# LES AFFICHEURS LCD

## Préambule

Nous avons vu précédemment comment assurer une communication entre un Arduino et un PC.

Mais maintenant, il est intéressant d'afficher des informations sans avoir besoin d'un ordinateur.

Avec les écrans LCD, nous allons pouvoir afficher du texte sur un écran qui n'est pas très coûteux et ainsi faire des projets autonomes vis-à-vis des PC !

# LES AFFICHEURS LCD

## Présentation d'un afficheur LCD

LCD signifie « Liquid Crystal Display » et se traduit, en français, par « Écran à Cristaux Liquides. Ces écrans sont PARTOUT !

Attention, il y'a une différence avec des écrans à LED. Il en existe deux types :

1. les écrans à rétro-éclairage LED : ceux sont des écrans LCD tout à fait ordinaires qui ont simplement la particularité d'avoir un rétro-éclairage à LED à la place des tubes néons. Leur prix est du même ordre de grandeur que les LCD « normaux ». En revanche, la qualité d'affichage des couleurs semble meilleure comparés aux LCD « normaux ».

1. les écrans à affichage LED : ceux si ne disposent pas de rétro-éclairage et ne sont ni des écrans LCD, ni des plasma. Ce sont des écrans qui, en lieu et place des pixels, se trouvent des LED de très petite taille. Leur coût est prohibitif pour le moment, mais la qualité de contraste et de couleur inégale tous les écrans existants !

# LES AFFICHEURS LCD

## Fonctionnement de l'écran

Comme son nom l'indique, un écran LCD possède des cristaux liquides. Mais ce n'est pas tout ! En effet, pour fonctionner il faut plusieurs choses. Si vous regardez de très près votre écran (éteint) vous pouvez voir une grille de carré. Ces carrés sont appelés des pixels (de l'anglais « Picture Element », soit « Élément d'image » en français).

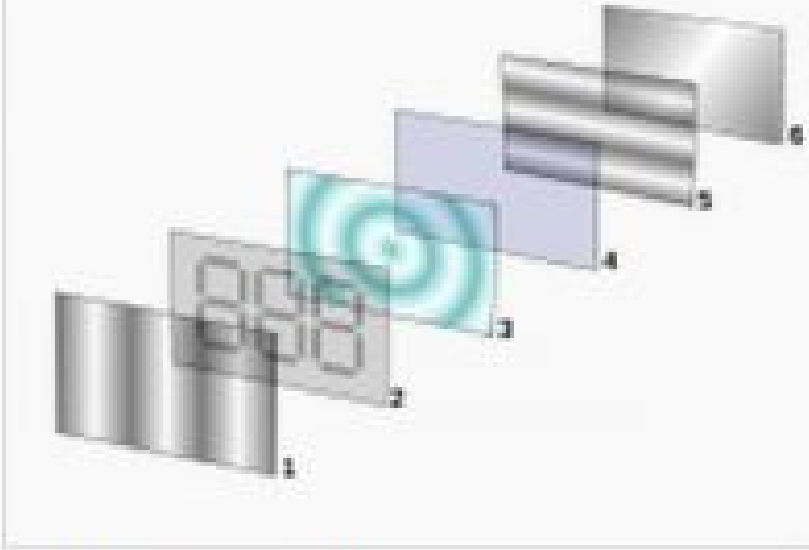
Chaque pixel est un cristal liquide. Lorsque aucun courant ne le traverse, ses molécules sont orientées dans un sens (admettons, 0°). En revanche lorsqu'un courant le traverse, ses molécules vont se tourner dans la même direction (90°). Voilà pour la base.

# LES AFFICHEURS LCD

## Fonctionnement de l'écran

Mais pourquoi il y a de la lumière dans un cas et pas dans l'autre ?
Tout simplement parce que cette lumière est **polarisée**. Cela signifie que la lumière est orientée dans une direction. En effet, entre les cristaux liquides et la source lumineuse se trouve un filtre polariseur de lumière.

Ce filtre va orienter la lumière dans une direction précise. Entre vos yeux et les cristaux se trouve un autre écran polariseur, qui est perpendiculaire au premier. Ainsi, il faut que les cristaux liquides soient dans la bonne direction pour que la lumière passe de bout en bout et revienne à vos yeux. Enfin, vient le rétro-éclairage (fait avec des LED) qui vous permettra de lire l'écran même en pleine nuit (sinon il vous faudrait l'éclairer pour voir le contraste).



