**Liquid Crystal Display**

The [**Liquid Crystal Librar**](https://www.arduino.cc/en/Reference/LiquidCrystal)**y** allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This example sketch shows how to use the display() and noDisplay() methods to turn on and off the display. The text to be displayed will still be preserved when you use noDisplay() so it's a quick way to blank the display without losing everything on it.

**Hardware Required**

* Arduino or Genuino Board
* LCD Screen (compatible with Hitachi HD44780 driver)
* pin headers to solder to the LCD display pins
* 10k ohm potentiometer
* 220 ohm resistor
* hook-up wires
* breadboard

**Circuit**

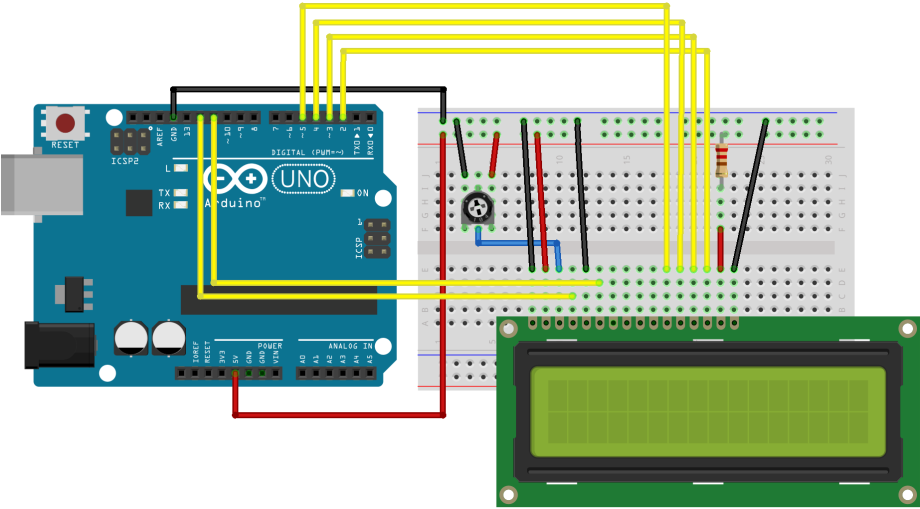
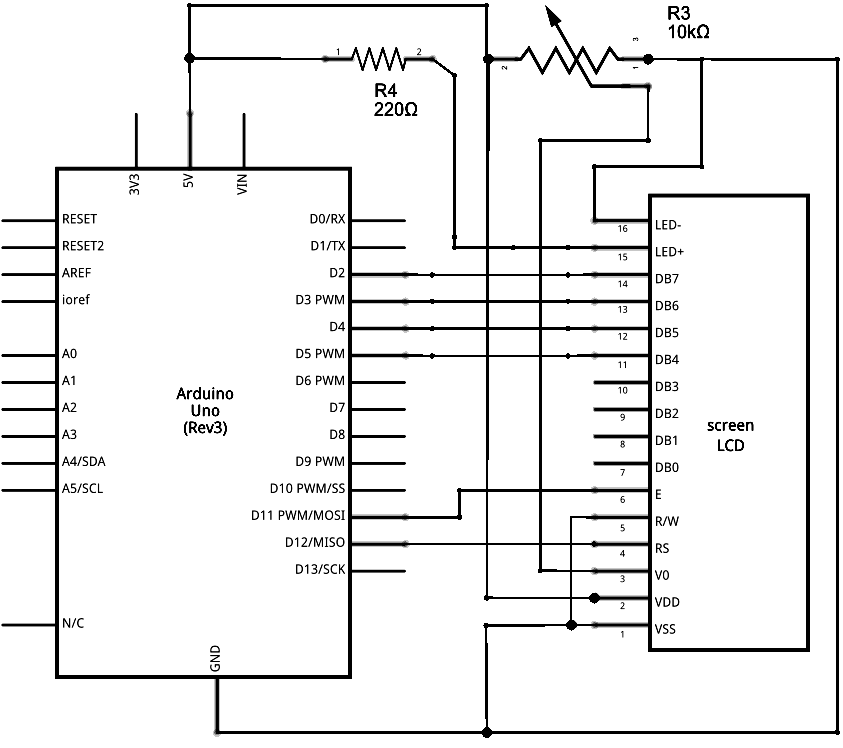
Before wiring the LCD screen to your Arduino or Genuino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image above.  
To wire your LCD screen to your board, connect the following pins:

