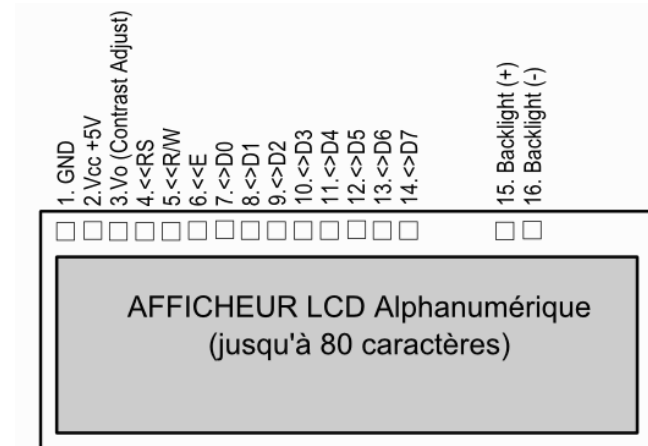
### Brochage

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.**

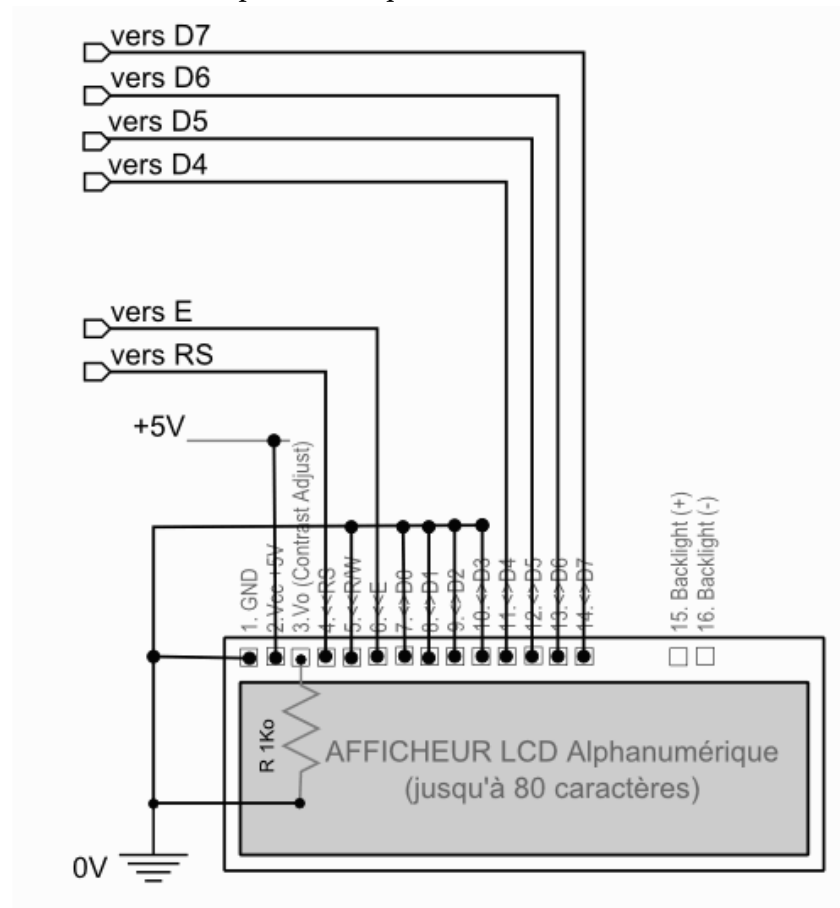


## 6. Rappel : Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : le schéma théorique

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).

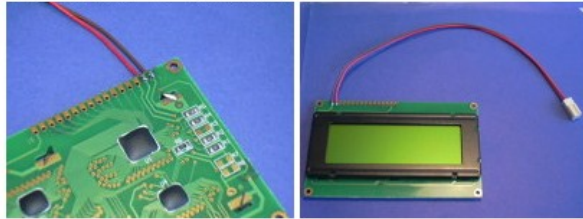
Voici le schéma de cette connexion simplifiée de l'afficheur LCD alpha-numérique :



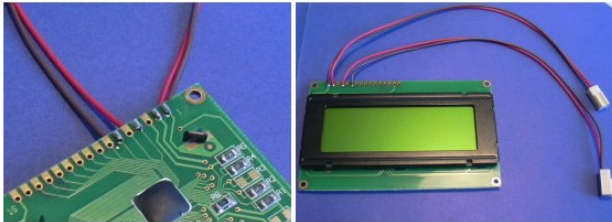
## 7. Préparation d'un afficheur LCD pour une utilisation simplifiée avec Arduino : description concrète

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.

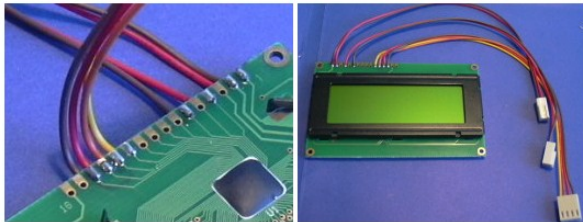
On commence par souder le connecteur 2 broches sur fils sur le + et - de l'afficheur :



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



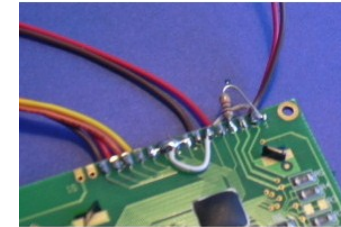
On soude ensuite le connecteur 4 broches sur fils sur les broches D4 à D7 :



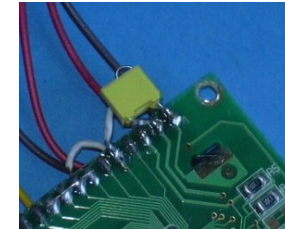
On soude ensuite entre-elles les broches Do à D3, la broche RW et la broche 0V



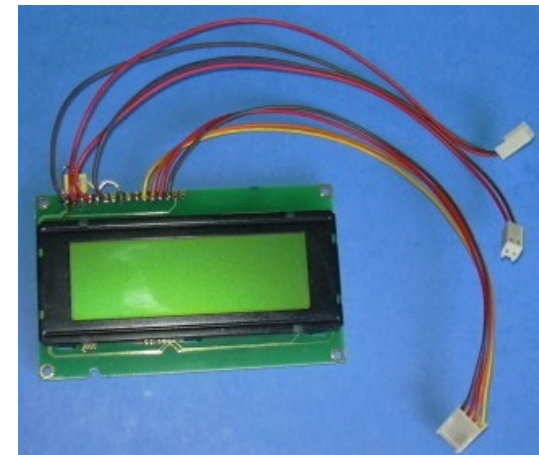
On soude la résistance de 1Ko entre le 0V et la broche Vo



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 !



## 8. Rappel : Fiche technique : clavier matriciel 16 touches (4 lignes x 4 colonnes)

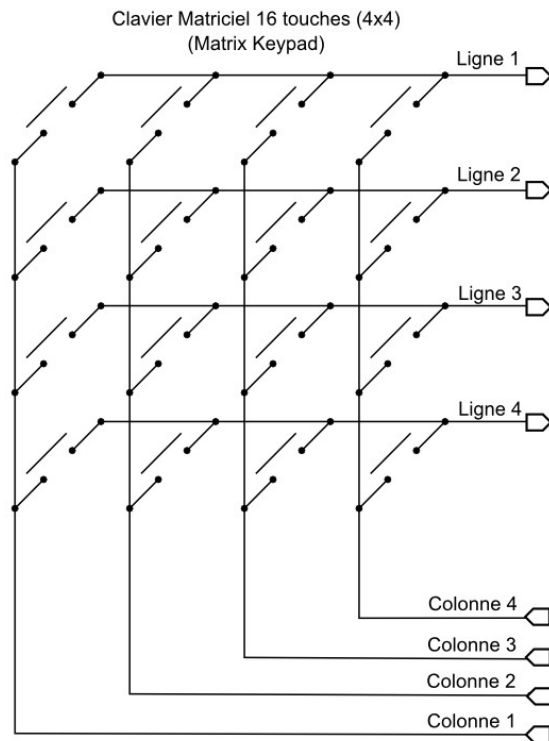
### Description

Un clavier matriciel 16 touches 4 x 4 n'est ni plus ni moins qu'un boîtier plastique dans lequel 16 boutons poussoirs ont été regroupés. Le clavier dispose d'un connecteur à 8 broches.