Afficheur 3 chiffres
1 et 5 : filtres polarisants ;
2 : électrodes avant ;
4 : électrode arrière ;
3 : cristaux liquides ;
6 : miroir.

# LES AFFICHEURS LCD

## Commande du LCD

Normalement, pour pouvoir afficher des caractères sur l'écran il nous faudrait activer individuellement chaque pixel de l'écran. Un caractère est représenté par un bloc de 7*5 pixels. Ce qui fait qu'un écran de 16 colonnes et 2 lignes représente un total de 16*2*7*5 = 1120 pixels ! Heureusement c'est la tâche est plus simplifiée.

**Le décodeur de caractères**

Tout comme il existe un driver vidéo pour votre carte graphique d'ordinateur, il existe un driver « LCD » pour votre afficheur.

Ce composant va servir à décoder un ensemble « simple » de bits pour afficher un caractère à une position précise ou exécuter des commandes comme déplacer le curseur par exemple.

Ce composant est fabriqué principalement par *Hitachi* et se nomme le **HC44780**. Il sert de **décodeur de caractères**. Ainsi, plutôt que de devoir multiplier les signaux pour commander les pixels un à un, il nous suffira d'envoyer des octets de commandes pour lui dire « écris moi 'LICENCE' à partir de la colonne 3 sur la ligne 1 ».

Ce composant possède 16 broches que je vais brièvement décrire :

# LES AFFICHEURS LCD

| N° | Nom | Rôle |
|---|---|---|
| 1 | VSS | Masse |
| 2 | Vdd | +5V |
| 3 | V0 | Réglage du contraste |
| 4 | RS | Sélection du registre (commande ou donnée) |
| 5 | R/W | Lecture ou écriture |
| 6 | E | Entrée de validation |
| 7 à 14 | D0 à D7 | Bits de données |
| 15 | A | Anode du rétroéclairage (+5V) |
| 16 | K | Cathode du rétroéclairage (masse) |

# LES AFFICHEURS LCD

Normalement, pour tous les écrans LCD (non graphiques) ce brochage est le même. Donc pas d'inquiétude lors des branchements, il vous suffira de vous rendre sur cette page pour consulter le tableau.
Par la suite, les broches utiles qu'il faudra relier à l'Arduino sont les broches 4, 5 (facultatives), 6 et les données (7 à 14 pouvant être réduite à 8 à 14) en oubliant pas l'alimentation et la broche de réglage du contraste. Ce composant possède tout le système de traitement pour afficher les caractères. Il contient dans sa mémoire le schéma d'allumage des pixels pour afficher chacun d'entre eux. Voici la table des caractères affichables :

# LES AFFICHEURS LCD

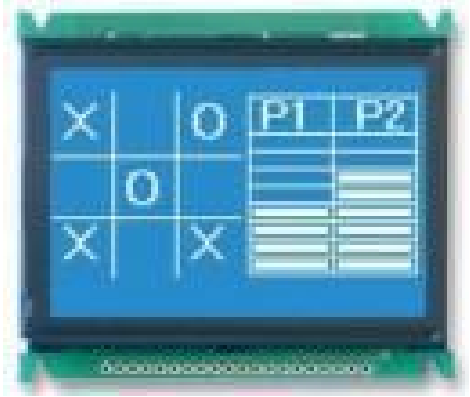
# LES AFFICHEURS LCD

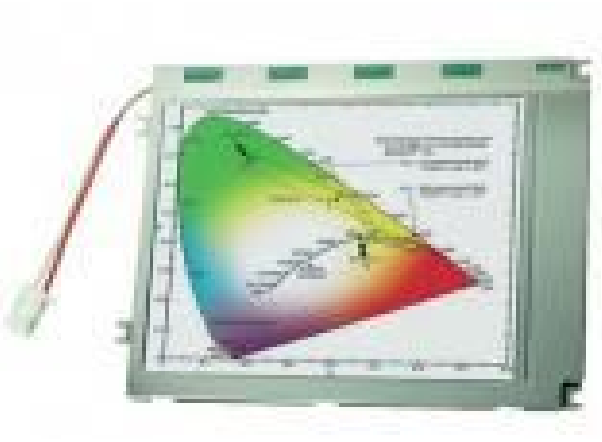
## Les caractéristiques d'un afficheur

Dans la grande famille afficheur LCD, on distingue plusieurs catégories :
Les afficheurs alphanumériques
Les afficheurs graphiques monochromes
Les afficheurs graphiques couleur

Les premiers sont les plus courants. Ils permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les caractères sont prédéfinis (voir table juste au-dessus) et on a donc aucunement besoin de gérer chaque pixel de l'écran. Les seconds sont déjà plus avancés. On a accès à chacun des pixels et on peut donc produire des dessins beaucoup plus évolués. Ils sont cependant légèrement plus onéreux que les premiers. Les derniers sont l'évolution des précédents, la couleur en plus (soit 3 fois plus de pixels à gérer : un sous-pixel pour le rouge, un autre pour le bleu et un dernier pour le vert, le tout forme la couleur d'un seul pixel). Pour le TP on se servira d'afficheur de la première catégorie car ils suffisent à faire de nombreux montages et restent accessibles pour des zéros.