* RS pin = digital pin 12
* Enable pin = digital pin 11
* D4 pin = digital pin 5
* D5 pin = digital pin 4
* D6 pin = digital pin 3
* D7 pin = digital pin 2

For an LCD –

* 1 = VSS / GND– Connected to gnd
* 2 = VDD / VCC– Connected to 5V
* 3 = V0 / Contrast Setter Pin – acts as a variable in potentiometer
* 4 = RS – Register Select pin -- enables a user to select the instruction mode or the character mode of a LCD.
* Depending on which mode is selected, the data on the 8 data pins (D0-D7) is treated either as instruction or character data.
* **Instruction Mode**
* Instruction Mode- RS pin set to low (or ground)
* When the instruction mode is selected, the data that the LCD receives on the 8 data pins (D0 to D7) will be interpreted as instruction data. **These can be instructions such as "clear the display", "move the cursor", etc.** Depending on which pins are high and low on the 8 data pins determines the instruction that the LCD will be sent and, thus, execute. For example, when in instruction mode, t**he LCD will clear the entire screen display when fed the voltages 00000001 on its 8 data pins.**
* **Character Mode**
* Character Mode- RS pin set to high (or Vcc)
* When the character mode is selected, the data that the LCD receives on the 8 data pins (D0 to D7) will be interpreted as character data. Depending on which pins are high and low on the 8 data pins determines the character(s) which will be shown on the LCD. For example, when in character mode, the LCD will display the "$" (dollar sign) when the data pins are fed 01000010.
* 5 = R/W – **Connected to gnd** // Read or Write pin in an lcd -- The Read/Write (RW) pin (pin 5) of a HD44780 LCD is the pin which either enables a user to write to the LCD so that it displays characters (**write mode**) or to read the characters from the LCD so that are on it (**read mode**). -- **The dominant majority of the time, the RW pin will be set to Write mode**. In some rare cases, we may need to read from the LCD what it is displaying. **In such cases, the RW pin will be set high**. **However, the vast majority of the time, the RW pin will be set low, since we are concerned about writing characters to the LCD.**
* 6 = E // Enable pin
* 7 = DB0 / D0
* 8 = DB1 / D1
* 9 = DB2 / D2
* 10 = DB3 / D3
* 11 = DB4 / D4
* 12 = DB5 / D5
* 13 = DB6 / D6
* 14 = DB7 / D7
* 15 = LED+ – **Connected to 5V through 220 ohm resistor** // Controls the backlight of the lcd
* 16 = LED- – Connected to gnd // Controls the backlight of the lcd

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

# Functions in Liquid Crystal Library

## LiquidCrystal()

#### Description

Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, **omit the pin numbers for d0 to d3 and leave those lines unconnected**. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

#### Syntax

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)

## begin()

#### Description

Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands.

#### Syntax

lcd.begin(cols, rows)

## clear()

#### Description

Clears the LCD screen and positions the cursor in the upper-left corner.

#### Syntax

lcd.clear()

## home()

#### Description

Positions the cursor in the upper-left of the LCD. That is, use that location in outputting subsequent text to the display. To also clear the display, use the [clear()](https://www.arduino.cc/en/Reference/LiquidCrystalClear) function instead.

#### Syntax

lcd.home()

## setCursor()

#### Description

Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

#### Syntax

lcd.setCursor(col, row)

## write()

#### Description

Write a character to the LCD.

#### Syntax

lcd.write(data)

#### Parameters

lcd: a variable of type LiquidCrystal

data: the character to write to the display

#### Returns

byte  
write() will return the number of bytes written, though reading that number is optional

#### Example

#include <LiquidCrystal.h>  
  
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);  
  
void setup()  
{  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  if (Serial.available()) {  
    lcd.write(Serial.read());  
  }  
}

## print()

#### Description

Prints text to the LCD.

#### Syntax

lcd.print(data)   
lcd.print(data, BASE)

#### Parameters

lcd: a variable of type LiquidCrystal

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

#### Returns

byte  
print() will return the number of bytes written, though reading that number is optional

#### Example

#include <LiquidCrystal.h>  
  
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);  
  
void setup()  
{  
  lcd.print("hello, world!");  
}

## cursor()

#### Description

Display the LCD cursor: an underscore (line) at the position to which the next character will be written.

#### Syntax

lcd.cursor()

## noCursor()

#### Description

Hides the LCD cursor.