### Schéma interne

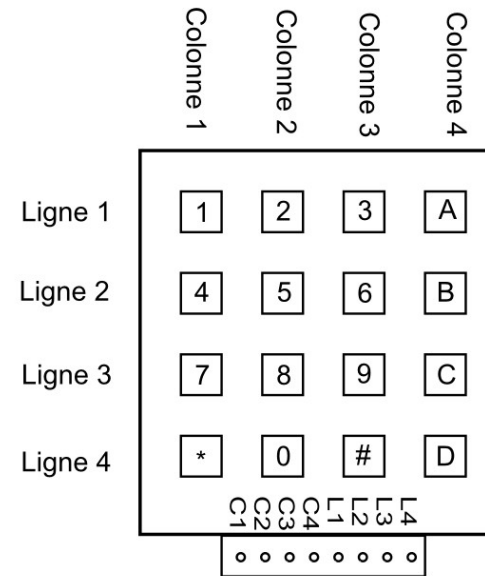
Le principe de la connexion interne consiste à mettre en contact la ligne et la colonne d'un bouton poussoir du clavier lors de l'appui sur ce bouton poussoir. Le schéma interne correspondant est le suivant :



### Brochage

Un afficheur LCD standard dispose typiquement :

- de 4 broches de lignes
- de 4 broches de colonnes



### Caractéristiques électriques

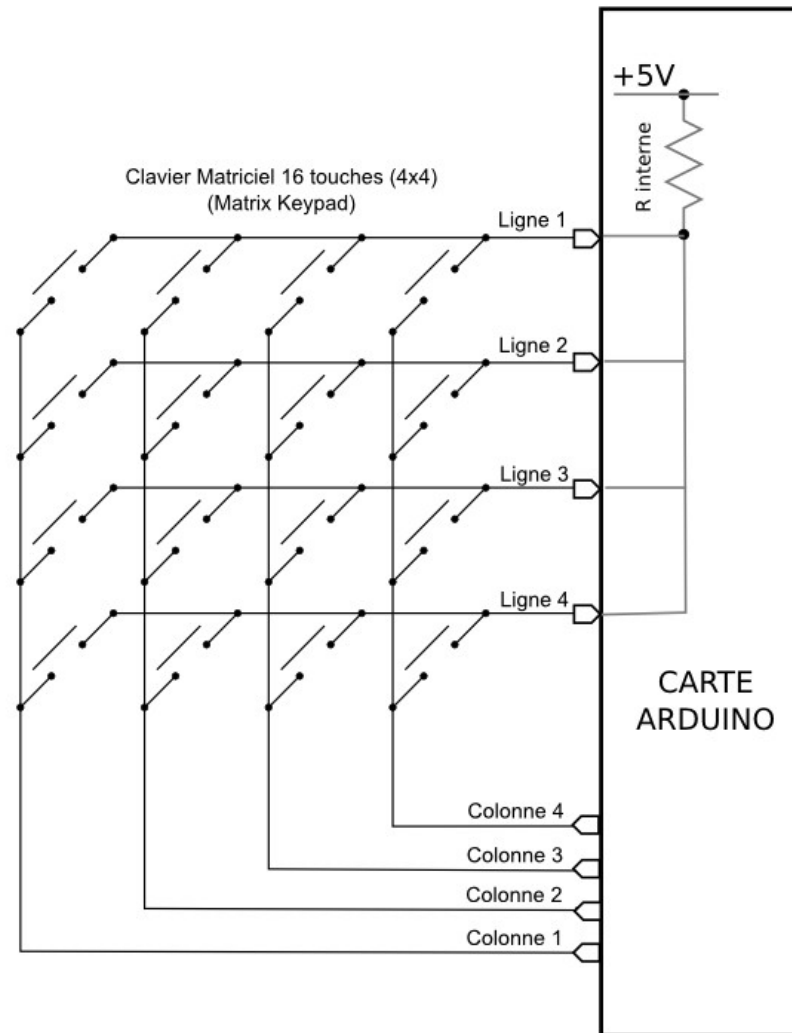
- Un clavier matriciel ne consomme rien : il se contente de mettre en contact la ligne et la colonne.

### Mode de fonctionnement

- L'utilisation avec une carte Arduino va consister à connecter directement la ligne sur une broche numérique en entrée et la colonne sur une broche numérique en sortie.
- Le programme va « scanner » l'état des broches de lignes en changeant successivement l'état des broches de colonne pour savoir quel est le bouton poussoir actuellement appuyé.

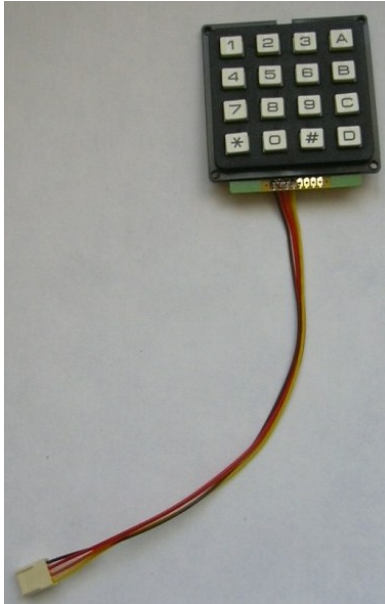
## 9. Rappel : Utilisation d'un clavier matriciel « préparé » avec une carte Arduino : le schéma théorique

- Le principe de l'utilisation du clavier matriciel 4x4 avec une carte Arduino est simple :
  - on va connecter les 4 broches de lignes sur 4 broches numériques en entrée (avec rappel au plus interne activé).
  - on va connecter les 4 broches de colonnes sur 4 broches numériques en sortie
- Pour connaître le bouton appuyé dans une colonne, il suffira de mettre au niveau LOW la broche de colonne et à lire successivement l'état des broches de lignes. La broche de ligne qui sera au niveau LOW indiquera l'appui sur le bouton poussoir d'intersection entre la ligne et la colonne tout simplement.

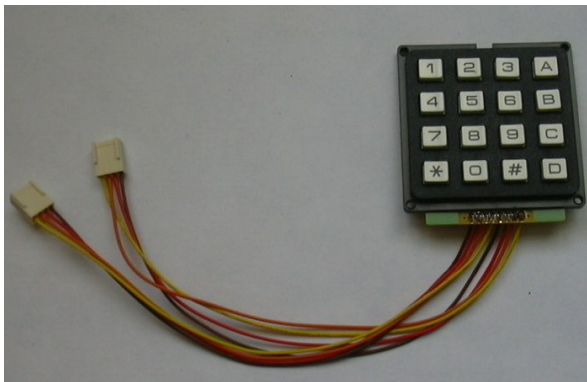


## 10. Préparation d'un clavier matriciel 16 touches (4 lignes x 4 colonnes) et utilisation avec une carte Arduino

- La préparation du clavier matriciel est simple 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 4 broches sur fils sur les 4 broches de colonnes :

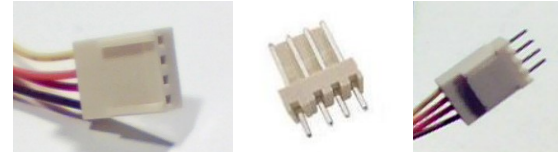


- On soude ensuite de la même façon le connecteur 4 broches sur fils sur les 4 broches de lignes :

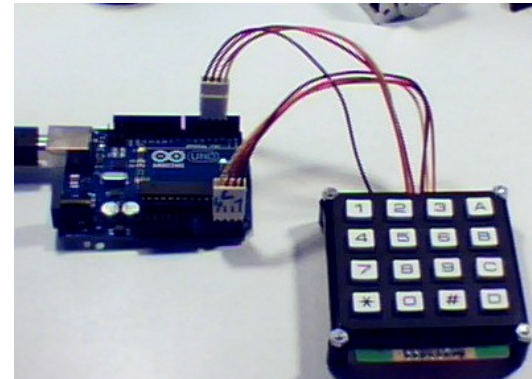


- C'est tout ! Le clavier est prêt pour être utilisé avec une carte Arduino !