# LES AFFICHEURS LCD

Afficheur alphanumérique Afficheur graphique (monochrome) Afficheur graphique (couleur)

# LES AFFICHEURS LCD

## Communication avec l'écran

### La communication parallèle

De manière classique, on communique avec l'écran de manière **parallèle**. Cela signifie que l'on envoie des bits par blocs, en utilisant plusieurs broches en même temps (opposée à une transmission série où les bits sont envoyés un par un sur une seule broche).
Nous utilisons 10 broches différentes, 8 pour les données (en parallèle donc) et 2 pour de la commande (E : Enable et RS : Registre Selector). La ligne R/W peut être connecté à la masse si l'on souhaite uniquement faire de l'écriture. Pour envoyer des données sur l'écran, c'est en fait assez simple. Il suffit de suivre un ordre logique et un certain timing pour que tout se passe bien. Tout d'abord, il nous faut placer la broche RS à 1 ou 0 selon que l'on veut envoyer une commande, comme par exemple « déplacer le curseur à la position (1;1) » ou que l'on veut envoyer une donnée : « écris le caractère 'a' « . Ensuite, on place sur les 8 broches de données (D0 à D7) la valeur de la donnée à afficher. Enfin, il suffit de faire une impulsion d'au moins 450 ns pour indiquer à l'écran que les données sont prêtes.

# LES AFFICHEURS LCD

## Communication avec l'écran

### La communication parallèle

Cependant, comme les ingénieurs d'écrans sont conscients que la communication parallèle prend beaucoup de broches, ils ont inventé un autre mode que j'appellerai « semi-parallèle ». Ce dernier se contente de travailler avec seulement les broches de données D4 à D7 (en plus de RS et E) et il faudra mettre les quatre autres (D0 à D3) à la masse. Il libère donc quatre broches. Dans ce mode, on fera donc deux fois le cycle « envoi des données puis impulsion sur E » pour envoyer un octet complet.

Nous utiliserons une libraire nommée **LiquidCrystal** qui se chargera de gérer les timings et l'ensemble du protocole.
Pour continuer, le mode « semi-parallèle » sera choisi. Il nous permettra de garder plus de broches disponibles pour de futurs montages et est souvent câblé par défaut dans de nombreux shields (dont le mien). La partie suivante vous montrera ce type de branchement.

# LES AFFICHEURS LCD

## Communication avec l'écran

### La communication Série

Lorsque l'on ne possède que très peu de broches disponibles sur notre Arduino, il peut être intéressant de faire appel à un composant permettant de communiquer par voie série avec l'écran. Un tel composant se chargera de faire la conversion entre les données envoyées sur la voie série et ce qu'il faut afficher sur l'écran. Le gros avantage de cette solution est qu'elle nécessite seulement un seul fil de donnée (avec une masse et le VCC) pour fonctionner là où les autres méthodes ont besoin de presque une dizaine de broches.

En effet, elle nous permet de garder l'approche « standard » de l'écran et nous permet de garder la liaison série pour autre chose (encore que l'on pourrait en émuler une sans trop de difficulté).

