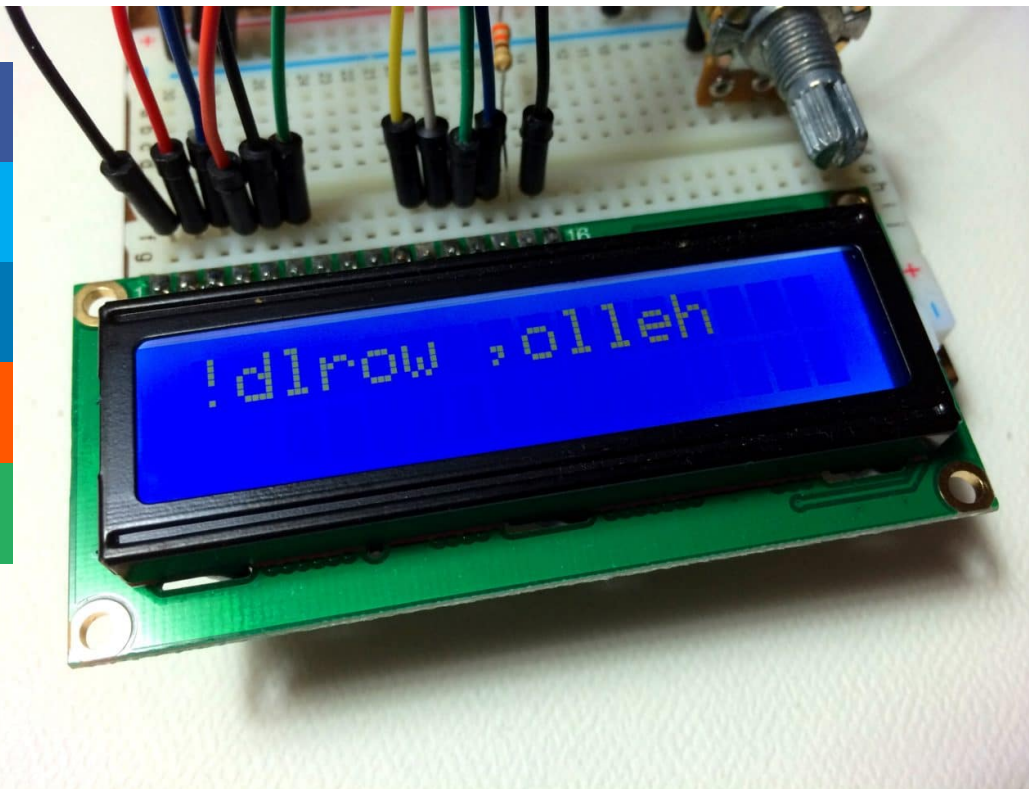


ARDUINO LCD SET UP AND PROGRAMMING GUIDE

Posted by Scott Campbell | Arduino | 57



In this tutorial, I'll explain how to set up an LCD on an [Arduino](#) and show you all the different ways you can program it. I'll show you how to print text, scroll text, make custom characters, blink text, and position text. They're great for any project that outputs data, and they can make your project a lot more interesting and interactive.

SEARCH ...

FOLLOW US

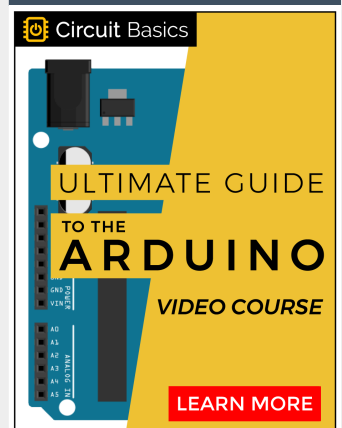


SUBSCRIBE

Get new tutorials sent to your inbox!

EMAIL ADDR

SUBSCRIBE



PCBWay HIGH-QUALITY PCB

ONLY \$5 FOR 10 PIECES

- Rogers, HDI, aluminum and rigid-flex PCB are available now
- Production time 24 hours

PCB ASSEMBLY
Free shipping + Free stencil

ONLY \$30

- Component sourcing
- Quality assurance



The display I'm using is a **16×2 LCD display** that I bought for about \$5. You may be wondering why it's called a 16×2 LCD. The part 16×2 means that the LCD has 2 lines, and can display 16 characters per line. Therefore, a 16×2 LCD screen can display up to 32 characters at once. It is possible to display more than 32 characters with scrolling though.

The code in this article is written for LCD's that use the standard Hitachi HD44780 driver. If your LCD has 16 pins, then it probably has the Hitachi HD44780 driver. These displays can be wired in either 4 bit mode or 8 bit mode. Wiring the LCD in 4 bit mode is usually preferred since it uses four less wires than 8 bit mode. In practice, there isn't a noticeable difference in performance between the two modes. In this tutorial, I'll connect the LCD in 4 bit mode.

BONUS: I made a quick start guide for this tutorial that you can [download](#) and go back to later if you can't set this up right now. It covers all of the steps, diagrams, and code you need to get started.