- Pour pouvoir connecter facilement les connecteurs sur fils à la carte Arduino, il va falloir utiliser 2 connecteurs droits – 4 contacts que l'on va pouvoir facilement enficher dans les connecteurs femelles de la carte Arduino :



- La connexion avec la carte Arduino se réalise alors très simplement :



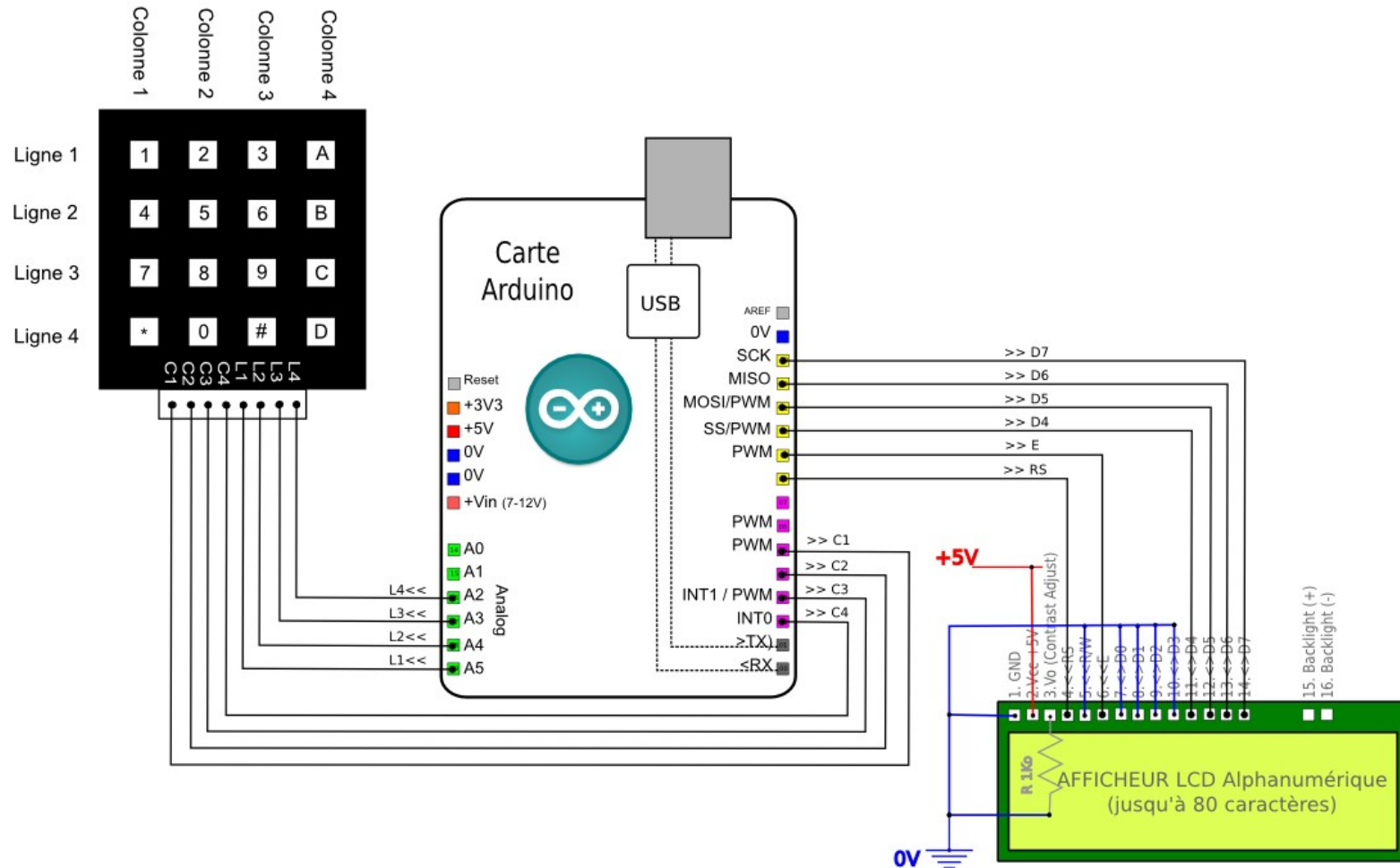
### Note technique

Lors de l'utilisation du clavier, la question se pose de mettre ou non des résistances de limitation de courant sur les broches de colonnes et de lignes... En consultant plusieurs notes d'applications de chez Atmel (fabricant de l'ATmega328 et autres), on retrouve des exemples sans résistances et de plus, dans un cas où des résistances sont utilisées il est indiqué qu'elles ont pour but d'éviter les décharge électro-statiques (ESD). Ainsi, les résistances peuvent-être omises dans la grande majorité des cas.

Par contre, il faut activer le « rappel au plus » interne sur toutes les broches de lignes qui sont en entrée.... ce qui est fait par la librairie keypad lors de l'initialisation.

## 11. Utilisation d'un afficheur LCD et d'un clavier matriciel 4X4 « préparé » avec une carte Arduino : le montage

- Le montage à réaliser pour le clavier matriciel se superpose strictement au schéma théorique :
  - on va connecter les 4 broches de lignes sur 4 broches numériques en entrée
  - on va connecter les 4 broches de colonnes sur 4 broches numériques en sortie
- Le montage à réaliser pour l'afficheur LCD consiste de la même façon 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



## 12. 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...





## 13. Rappel : Installation et présentation d'une librairie de la communauté : la librairie Keypad

### La librairie d'exemple utilisée

- Pour la suite, nous allons utiliser une librairie de la communauté qui permet d'utiliser facilement un clavier matriciel avec Arduino : la librairie Keypad
- Avec cette librairie, on va facilement pouvoir lire l'état du clavier matriciel.

### Télécharger la librairie

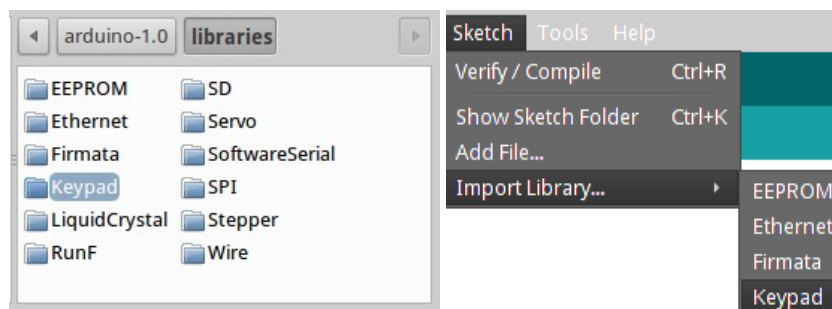
- L'archive est disponible ici :  
<http://arduino.cc/playground/Code/Keypad#Download>

### Documentation de la librairie

- Le site officiel de la librairie est ici :  
<http://arduino.cc/playground/Code/Keypad>
- Un exemple d'utilisation est fourni ici :  
<http://arduino.cc/playground/Code/Keypad#Example>

### Installation

- Télécharger l'archive. au format zip ou autre. L'extraire
- **Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier \*.h ou \*.cpp principal. Corriger au besoin. Ici le nom est Keypad**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino,
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch** > **ImportLibrary** :



### Le constructeur principal

- Le constructeur principal se nomme Keypad et est de la forme :  
`Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES, COLONNES );`

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

### Fonctions de la librairie

- Les fonctions de la librairie Keypad sont les suivantes
  - begin()
  - waitForKey()
  - getKey()
  - getState()
  - keyStateChanged()
  - setHoldTime(unsigned int time)
  - setDebounceTime(unsigned int time)
  - addEventListener(keypadEvent)

### Code d'exemple fourni avec la librairie

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};

byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

## 14. Détecter l'appui sur une touche et afficher le caractère sur le LCD : le programme

### Ce qu'on va faire ici...

- Pour commencer, je vous propose un programme simple : afficher sur l'afficheur LCD le caractère de la touche appuyée sur le clavier. Rien de très compliqué, mais c'est une façon simple de vérifier que tout fonctionne bien.

### Entête déclarative

#### Importation des bibliothèques utilisées

- On commence par importer les bibliothèques **Keypad** et **LiquidCrystal**

#### Configuration du clavier matriciel

- Ensuite on configure le clavier utilisé en définissant :
  - 2 constantes correspondant au nombre de broches et lignes
  - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

**Note : Noter la souplesse avec laquelle on définit le tableau de touches !**

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

#### Configuration de l'afficheur LCD

- 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

// affiche touche appuyée sur afficheur LCD standard

//--- le programme minimum ---
#include <LiquidCrystal.h> // inclusion de la bibliothèque LCD
#include <Keypad.h>

//== configuration de l'afficheur LCD ==

//-- déclaration des broches de l'afficheurs --
const int RS=8; // broche RS
const int E=9; // broche E
const int D4=10; // broche D4
const int D5=11; // broche D5
const int D6=12; // broche D6
const int D7=13; // broche D7

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

//== configuration du clavier matriciel ==

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

## Fonction **setup()**

### Test initial 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()** 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() {

    //----- test initial de l'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()

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

## Fonction **loop()**

### Lecture du clavier matriciel

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.

### Gestion de la touche appuyée

- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO\_KEY**
- Si c'est le cas, on affiche simplement le caractère de la touche sur l'afficheur LCD.