# LES AFFICHEURS LCD

## Communication avec l'écran

### La communication $I^2C$

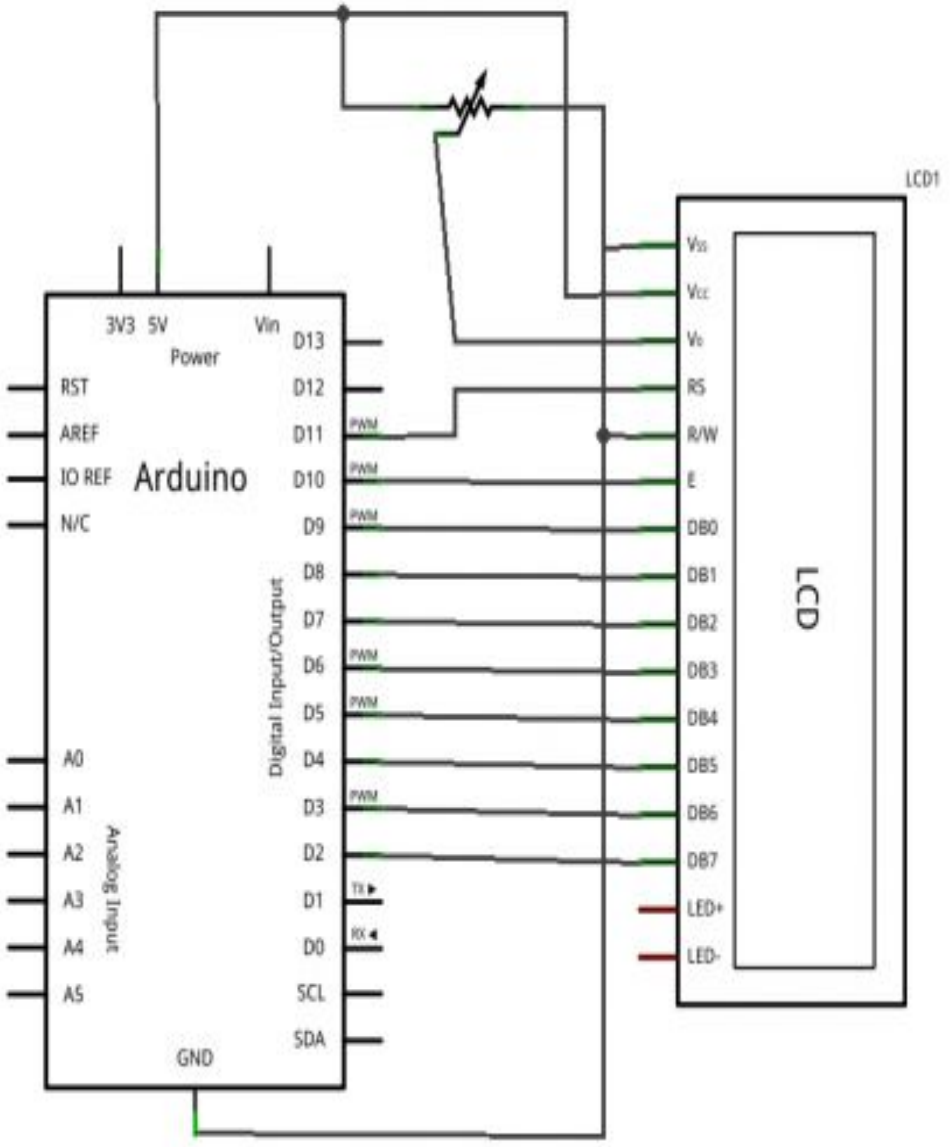
Un dernier point à voir, c'est la communication de la carte Arduino vers l'écran par la liaison $I^2C$. Cette liaison est utilisable avec seulement 2 broches (une broche de donnée et une broche d'horloge) et nécessite l'utilisation de deux broches analogiques de l'Arduino (broche 4 et 5).

# LES AFFICHEURS LCD

## Le branchement

L'afficheur LCD utilise 6 à 10 broches de données ((D0 à D7) ou (D4 à D7) + RS + E) et deux d'alimentations (+5V et masse). La plupart des écrans possèdent aussi une entrée analogique pour régler le contraste des caractères. Nous brancherons dessus un potentiomètre de 10 kOhms. Les 10 broches de données peuvent être placées sur n'importe quelles entrées/sorties numériques de l'Arduino. En effet, nous indiquerons ensuite à la librairie LiquidCrystal qui est branché où.

