# Biblioteca Lyquid Crystal

*(Es aconsejable ensayar en su hardware)*

*Fuente: http://manueldelgadocrespo.blogspot.com/p/biblioteca-lyquid-crystal.html*

Esta biblioteca permite a una placa Arduino controlar las pantallas LiquidCrystal (LCD) en base a la Hitachi HD44780 chipset (o compatible), que se encuentra en la mayoría de las pantallas LCD basadas en texto. La biblioteca trabaja tanto en el modo de 4 como en 8 bits (es decir, usando 4 u 8 líneas de datos, además de los rs, enable y, opcionalmente, las líneas de control rw).

**FUNCIONES**

## LiquidCrystal()

**Descripción:**
Crea una variable de tipo LiquidCrystal. La pantalla se puede controlar por medio de 4 u 8 líneas de datos. En el primer caso, omitir los números de patas D0 a D3 y dejar esas líneas sin conectar. El pin RW puede estar conectado a tierra en lugar de conectarse a un pin en el Arduino; si es así, omite de los parámetros de esta función.

**Sintaxis**:
LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

**Parámetros**:
**rs**: el número depin de Arduino que está conectado al pin RS en la pantalla LCD
**rw**: el número de pin de Arduino que está conectado al pin de RW en la pantalla LCD (opcional)
**enable**: el número de pin de Arduino que está conectado al pin enable en la pantalla LCD
**d0**, **d1**, **d2**, **d3**, **d4**, **d5**, **d6**, **d7**: los números de los pines Arduino que están conectados a los correspondientes pines de datos en la pantalla LCD.

d0, d1, d2, d3 y son opcionales; Si se omite, la pantalla LCD se puede controlar utilizando sólo las cuatro líneas de datos (d4, d5, d6, d7).

**Ejemplo**:

```
#include <LiquidCrystal.h>

 LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

 void setup()
 {
   lcd.begin(16,1);
   lcd.print("Hola, mundo!");
 }

 void loop() {}
```

---

## begin()

**Descripción**:
Inicializa la interfaz de la pantalla LCD, y especifica las dimensiones (anchura y altura) de la pantalla. begin () debe ser llamado antes de cualquier otro comando de la biblioteca LCD.

**Sintaxis**:
lcd.begin(cols, rows)

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal
**cols**: el número de columnas que tiene la pantalla
**rows**: el número de filas que tiene la pantalla

---

## clear()

**Descripción**:
Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda.

**Sintaxis**:
lcd.clear()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

---

# home()

**Descripción**:
Pone el cursor en la esquina superior izquierda de la pantalla LCD. Es decir, utiliza esa ubicación en la salida de texto subsiguiente a la pantalla. Para borrar también la pantalla, utilice la función **clear**() en su lugar.

**Sintaxis**:
lcd.home()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

---

# setCursor()

**Descripción**;
Coloca el cursor del LCD; es decir, establece la ubicación en la que se mostrará el texto escrit a continuación en la LCD.

**Sintaxis**:
lcd.setCursor(col, row)

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal
**col**: la columna en la que para posicionar el cursor (siendo 0 la primera columna)
**row**: la fila en la que para posicionar el cursor (siendo 0 la primera fila)

---

# write()

**Descripción**:
Escribe un carácter en la pantalla LCD.

**Sintaxis**:
lvd.write(data)

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal
**data**: el carácter a escribir en la pantalla.

---

# print()

**Descripción**:
Imprime un texto en la LCD

**Sintaxis**:
lcd.print(data)
lcd.print(data, BASE)

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystallos
**data**: los datos a imprimir (char, byte, int, long o string)
**BASE** (opcional): la base en la que imprimir números: BIN para binario (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

**Retornos**:
**byte**
print () devolverá el número de bytes escritos, la lectura de ese número es opcional

**Ejemplo**:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup()
{
  lcd.print("hello, world!");
}

void loop() {}
```

---

# cursor() y noCursor()

**Descripción**:
**cursor()**; Muestra el cursor del LCD: como un guión bajo (línea) en la posición a la que se escribirá el siguiente carácter.
**noCursor()**: Muestra el cursor de forma normal.

**Sintaxis**:
lcd.cursor()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

**Ejemplo**:

```
// incluye la biblioteca :
 #include <LiquidCrystal.h>

 // inicializa la biblioteca con los números de los pines de la interfaz
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

 void setup() {
   // establecer el número de LEDs de columnas y filas:
   lcd.begin(16, 2);
   // Imprime un mensaje en la LCD.
   lcd.print("Hola, mundo!");
 }

 void loop() {
   // Apaga el cursor:
   lcd.noCursor();
   delay(500);
   // Enciende el cursor:
   lcd.cursor();
   delay(500);
 }
```

---

# blink() y noBlink()

**Descripción**:
**blink()**: Muestra el cursor LCD parpadeante.
**noBlink()**: Muestra el cursor fijo.
Si se utiliza en combinación con la función **cursor** (), el resultado dependerá de la pantalla en particular.

**Descripción**:
lcd.blink()
noBlink()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

**Ejemplo**:

```
// incluye la biblioteca :
 #include <LiquidCrystal.h>

 // initializa la bibñioteca con los numeros de pines de la interfaz
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

 void setup() {
   // establece el número de LEDs de columnas y filas:
   lcd.begin(16, 2);
   // Imprime un mensaje en la LCD.
   lcd.print("Hola, mundo!");
 }

 void loop() {
   // Apaga el parpadeo delcursor:
   lcd.noBlink();
   delay(3000);
   // Enciende el parpadeo del cursor:
   lcd.blink();
   delay(3000);
 }
```

---

# display() y noDisplay()

**Descripción**:
**display()**: Enciende la pantalla LCD, después de haber sido apagada con **noDisplay**().
**noDisplay()**: Apaga la pantalla LCD, después de haber sido encendida con **display**().
Esto, en ambos casos, restaurará el texto (y el cursor) que estaba en la pantalla.

**Sintaxis**:
lcd.display
lcd.noDisplay()()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

**Ejemplo**:
```
// incluye la biblioteca :
 #include <LiquidCrystal.h>

 // inicializa la bublioteca con los numeros de pines de la interfz
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

 void setup() {
   // establece el número de LEDs de columnas y filas:
   lcd.begin(16, 2);
   // Imprime un mensaje en la LCD.
   lcd.print("Hola, mundo!");
 }

 void loop() {
   // Apaga la pantalla:
   lcd.noDisplay();
   delay(500);
   // Enciende la pantalla:
   lcd.display();
   delay(500);
 }
```

---

# scrollDisplayLeft() y scrollDisplayRight()

**Descripción**:
**scrollDisplayLeft()**: Desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.
**scrollDisplayRight()**: Desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la derecha
.
**Sintaxis**:
scrollDisplayLeft()
scrollDisplayRight()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

**Ejemplo**:
```
// incluye la biblioteca :
#include <LiquidCrystal.h>

// inicializa la bublioteca con los numeros de pines de la interfz
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // establece el número de LEDs de columnas y filas:
  lcd.begin(16, 2);
  // Imprime un mensaje en la LCD.
  lcd.print("Hola, mundo!");
}

void loop() {
  // Desplaza 13 posiciones (longitud de cadena) a la izquierda
  // moviendo el texto hacia fuera de la pantalla a la izquierda:
  for (int positionCounter = 0; positionCounter < 13; positionCounter++) {
    // desplaza una posicion a la izquierda:
    lcd.scrollDisplayLeft();
    // espera un poco tiempo:
    delay(150);
  }

  // Desplaza 29 posiciones (longitud de cadena) a la derechs
  // moviendo el texto hacia fuera de la pantalla a la derecha:
```

```
  for (int positionCounter = 0; positionCounter < 29; positionCounter++) {
    // desplaza una posicion a la derecha:
    lcd.scrollDisplayRight();
    // espera un poco tiempo:
    delay(150);
  }

  // desplaza 16 posiciones (longitud de la pantalla + longitud de la cadena) a la
izquierda
  // devolviendo el texto al centro:
  for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
    // desplaza una posicion a la izquierda:
    lcd.scrollDisplayLeft();
    // espera un poco tiempo:
    delay(150);
  }

  // temporiza al final del bucle completo:
  delay(1000);

}
```

---

# autoscroll() y noAutoscroll()

**Descripción**:
**autoscroll()**: Activa el desplazamiento automático de la pantalla LCD. Esto hace que cada salida de caracteres de la pantalla empuje los caracteres anteriores por un espacio. Si la dirección del texto actual es de izquierda a derecha (el valor predeterminado), la pantalla se desplaza hacia la izquierda; si la dirección actual es de derecha a izquierda, la pantalla se desplaza a la derecha. Esto tiene el efecto de la salida de cada nuevo carácter en la misma ubicación en la pantalla LCD.
**noAutoscroll()**: Desactiva el desplazamiento automático de la pantalla LCD.

**Sintaxis**:
lcd.autoscroll()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

---

# leftToRight() y rightToLeft()

**Descripción**:
**leftToRight():** Establece la dirección del texto escrito para la pantalla LCD de izquierda a derecha, el valor predeterminado. Esto significa que los caracteres escritos posteriores en la pantalla irán de izquierda a derecha, pero no afecta el texto previamente escrito.
**rightToLeft():** Establece la dirección del texto escrito para la pantalla LCD de derecha a izquierda (por defecto es de izquierda a derecha. Esto significa que los caracteres escritos posteriores en la pantalla irán de derecha a izquierda, pero no afecta el texto previamente escrito.

**Sintaxis**:
lcd.leftToRight()
lcd.rightToLeft()

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal

---

createChar()

**Descripción**:
Crea un carácter personalizado (gylph) para su uso en la pantalla LCD. Hasta ocho caracteres de 5x8 píxeles son compatibles (numerados de 0 a 7). La apariencia de cada carácter se especifica mediante una serie de ocho bytes, uno para cada fila. Los cinco bits menos significativos de cada byte determinan los píxeles de la fila. Para mostrar un carácter personalizado en la pantalla, write() su número.

NB: Cuando se hace referencia al carácter personalizado "0", si no es en una variable, necesita para su emisión como un byte, de lo contrario el compilador genera un error. Véase el siguiente ejemplo.

**Sintaxis**:
lcd.createChar(num, data)

**Parámetros**:
**lcd**: una variable de tipo LiquidCrystal
**num**: el carácter a crear crear (0 a 7)
**data**: datos de píxeles del carácter

**Ejemplo**:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

byte smiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};

void setup() {
  lcd.createChar(0, smiley);
  lcd.begin(16, 2);
  lcd.write(byte(0));
}

void loop() {}
```

---