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

    //-- lecture de la touche appuyée
    char touche = clavier.getKey(); // lecture de la touche appuyée

    //--- gestion de la touche appuyée
    if (touche != NO_KEY){ // si une touche a été appuyée

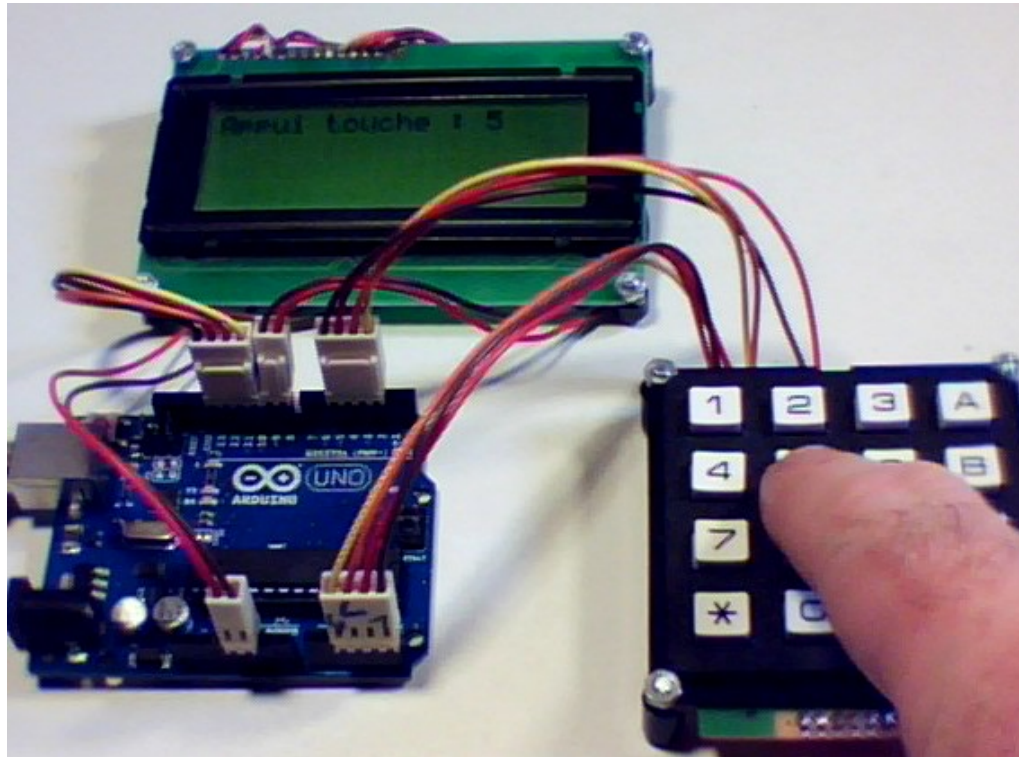
        lcd.setCursor(0,0); // position coin sup gauche
        lcd.print("Appui touche : "); // affiche message
        lcd.print(touche); // affiche le caractère saisi
        lcd.print(" "); // espace "propreté"

    } // fin if touche appuyée

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

## Fonctionnement du programme

- Une fois la carte programmée :
  - appuyer sur des touches du clavier
  - un message indiquant la touche saisie s'affiche



**Bravo : le plus dur est fait !**

Si vous obtenez bien ce message à l'appui sur une touche, vous êtes sûr que tout fonctionne bien !

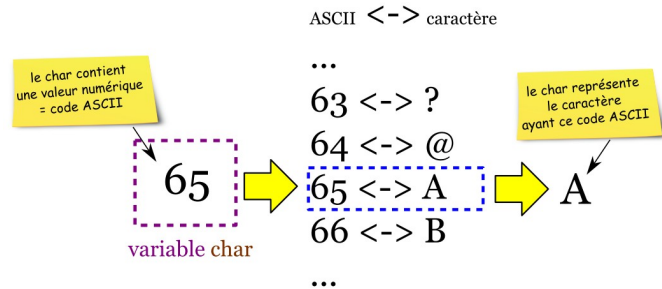
Vous disposez ainsi d'une base matérielle qui va vous permettre de créer toutes sortes d'applications concrètes (saisie de codes, de valeurs, etc...)

A présent, nous allons essentiellement écrire des codes utilisant ce montage pour explorer différentes fonctions utiles concrètement pour la saisie de valeurs.

« A vos marques, prêt... codez ! »

## 15. Récupérer la valeur numérique d'une touche appuyée et affichage sur le LCD : le programme

Ce qu'on va faire ici...



Rappel : Correspondance entre le code ASCII, le type char et le caractère

- Une fois que l'on est capable de récupérer la touche appuyée, il faut pouvoir récupérer la valeur numérique pour l'utiliser.
- Le truc consiste à retrancher la valeur 48 au code ASCII du caractère, comme cela a été vu dans l'atelier consacré à la réception de caractères sur le port série.

### Entête déclarative

#### Importation des bibliothèques utilisées

- On commence par importer les bibliothèques **Keypad** et **LiquidCrystal**

#### Configuration du clavier matriciel

- Ensuite on configure le clavier utilisé en définissant :
  - 2 constantes correspondant au nombre de broches et lignes
  - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche
  - 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

#### Configuration de l'afficheur LCD

- 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.

```
// affiche la valeur de la touche appuyée sur afficheur LCD standard
```

```
/*--- le programme minimum ---
```

```
#include <LiquidCrystal.h> // inclusion de la bibliothèque LCD
```

```
#include <Keypad.h>
```

```
//== configuration de l'afficheur LCD ==
```

```
/*--- déclaration des broches de l'afficheurs --
```

```
const int RS=8; // broche RS
```

```
const int E=9; // broche E
```

```
const int D4=10; // broche D4
```

```
const int D5=11; // broche D5
```

```
const int D6=12; // broche D6
```

```
const int D7=13; // broche D7
```

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

```
//== configuration du clavier matriciel ==
```

```
/*--- définition nombre de lignes et de colonne
```

```
const int LIGNES = 4; // nombre de lignes
```

```
const int COLONNES = 4; // nombre de colonnes
```

```
/* définition des touches
```

```
char touches[LIGNES][COLONNES] = {
```

```
{ '1', '2', '3', 'A' },
```

```
{ '4', '5', '6', 'B' },
```

```
{ '7', '8', '9', 'C' },
```

```
{ '*', '0', '#', 'D' }
```

```
};
```

```
// tableaux de ligne et colonnes - attention : type byte obligatoire !
```

```
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de lignes L1, L2, L3, L4
```

```
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de colonnes C1, C2,C3, C4
```

```
/* création objet Keypad représentant le clavier
```