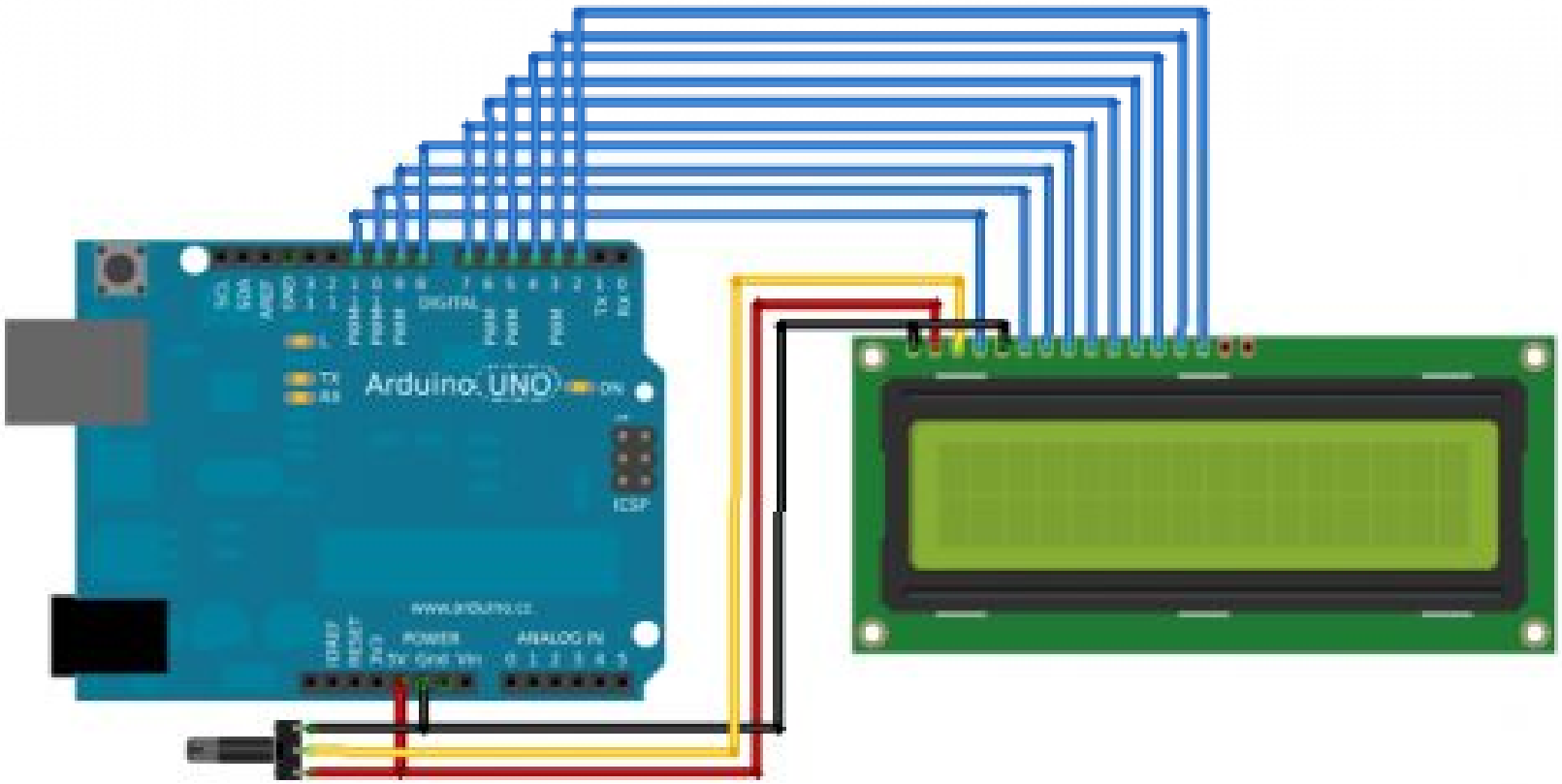
**Le montage à 8 broches de données**



# LES AFFICHEURS LCD

## Le branchement

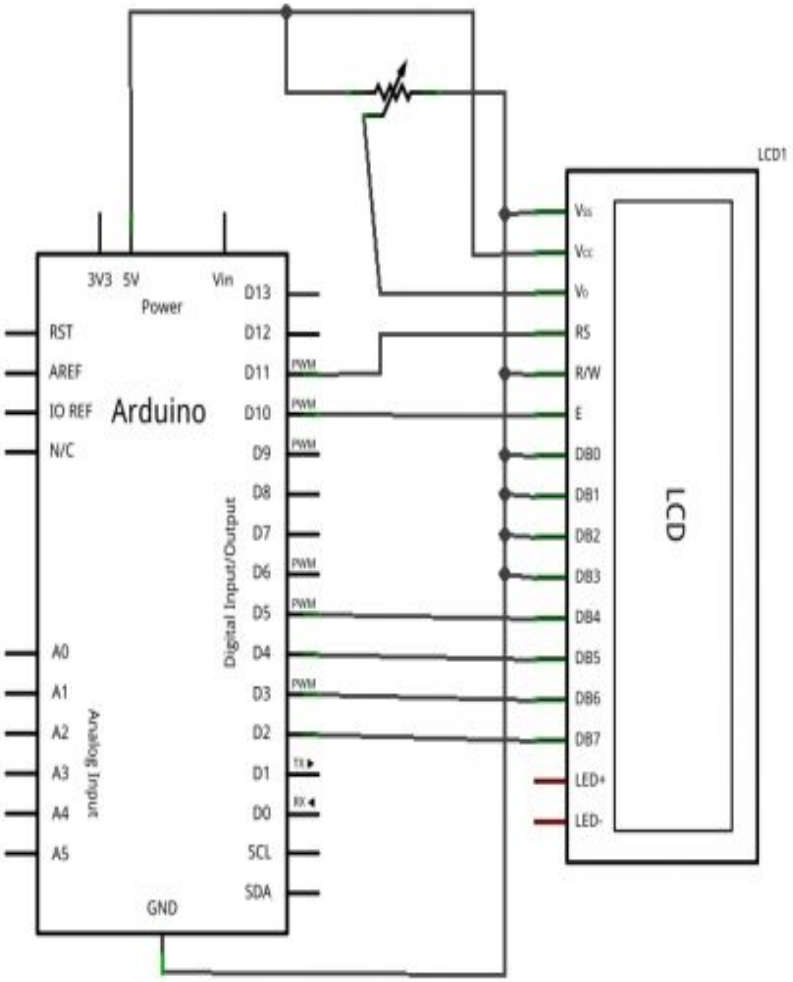
**Le montage à 8 broches de données**

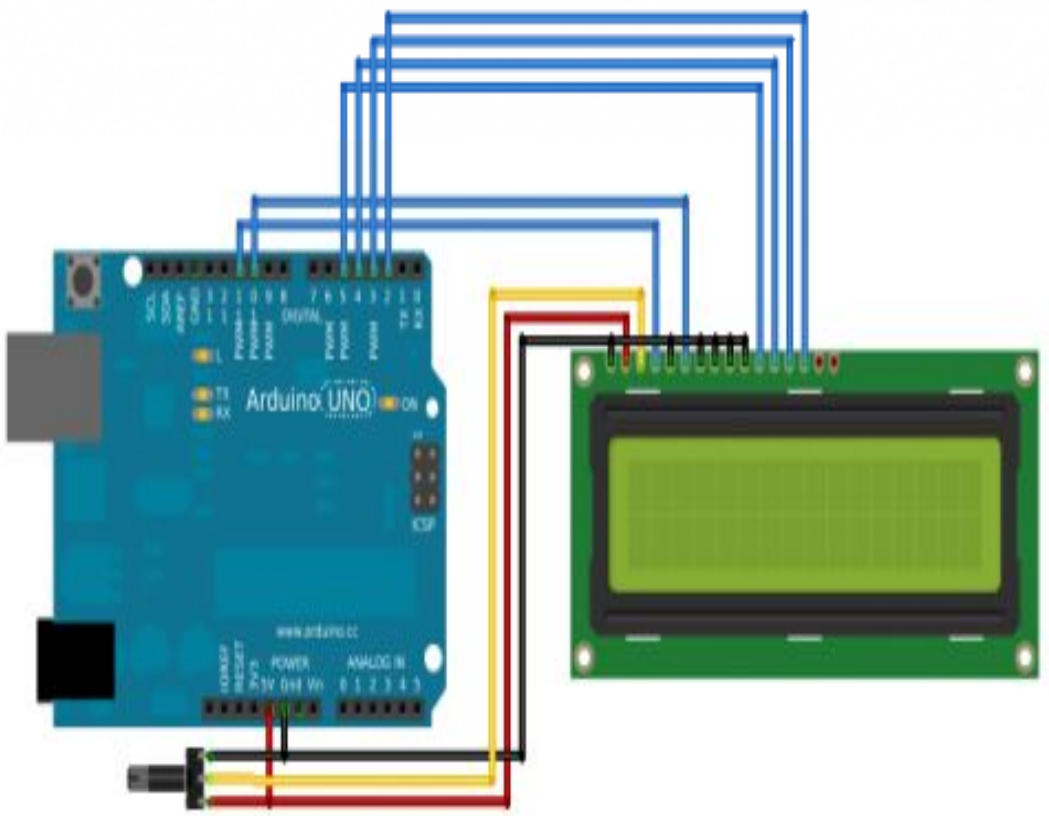


# LES AFFICHEURS LCD

## Le branchement

### Le montage à 4 broches de données



# LES AFFICHEURS LCD

## Le démarrage de l'écran avec Arduino

Comme écrit plus tôt, nous allons utiliser la librairie « LiquidCrystal ». Pour l'intégrer c'est très simple, il suffit de cliquer sur le menu « Import Library » et d'aller chercher la bonne. Une ligne #include « LiquidCrystal.h » doit apparaitre en haut de la page de code (les prochaines fois vous pourrez aussi taper cette ligne à la main directement, ça aura le même effet). Ensuite, il ne nous reste plus qu'à dire à notre carte Arduino où est branché l'écran (sur quelles broches) et quelle est la taille de ce dernier (nombre de lignes et de colonnes). Nous allons donc commencer par déclarer un objet (c'est en fait une variable évoluée, plus de détails dans la prochaine partie) lcd, de type LiquidCrystal et qui sera global à notre projet. La déclaration de cette variable possède plusieurs formes.

LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7) où rs est le numéro de la broche où est branché « RS », « enable » est la broche « E » et ainsi de suite pour les données.

LiquidCrystal(rs, enable, d4, d5, d6, d7) (même commentaires que précédemment

Ensuite, dans le setup() il nous faut démarrer l'écran en spécifiant son nombre de **colonnes** puis de **lignes**. Cela se fait grâce à la fonction begin(cols,rows). Voici un exemple complet de code correspondant aux deux branchements précédents :