#### Syntax

lcd.noCursor()

## blink()

#### Description

Display the blinking LCD cursor. If used in combination with the [cursor](https://www.arduino.cc/en/Reference/LiquidCrystalCursor)() function, the result will depend on the particular display.

#### Syntax

lcd.blink()

## noBlink()

#### Description

Turns off the blinking LCD cursor.

#### Syntax

lcd.noBlink()

## display()

#### Description

Turns on the LCD display, after it's been turned off with [noDisplay](https://www.arduino.cc/en/Reference/LiquidCrystalNoDisplay)(). This will restore the text (and cursor) that was on the display.

#### Syntax

lcd.display()

## noDisplay()

#### Description

Turns off the LCD display, without losing the text currently shown on it.

#### Syntax

lcd.noDisplay()

## scrollDisplayLeft()

#### Description

Scrolls the contents of the display (text and cursor) one space to the left.

#### Syntax

lcd.scrollDisplayLeft()

## scrollDisplayRight()

#### Description

Scrolls the contents of the display (text and cursor) one space to the right.

#### Syntax

lcd.scrollDisplayRight()

## autoscroll()

#### Description

Turns on automatic scrolling of the LCD. This causes each character output to the display to push previous characters over by one space. If the current text direction is left-to-right (the default), the display scrolls to the left; if the current direction is right-to-left, the display scrolls to the right. This has the effect of outputting each new character to the same location on the LCD.

#### Syntax

lcd.autoscroll()

## leftToRight()

#### Description

Set the direction for text written to the LCD to left-to-right, the default. This means that subsequent characters written to the display will go from left to right, but does not affect previously-output text.

#### Syntax

lcd.leftToRight()

## rightToLeft()

#### Description

Set the direction for text written to the LCD to right-to-left (the default is left-to-right). This means that subsequent characters written to the display will go from right to left, but does not affect previously-output text.

#### Syntax

lcd.rightToLeft()

## createChar()

#### Description

**Create a custom character (gylph) for use on the LCD. Up to eight characters of 5x8 pixels are supported (numbered 0 to 7). The appearance of each custom character is specified by an array of eight bytes, one for each row. The five least significant bits of each byte determine the pixels in that row. To display a custom character on the screen,** [**write**](https://www.arduino.cc/en/Reference/LiquidCrystalWrite)**() its number.**

**NB : When referencing custom character "0", if it is not in a variable, you need to cast it as a byte, otherwise the compiler throws an error. See the example below.**

#### Syntax

lcd.createChar(num, data)

#### Parameters

lcd: a variable of type LiquidCrystal

num: which character to create (0 to 7)

data: the character's pixel data

#### Example

#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() {}

* Character Lcd's can be interfaced in 8-bit and 4-bit Mode. In 8-bit mode the 8-bit ascii value of the character is send on the data line of lcd. In 4-bit mode only four data pins of lcd are utilized (DB7-DB4). In 4-bit mode first the 8-bit ascii value is divided in to two nibbles, first the upper nibble is send on data line and then the lower nibble. 4-bit mode is used when we want to save GPIO pins of our controller device

# For 16X1 lcd display:

Using Arduino to test the 16x1 modules and check the functions; I found that the 16x2 code needed to be modified because the IC looked at the 16x1 LCD as two lines and all you see is the first eight segments of the display.  
  
This is the code for the 16x1 14 and 16 pin LCD.  
  
/\*   
  
LiquidCrystal Library - Hello World  
  
Demonstrates the use a 16x1 LCD display; the LiquidCrystal library works with all LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface however these are 16x1 LCD, 14 pin interface.   
  
This sketch prints "Hello World!" to the 16x1 LCD.  
  
\*/  
  
// include the library code:  
#include <LiquidCrystal.h>  
  
// initialize the library with the numbers of the interface pins  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
  
void setup() {  
  
// set up the LCD's number of columns and rows:  
  lcd.begin(8, 2);     //is 16x1, adressed as 8x2  
  lcd.setCursor(0,1);  //init right hand side  
  lcd.home();          //back to start  
  lcd.clear();  
  // Print a message to the LCD.  
  lcd.print("hello wo");  //print left side  
  lcd.setCursor(0,1);     //go to right  
  lcd.print("rld!");      //print right side  
  //blinking cursor  
  lcd.cursor();  
  lcd.blink();  
  
}  
  
void loop() {  
  
}