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES, COLONNES );
```



## Fonction **setup()**

### Test initial 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()** 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**(10).
- 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() {

  //----- test initial de l'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()

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

## Fonction `loop()`

### Lecture du clavier matriciel

- On mémorise le résultat de la fonction `getKey()` dans une variable `char` correspondant à la touche saisie.

### Gestion de la touche appuyée

- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de `NO_KEY`
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type `int` et on l'affiche.
- Si la valeur n'est pas comprise entre 0 et 9 un message indique que la touche appuyée n'est pas un chiffre.

**Un « truc » très utile à retenir : pour les caractères 0 à 9, on obtient la valeur numérique correspondante en faisant (code ASCII – 48 )**

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

    //-- lecture de la touche appuyée
    char touche = clavier.getKey(); // lecture de la touche appuyée

    //-- gestion de la touche appuyée
    if (touche != NO_KEY){ // si une touche a été appuyée

        lcd.setCursor(0,0); // position coin sup gauche
        lcd.print ("Appui sur : ") ; // affiche message
        lcd.print(touche); // affiche le caractère saisi
        lcd.print (" ") ; // espace "propreté"

        int valeur = touche-48;
        if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et
9

            lcd.setCursor(0,1); // position debut premiere ligne
            lcd.print ("valeur = ") ; // affiche message
            lcd.print(valeur); // affiche le caractère saisi
            lcd.print (" ") ; // espace "propreté"

        }
        else {

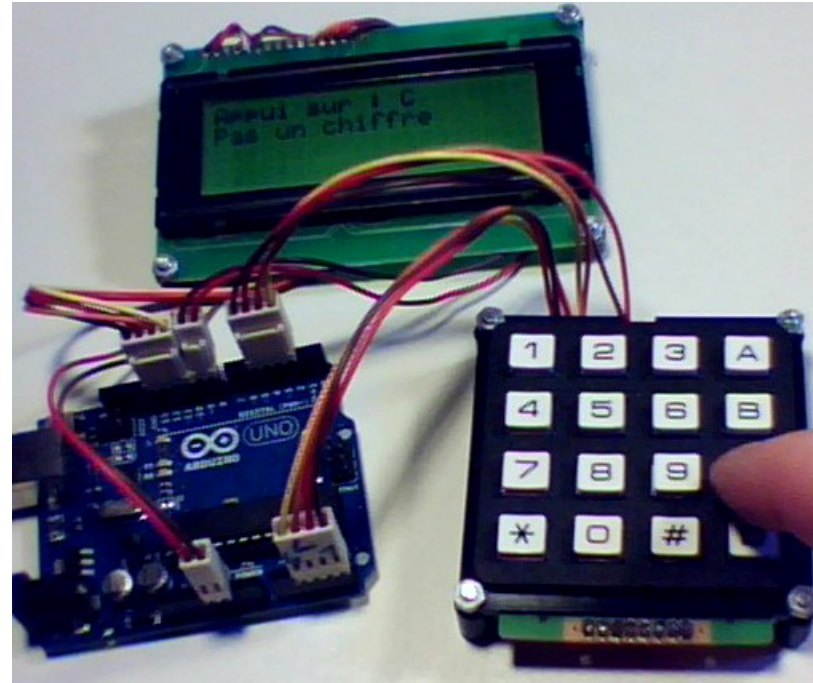
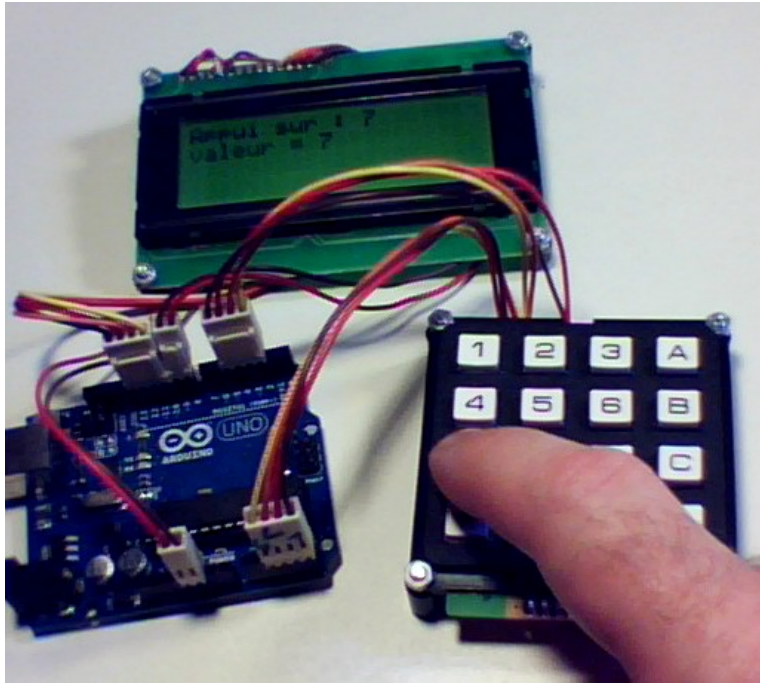
            lcd.setCursor(0,1); // position debut premiere ligne
            lcd.print ("Pas un chiffre ") ; // affiche message

        }

    } // fin if touche appuyée
} // fin de la fonction loop()
```

## Fonctionnement du programme

- Une fois la carte programmée :
  - appuyer sur des touches du clavier
  - un message indique la valeur de la touche si c'est un chiffre
  - ou bien un autre message s'affiche indiquant que la touche n'est pas un chiffre.



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

Atelier Arduino : Utiliser un afficheur LCD alpha-numérique et un clavier matriciel avec Arduino : les bases et manipulation de valeurs entières.

## 16. Décodeur de « code secret » : le programme

### Ce qu'on va faire ici...

Je vous propose ici de reprendre le programme de décodeur de code secret présenté dans l'atelier consacré aux claviers matriciels mais en l'adaptant à l'afficheur LCD. Le cœur du programme est une fonction qui permet de mémoriser une série de touche : il devient possible de vérifier si la chaîne saisie correspond à une chaîne en mémoire. On peut ainsi réaliser un simple décodeur de « code secret » qui pourra permettre de sécuriser l'utilisation d'un montage. Tout bête mais potentiellement pratique. Let's go again!

### Entête déclarative

#### Importation des bibliothèques utilisées

- On commence par importer les bibliothèques **Keypad** et **LiquidCrystal**

#### Configuration du clavier matriciel

- Ensuite on configure le clavier utilisé en définissant :
  - 2 constantes correspondant au nombre de broches et lignes
  - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche
  - 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

#### Configuration de l'afficheur LCD

- 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 des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un **char** pour mémoriser la touche appuyée
- un **int** pour mémoriser la valeur numérique de la touche
- une variable de comptage du nombre de caractères saisis
- un objet String pour stocker la chaîne saisie.
- un objet String pour stocker le code secret.

```
// décodeur de code secret avec clavier + afficheur LCD standard
```

```
///-- le programme minimum ---
```

```
#include <LiquidCrystal.h> // inclusion de la bibliothèque LCD
```

```
#include <Keypad.h>
```

```
///== configuration de l'afficheur LCD ==
```

```
///-- déclaration des broches de l'afficheurs --
```

```
const int RS=8; // broche RS
```

```
const int E=9; // broche E
```

```
const int D4=10; // broche D4
```

```
const int D5=11; // broche D5
```

```
const int D6=12; // broche D6
```

```
const int D7=13; // broche D7
```

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

```
///== configuration du clavier matriciel ==
```

```
///-- définition nombre de lignes et de colonne
```

```
const int LIGNES = 4; // nombre de lignes
```

```
const int COLONNES = 4; // nombre de colonnes
```

```
// définition des touches
```

```
char touches[LIGNES][COLONNES] = {
```

```
{ '1', '2', '3', 'A' },
```

```
{ '4', '5', '6', 'B' },
```

```
{ '7', '8', '9', 'C' },
```

```
{ '*', '0', '#', 'D' }
```

```
};
```

```
// tableaux de ligne et colonnes - attention : type byte obligatoire !
```

```
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de lignes L1, L2, L3, L4
```

```
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de colonnes C1, C2,C3, C4
```

```
// création objet Keypad représentant le clavier
```

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES, COLONNES );
```

```
///-- variables globales utiles ---
```

```
char touche=0; // variable pour stockage touche appuyée
```

```
int valeur=0; // variable pour la valeur numérique de la touche
```

```
int compt=0; // variable comptage caractères saisis
```

```
String chaineSaisie=""; // déclare un objet String vide pour saisie chaîne
```