# LES AFFICHEURS LCD

## Un exemple de programme

```
#include "LiquidCrystal.h" //ajout de la librairie

//Vérifier les broches !
LiquidCrystal lcd(11,10,9,8,7,6,5,4,3,2); //liaison 8 bits de données
LiquidCrystal lcd(11,10,5,4,3,2); //liaison 4 bits de données

void setup()
{
  lcd.begin(16,2); //utilisation d'un écran 16 colonnes et 2 lignes
  lcd.write("Salut les Etudiants Télécoms !"); //petit test pour vérifier que tout marche
}

void loop() {}
```

# LES AFFICHEURS LCD

## Un exemple de programme

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

// initialise l'écran avec les bonnes broches

// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !

LiquidCrystal lcd(8,9,4,5,6,7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Salut les Etudiants en Télécoms!");
}

void loop() {

}
```

# LES AFFICHEURS LCD

## Un exemple complet de programme (une Horloge)

Il s'agit de réaliser une petite horloge. Bien entendu elle ne sera pas fiable du tout car nous n'avons aucun repère réel dans le temps, mais ça reste un bon exercice.

L'objectif sera donc d'afficher le message suivant : « Il est hh:mm:ss » avec 'hh' pour les heures, 'mm' pour les minutes et 'ss' pour les secondes.

Une dernière chose avant de commencer. Si vous tentez de faire plusieurs affichages successifs, le curseur ne se replacera pas et votre écriture sera vite chaotique.

Cette fonction vous permet de revenir à la position en haut à gauche de l'écran : home().

Il vous suffira de faire un lcd.home() pour replacer le curseur en haut à gauche. Nous reparlerons de la position curseur dans le chapitre suivant !

# LES AFFICHEURS LCD

## Un exemple complet de programme (une Horloge)

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

// initialise l'écran avec les bonnes broches
// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !
LiquidCrystal lcd(8,9,4,5,6,7);

int heures,minutes,secondes;
char message[16] = "";

void setup()
{
   lcd.begin(16, 2); // règle la taille du LCD : 16 colonnes et 2 lignes

   //changer les valeurs pour démarrer à l'heure souhaitée !
   heures = 0;
   minutes = 0;
   secondes = 0;
}
```

# LES AFFICHEURS LCD

## Un exemple complet de programme (une Horloge)

```
void loop()
{
   //on commence par gérer le temps qui passe...
   if(secondes == 60) //une minutes est atteinte ?
   {

 secondes = 0; //on recompte à partir de 0
       minutes++;
   }
   if(minutes == 60) //une heure est atteinte ?
   {
      minutes = 0;
      heures++;
   }
   if(heures == 24) //une journée est atteinte ?
   {
      heures = 0;
   }
```

# LES AFFICHEURS LCD

## Un exemple complet de programme (une Horloge)

```
//met le message dans la chaine à transmettre
   sprintf(message,"Il est %2d:%2d:%2d",heures,minutes,secondes);

   lcd.home();          //met le curseur en position (0;0) sur l'écran

   lcd.write(message);   //envoi le message sur l'écran

   delay(1000);         //attend une seconde
   //une seconde s'écoule...
   secondes++;
}
```

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

Les deux fonctions permettent de rendre le texte invisible ou visible :
noDisplay() : fait disparaître le texte
display() : fait apparaître le texte (s'il y en a évidemment)