```
String codeSecret="A1B2C3"; // déclare un objet String pour mémoriser code secret
```

## Fonction **setup()**

### Test initial 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**() 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() {

  //----- test initial de l'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()

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



## Fonction **loop()**

- On affiche un message invitant à saisir le code
- Le résultat de la fonction saisieCode() est mémorisé
- On compare ensuite la chaîne saisie avec le code mémorisé et on affiche le message en conséquence.
- Le programme recommence en effaçant l'écran après quelques secondes...

Remarquer la souplesse d'utilisation des chaînes de caractères **String**, notamment pour la comparaison.

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

    lcd.setCursor(0,0); //1ère colonne, 1ère ligne
    lcd.print("Saisir votre code: ");

    lcd.setCursor(0,1); // 1ère colonne - 2ème ligne
    chaineSaisie=saisieCode();

    if (chaineSaisie==codeSecret) {

        lcd.setCursor(0,2); // 1ère colonne - 3ème ligne
        lcd.print("Code valide ! ");
    } // fin if
    else {

        lcd.setCursor(0,2); // 1ère colonne - 3ème ligne
        lcd.print("Code faux ! ");

    } // fin else

    //--- on attend quelques secondes et on recommence
    delay(2000);

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

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

## Fonction saisieCode()

### Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur la touche #.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
  - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
  - on place tout le code de la saisie des touches dans cette boucle
  - on sort de la boucle while au moment voulu avec l'instruction **break**

### Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO\_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
  - on ajoute le caractère à la chaîne **String**
- Si la touche appuyée est le # :
  - On valide la chaîne saisie
  - On réinitialise les variables
  - On sort de la boucle while avec l'instruction **break**
- Si la valeur n'est pas comprise entre 0 et 9 et n'est pas le # :
  - on ajoute le caractère à la chaîne **String**

### Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie d'une chaîne au clavier -----
String saisieCode() {

    String stringOut="";

    while(true) { // boucle tant que l'on ne sort pas de la boucle while
        // la sortie de la boucle while est provoquée par l'appui sur la touche # = validation

        touche = clavier.getKey(); // lecture de la touche appuyée

        if (touche != NO_KEY){ // si une touche a été appuyée

            lcd.print(touche); // affiche le caractère saisi

            valeur=touche-48; // convertit l'ASCII en valeur numérique

            if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et 9

                compt=compt+1; // incrémente compteur
                stringOut=stringOut+touche; // ajoute le caractère au String

            } // fin if

            else if (touche=='#') { // si appui sur # = validation de la chaîne

                //stringOut=""; //RAZ le String de réception - pas nécessaire ici
                compt=0; // RAZ compteur
                delay(100); // pause
                break; // sort de la boucle while

            } // fin else if

            else { // si le caractère reçu n'est pas un chiffre ni le #

                stringOut=stringOut+touche; // ajoute le caractère au String

            } // fin else

        } // fin if touche appuyée

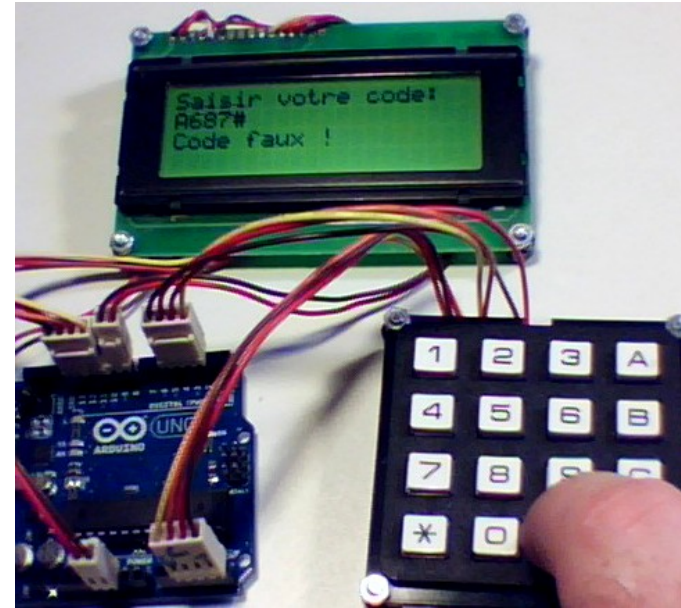
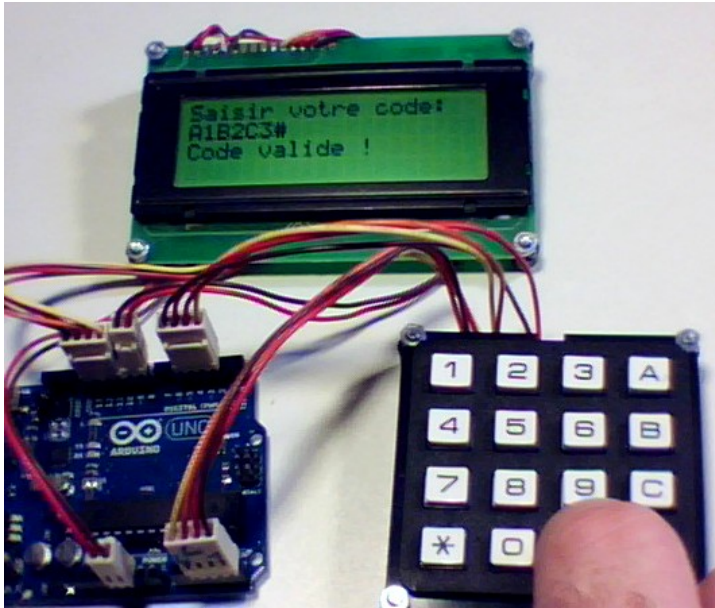
    } // fin while

    return(stringOut); // renvoie la chaîne saisie

} // fin saisie Code
```

## Fonctionnement du programme

- Une fois la carte programmée :
  - saisir le code qui s'affiche sur la 2ème ligne
  - valider avec la touche # : un message indique si le code est valide ou faux.



## 17. Saisie de valeur numérique entière au clavier et affichage sur LCD : le programme

### Ce qu'on va faire ici...

Comme on l'a vu dans l'atelier consacré aux claviers matriciels, une fois que l'on sait recevoir une série d'appui, il n'y a qu'un pas pour obtenir la valeur numérique correspondant à l'appui successif sur plusieurs touches. C'est ce que nous allons faire ici : toutes les touches numériques appuyées seront mémorisées jusqu'à l'appui sur une touche non numérique. A ce moment là, la valeur numérique saisie sera extraite à partir de la chaîne. C'est reparti... !

### Entête déclarative

#### Importation des bibliothèques utilisées

- On commence par importer les bibliothèques **Keypad** et **LiquidCrystal**

#### Configuration du clavier matriciel

- Ensuite on configure le clavier utilisé en définissant :
  - 2 constantes correspondant au nombre de broches et lignes
  - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche
  - 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

#### Configuration de l'afficheur LCD

- 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 des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un **char** pour mémoriser la touche appuyée
- un **int** pour mémoriser la valeur numérique de la touche
- une variable de comptage du nombre de caractères saisis
- un objet String pour stocker la chaîne saisie.
- une variable long pour stocker la valeur numérique saisie

```
// Saisie d'une valeur numérique entière avec clavier + afficheur LCD standard
//--- le programme minimum ---
#include <LiquidCrystal.h> // inclusion de la bibliothèque LCD
#include <Keypad.h>

//== configuration de l'afficheur LCD ==
//--- déclaration des broches de l'afficheurs --
const int RS=8; // broche RS
const int E=9; // broche E
const int D4=10; // broche D4
const int D5=11; // broche D5
const int D6=12; // broche D6
const int D7=13; // broche D7

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

//== configuration du clavier matriciel ==
//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de lignes L1,
L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de colonnes
C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes,
LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche
int compt=0; // variable comptage caractères saisis

String chaineSaisie=""; // déclare un objet String vide pour saisie chaîne
long valeurSaisie =0; // valeur saisie
boolean debug=false;
```

## Fonction **setup()**

### Test initial 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()** 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**(10).
- 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() {

  //----- test initial de l'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()

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

## Fonction **loop()**

### **Saisie des touches**

- On affiche un message invitant à saisir le code
- Le résultat de la fonction saisieCode() est mémorisé

### **Extraction de la valeur numérique**

- Si la chaîne String n'est pas vide, on appelle la fonction stringToLong() qui permet d'extraire la valeur numérique correspondante. Cette fonction est décrite ci-dessous.
- La valeur est mémorisée dans une variable **long** et affichée.
- Si la chaîne est vide, un message est affiché.
- Le programme recommence en effaçant l'écran après quelques secondes...

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

    lcd.setCursor(0,0); //1ère colonne, 1ère ligne
    lcd.print("Saisir un nombre : ");

    lcd.setCursor(0,1); // 1ère colonne - 2ème ligne
    chaineSaisie=saisieChaine();

    //----- on se retrouve ici une fois la saisie terminée

    if (chaineSaisie!="") {
        valeurSaisie=stringToLong(chaineSaisie);

        lcd.setCursor(0,2); // 1ère colonne - 3ème ligne
        lcd.print("Valeur =");
        lcd.print(valeurSaisie);
        chaineSaisie=""; //RAZ le String de réception
    }
    else {
        lcd.setCursor(0,2); // 1ère colonne - 3ème ligne
        lcd.print("Aucune saisie");
    }
    //--- on attend quelques secondes et on recommence
    delay(2000);

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

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



## Fonction saisieChaine()

### Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'une touche non numérique soit appuyée.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
  - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
  - on place tout le code de la saisie des touches dans cette boucle,
  - on sort de la boucle while au moment voulu avec l'instruction **break**.

### Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO\_KEY**
- Si c'est le cas, on récupère la valeur numérique de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
  - on ajoute le caractère à la chaîne **String**
- Si la touche appuyée n'est pas un chiffre : on sort de la boucle **while()**

### Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie d'une chaîne au clavier -----
String saisieChaine() {

    String stringOut="";

    while(true) { // boucle tant que l'on ne sort pas de la boucle while
        // la sortie de la boucle while est provoquée par l'appui sur la
        // touche # = validation

        touche = clavier.getKey(); // lecture de la touche appuyée

        if (touche != NO_KEY){ // si une touche a été appuyée

            lcd.print(touche); // affiche le caractère saisi

            valeur=touche-48; // convertit l'ASCII en valeur numérique

            if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et
9

                compt=compt+1; // incrémente compteur
                stringOut=stringOut+touche; // ajoute le caractère au String

            } // fin if

            else { // si le caractère reçu n'est pas un chiffre

                //stringOut=""; //RAZ le String de réception - pas nécessaire
ici

                compt=0; // RAZ compteur
                delay(100); // pause
                break; // sort de la boucle while

            } // fin else

        } // fin if touche appuyée

    } // fin while

    return(stringOut); // renvoie la chaîne saisie

} // fin saisie Chaîne
```

## Fonction stringToLong()

- Cette fonction :
  - reçoit un **String**
  - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
  - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
  - À l'aide de la fonction `charAt(index)` de la classe String, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
  - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

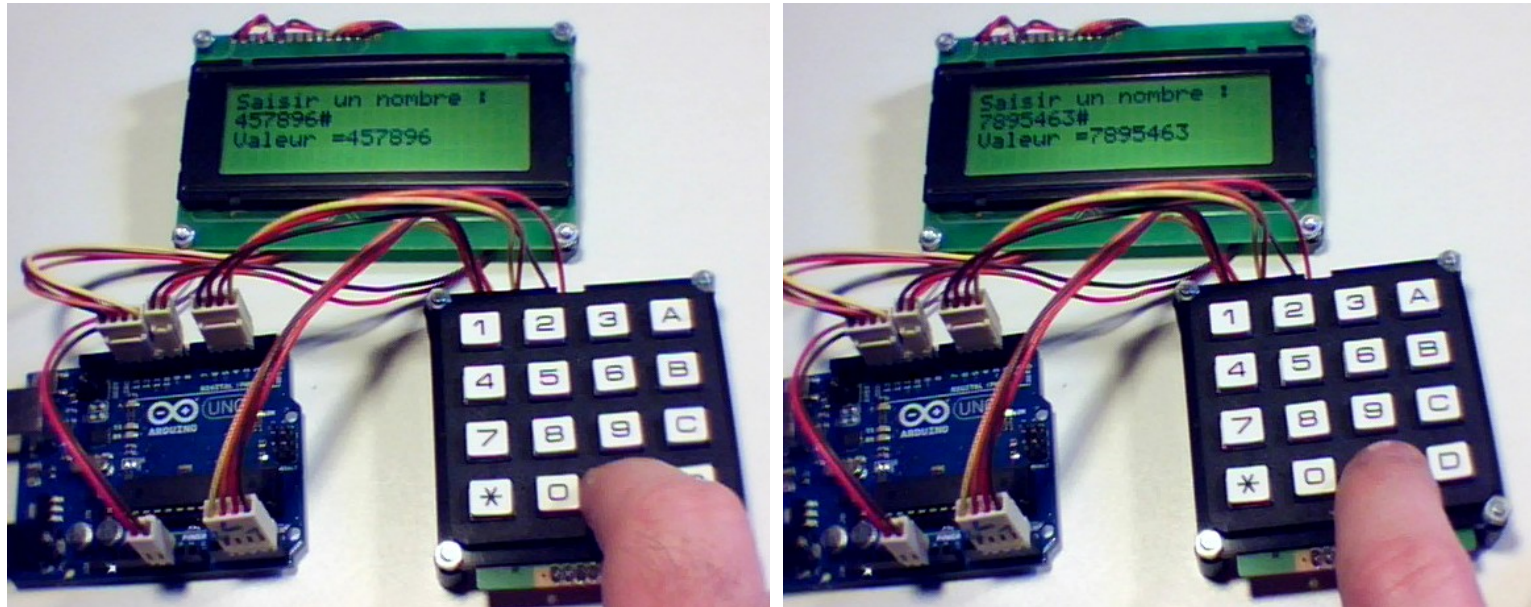
    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

## Fonctionnement du programme

- Une fois la carte programmée :
  - saisir une valeur numérique qui s'affiche sur la 2ème ligne
  - valider avec appui sur une touche non numérique : la valeur numérique est affichée.



**A partir du moment où l'on est capable de saisir une valeur numérique au clavier, de nombreuses applications deviennent possibles** (j'en propose quelques unes dans les pages qui suivent) : minuteur, « goto » de servomoteur ou de moteurs pas à pas, réglage de paramètres divers (vitesse, température max/min, heure,...), etc... et même une calculatrice (just for fun... mais vous serez content de l'avoir fait vous-même !)

A vos idées !

Noter que les fonctions de saisie de la chaîne des touches et de conversion de la chaîne en long pourront être mises dans une librairie séparée (ce que nous présenterons dans un autre atelier...) afin de simplifier l'écriture du code. Ici, je laisse volontairement le code des fonctions utilisées complet dans un but didactique.

## 18. Bases théoriques : le codage décimal / Binaire et Hexadécimal

### Pour comprendre

- Nous utilisons pour compter, 10 signes différents, que l'on appelle « chiffres » et qui permettent de compter comme on en a l'habitude : 0,1,2,3,4,5,6,7,8,9,10,11,12, etc... Si vous réfléchissez bien, vous constaterez que compter, c'est faire défiler les 10 signes des unités, puis à chaque tour faire défiler de un cran le chiffre des dizaines, puis quand les dizaines ont été parcourues, on fait de même avec les centaines, etc... On dit que ce comptage est en base 10.
- Un ordinateur, un circuit numérique n'utilise que 2 états, HAUT ou BAS, 0 ou 1 pour compter. Le principe est le même : on fait défiler les 2 signes des unités » puis des « dizaines », puis des « centaines »... plus exactement en 1ère position, en 2ème position et en 3ème position. Ce qui donne 0,1,10,11,100,101,110,111,1000, etc... C'est le système binaire, ou base 2. Remarquer que le comptage binaire correspond à une division de la fréquence par 2 pour chaque rang de chiffre.
- Pour des raisons pratiques, en informatique, on regroupe parfois un groupe de 4 chiffres xxxx de 0 et de 1 en un symbole unique. Du coup, il faut utiliser 16 symboles pour pouvoir représenter les 16 valeurs possibles (de 0000 à 1111 soit de 0 à 15) . Le comptage se fera sous la forme : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,10, 12, 13, ... On utilise les chiffres 0 à 9 et les lettres A, B, C, D, E, F comme symbole. C'est le système hexadécimal ou base 16.

### Tableau Décimal / Binaire / Hexadécimal

Décimal	Binaire	Hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1110	A
11	1111	B
12	1000	C
13	1001	D
14	1110	E
15	1111	F
16	10000	10
...	...	...
255	11111111	FF
...	...	...
65535	1111111111111111	FFFF

Remarquer comment le comptage hexadécimal « allège » le comptage binaire...

## 19. Convertisseur Décimal / Binaire / Hexadécimal et affichage sur LCD : le programme

### Ce qu'on va faire ici...

Je vous propose ici une simple variante du programme précédent ([seule la fonction loop\(\) est modifiée par rapport au code précédent](#)) qui utilise la possibilité du langage Arduino d'afficher simplement une valeur décimale en une valeur binaire ou hexadécimale. C'est un simple exercice de codage qui pourra vous permettre de vous familiariser avec le codage binaire/ décimal / hexadécimal. Ici, nous utiliserons une valeur comprise entre 0 et 255 pour cette conversion. Y'a plus qu'à !

### Entête déclarative

#### Importation des bibliothèques utilisées

- On commence par importer les bibliothèques **Keypad** et **LiquidCrystal**

#### Configuration du clavier matriciel

- Ensuite on configure le clavier utilisé en définissant :
  - 2 constantes correspondant au nombre de broches et lignes
  - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche
  - 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

#### Configuration de l'afficheur LCD

- 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 des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un **char** pour mémoriser la touche appuyée
- un **int** pour mémoriser la valeur numérique de la touche
- une variable de comptage du nombre de caractères saisis
- un objet String pour stocker la chaîne saisie.
- une variable long pour stocker la valeur numérique saisie

```
// Saisie d'une valeur numérique entière avec clavier + afficheur LCD standard
//--- le programme minimum ---
#include <LiquidCrystal.h> // inclusion de la bibliothèque LCD
#include <Keypad.h>

//== configuration de l'afficheur LCD ==
//-- déclaration des broches de l'afficheurs --
const int RS=8; // broche RS
const int E=9; // broche E
const int D4=10; // broche D4
const int D5=11; // broche D5
const int D6=12; // broche D6
const int D7=13; // broche D7

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

//== configuration du clavier matriciel ==
//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de lignes L1,
L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de colonnes
C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes,
LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche
int compt=0; // variable comptage caractères saisis