Si vous tapez le code suivant, vous verrez le texte clignoter toutes les secondes :

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

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

// initialise l'écran avec les bonnes broches
// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !
LiquidCrystal lcd(8,9,4,5,6,7);

void setup() {
   // règle la  taille du LCD
   lcd.begin(16, 2);
   lcd.print("Hello World !");
}

void loop() {
   lcd.noDisplay();
   delay(500);
   lcd.display();
   delay(500);
}
```

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

Une autre fonction utile est celle vous permettant de nettoyer l'écran. Contrairement à la précédente, cette fonction va supprimer le texte de manière permanente. Pour le ré-afficher il faudra le renvoyer à l'afficheur. Cette fonction au nom évident est : clear(). Le code suivant vous permettra ainsi d'afficher un texte puis, au bout de 2 secondes, il disparaitra (pas de loop(), pas nécessaire) :

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

// initialise l'écran avec les bonnes broches
// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !
LiquidCrystal lcd(8,9,4,5,6,7);

void setup() {
   // règle la  taille du LCD
   lcd.begin(16, 2);
   lcd.print("Hello World !");
   delay(2000);
   lcd.clear();
}
```

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

### Déplacer le texte à la main

Pour commencer, nous allons déplacer le texte manuellement, vers la droite ou vers la gauche.. Après avoir écrit du texte sur l'écran, on peut faire appel aux fonctions scrollDisplayRight() et scrollDisplayLeft() vous pourrez déplacer le texte d'un carré vers la droite ou vers la gauche. S'il y a du texte sur chacune des lignes avant de faire appel aux fonctions, c'est le texte de chaque ligne qui sera déplacé par la fonction. Utilisez deux petits boutons poussoirs pour utiliser le code suivant. Vous pourrez déplacer le texte en appuyant sur chacun des poussoirs !

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

### Déplacer le texte à la main

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

//les branchements
const int boutonGauche = 2; //le bouton de gauche
const int boutonDroite = 3; //le bouton de droite

// initialise l'écran avec les bonnes broches
// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !
LiquidCrystal lcd(8,9,4,5,6,7);

//--------------------------------------------------------------------------

void setup() {
   //règlage des entrées/sorties
   pinMode(boutonGauche, INPUT);
   pinMode(boutonDroite, INPUT);
```

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

### Déplacer le texte à la main

```
  //on attache des fonctions aux deux interruptions externes (les boutons)
  attachInterrupt(0, aDroite, RISING);
  attachInterrupt(1, aGauche, RISING);
  //paramètrage du LCD
  lcd.begin(16, 2); // règle la  taille du LCD
  lcd.print("Hello les Télécoms !");
}
void loop() {
  //pas besoin de loop pour le moment
}
//fonction appelée par l'interruption du premier bouton
void aGauche() {
  lcd.scrollDisplayLeft(); //on va à gauche !
}
//fonction appelée par l'interruption du deuxième bouton
void aDroite() {
  lcd.scrollDisplayRight(); //on va à droite !
}
```

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

### Déplacer le texte automatiquement

De temps en temps, il peut être utile d'écrire toujours sur le même pixel et de faire en sorte que le texte se décale tout seul (pour faire des effets zolis par exemple). Un couple de fonctions va nous aider dans cette tâche. La première sert à définir la direction du défilement. Elle s'appelle leftToRight() pour aller de la gauche vers la droite et rightToLeft() pour l'autre sens. Ensuite, il suffit d'activer (ou pas si vous voulez arrêter l'effet) avec la fonction autoScroll() (et noAutoScroll() pour l'arrêter). Pour mieux voir cet effet, je vous propose d'essayer le code qui suit. Vous verrez ainsi les chiffres de 0 à 9 apparaitre et se « pousser » les uns après les autres :

# LES AFFICHEURS LCD

## Des fonctions utiles pour gérer l'affichage

### Déplacer le texte automatiquement

```
#include <LiquidCrystal.h> //on inclut la librairie
// ATTENTION, REMPLACER LES NOMBRES PAR VOS BRANCHEMENTS À VOUS !
LiquidCrystal lcd(8,9,4,5,6,7);
void setup()
{
   lcd.begin(16, 2);
   lcd.setCursor(14,0);
   lcd.leftToRight();  //indique que le texte doit être déplacer vers la gauche
   lcd.autoscroll();   //rend automatique ce déplacement
   lcd.print("{");
   int i=0;
   for(i=0; i&lt;10; i++)
   {
      lcd.print(i);
      delay(1000);
   }
   lcd.print("}");
}
```