String chaineSaisie=""; // déclare un objet String vide pour saisie chaîne
long valeurSaisie =0; // valeur saisie
boolean debug=false;
```

## Fonction **setup()**

### Test initial 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()** 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**(10).
- 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() {

  //----- test initial de l'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()

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



## Fonction **loop()**

### Saisie des touches

- On affiche un message invitant à saisir le code
- Le résultat de la fonction `saisieCode()` est mémorisé

### Extraction de la valeur numérique

- Si la chaîne String n'est pas vide, on appelle la fonction `stringToLong()` qui permet d'extraire la valeur numérique correspondante. Cette fonction est décrite ci-dessous.
- La valeur est mémorisée dans une variable **long**.
- On commence par obliger la valeur à rester entre 0 et 255. Ceci est réalisé à l'aide de l'instruction `constrain()`
- Ensuite, on affiche sur 3 lignes, les 3 valeurs décimale, binaire et hexadécimale correspondantes, en utilisant l'argument **HEX** ou **BIN** passé à la fonction `print()`, tout simplement !
- Si la chaîne est vide, un message est affiché.
- Le programme recommence en effaçant l'écran après quelques secondes...

Une nouvelle fois, le langage Arduino montre sa souplesse en rendant très simple la conversion des affichages décimal, binaire et hexadécimal.

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

    lcd.setCursor(0,0); //1ère colonne, 1ère ligne
    lcd.print("0-255 ? ");

    lcd.blink();
    chaineSaisie=saisieChaine();
    lcd.noBlink();
    //----- on se retrouve ici une fois la saisie terminée

    if (chaineSaisie!="") {
        valeurSaisie=stringToLong(chaineSaisie);

        constrain(valeurSaisie, 0,255); // Oblige valeur entre 0 et 255

        //--- affichage décimal
        lcd.setCursor(0,1); // 1ère colonne - 2ème ligne
        lcd.print("Dec =");
        lcd.print(valeurSaisie);

        //--- affichage binaire
        lcd.setCursor(0,2); // 1ère colonne - 3ème ligne
        lcd.print("Bin =");
        lcd.print(valeurSaisie, BIN);

        //--- affichage hexadécimal
        lcd.setCursor(0,3); // 1ère colonne - 4ème ligne
        lcd.print("Hexa =");
        lcd.print(valeurSaisie, HEX);

        chaineSaisie=""; //RAZ le String de réception
    }
    else {
        lcd.setCursor(0,2); // 1ère colonne - 2ème ligne
        lcd.print("Aucune saisie");
    }
    //--- on attend quelques secondes et on recommence
    delay(5000);

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

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

## Fonction saisieChaine()

### Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'une touche non numérique soit appuyée.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
  - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
  - on place tout le code de la saisie des touches dans cette boucle,
  - on sort de la boucle while au moment voulu avec l'instruction **break**.

### Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO\_KEY**
- On affiche la touche correspondante sur le LCD à l'emplacement courant,
- Si la valeur est comprise entre 0 et 9 :
  - on ajoute le caractère à la chaîne **String**
- Si la touche appuyée n'est pas un chiffre : on sort de la boucle **while()**

### Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie d'une chaîne au clavier -----
String saisieChaine() {

    String stringOut="";

    while(true) { // boucle tant que l'on ne sort pas de la boucle while
        // la sortie de la boucle while est provoquée par l'appui sur la
        // touche # = validation

        touche = clavier.getKey(); // lecture de la touche appuyée

        if (touche != NO_KEY){ // si une touche a été appuyée

            lcd.print(touche); // affiche le caractère saisi

            valeur=touche-48; // convertit l'ASCII en valeur numérique

            if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et
9

                compt=compt+1; // incrémente compteur
                stringOut=stringOut+touche; // ajoute le caractère au String

            } // fin if

            else { // si le caractère reçu n'est pas un chiffre

                //stringOut=""; //RAZ le String de réception - pas nécessaire
ici

                compt=0; // RAZ compteur
                delay(100); // pause
                break; // sort de la boucle while

            } // fin else

        } // fin if touche appuyée

    } // fin while

    return(stringOut); // renvoie la chaîne saisie

} // fin saisie Chaîne
```

## Fonction stringToLong()

- Cette fonction :
  - reçoit un **String**
  - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
  - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
  - À l'aide de la fonction `charAt(index)` de la classe `String`, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
  - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

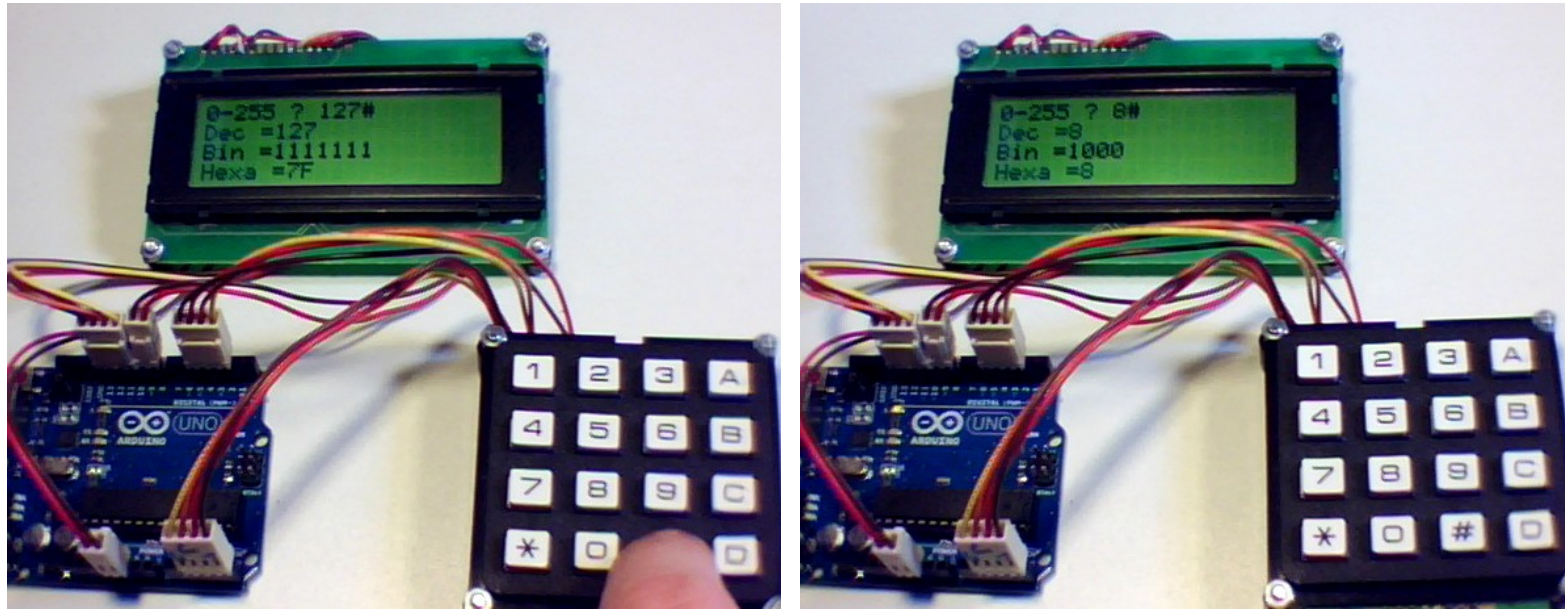
    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

## Fonctionnement du programme

- Une fois la carte programmée :
  - saisir une valeur numérique qui s'affiche sur la première ligne
  - valider avec appui sur une touche non numérique : les 3 valeurs numériques décimales, binaires et hexadécimales s'affichent sur 3 lignes.



« Just for fun ! » mais sympa de l'avoir fait soi-même non ?

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

**Structure**

**Variables et constantes**

**Fonctions**

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)

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



# Table des matières

Utiliser un afficheur LCD alpha-numérique et un clavier matriciel avec Arduino.

Intro |

Matériel nécessaire pour les ateliers Arduino |

Matériel spécifique nécessaire pour utiliser un afficheur LCD dans cet atelier |

Matériel spécifique nécessaire pour utiliser un clavier matriciel dans 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 |

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

Rappel : Fiche technique : clavier matriciel 16 touches (4 lignes x 4 colonnes) |

Rappel : Utilisation d'un clavier matriciel « préparé » avec une carte Arduino : le schéma théorique |

Préparation d'un clavier matriciel 16 touches (4 lignes x 4 colonnes) et utilisation avec une carte Arduino |

Utilisation d'un afficheur LCD et d'un clavier matriciel 4X4 « préparé » avec une carte Arduino : le montage |

Rappel : Langage Arduino : la librairie LiquidCrystal pour le contrôle des afficheurs LCD standards |

Rappel : Installation et présentation d'une librairie de la communauté : la librairie Keypad |

Détecter l'appui sur une touche et afficher le caractère sur le LCD : le programme |

Récupérer la valeur numérique d'une touche appuyée et affichage sur le LCD : le programme |

Décodeur de « code secret » : le programme |

Saisie de valeur numérique entière au clavier et affichage sur LCD : le programme |

Bases théoriques : le codage décimal / Binaire et Hexadécimal |

Convertisseur Décimal / Binaire / Hexadécimal et affichage sur LCD : le programme |

Saisie de valeur numérique à virgule au clavier et affichage sur LCD : le programme |

Calculatrice Arduino avec saisie au clavier et affichage sur LCD : le programme |

« Goto » pour servomoteur avec saisie au clavier et affichage sur LCD : le montage |

« Goto » pour servomoteur avec saisie au clavier et affichage sur LCD : 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)