CONNECTING THE LCD TO THE ARDUINO

Here's a diagram of the pins on the LCD I'm using. The connections from each pin to the Arduino will be the same, but your pins might be arranged differently on the LCD. Be sure to check the datasheet or look for labels on your particular LCD:

PCBWay HIGH-QUALITY PCB


ONLY \$5 FOR 10 PIECES

- Rogers, HDI, aluminum and rigid-flex PCB are available now
- Production time 24 hours



COMPLEX PCB CHOOSE

PCBONLINE



www.pcbonline.com

JLCPCB

Only \$2 for any color

5pcs, 10x10cm

ORDER NOW

JLCPCB.COM

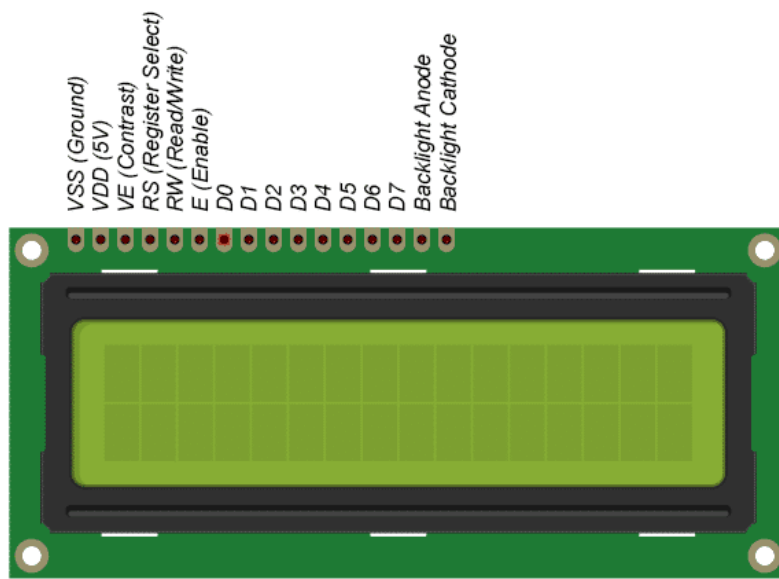


ourPCB

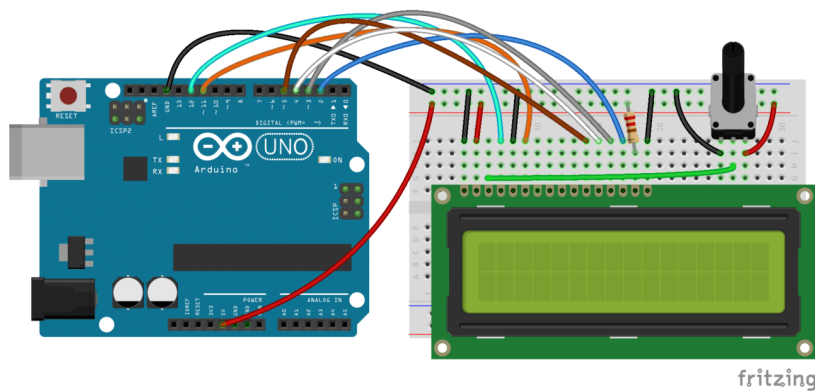
One-Stop Service For PCB&PCBA

2-32 Layers PCB Manufacturing
Turn-Key PCB Assembly
Quick Turn Prototype



Also, you might need to solder a [16 pin header](#) to your LCD before connecting it to a breadboard. Follow the diagram below to wire the LCD to your Arduino:



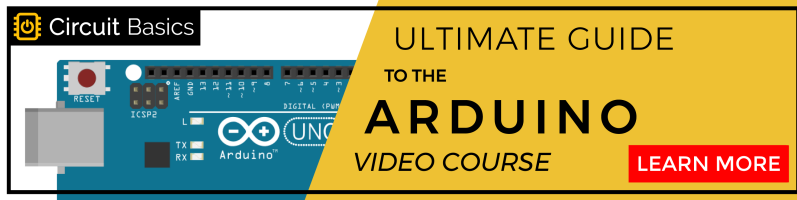
The resistor in the diagram above sets the backlight brightness. A typical value is 220 Ohms, but other values will work too. Smaller resistors will make the backlight brighter.

The potentiometer is used to adjust the screen contrast. I typically use a [10K Ohm potentiometer](#), but other values will also work.

Here's the datasheet for the 16x2 LCD with all of the technical information about the display:



[16x2 LCD Datasheet](#)



PROGRAMMING THE ARDUINO

All of the code below uses the *LiquidCrystal* library that comes pre-installed with the Arduino IDE. A library is a set of functions that can be easily added to a program in an abbreviated format.

In order to use a library, it needs be *included* in the program. Line 1 in the code below does this with the command `#include <LiquidCrystal.h>`. When you include a library in a program, all of the code in the library gets uploaded to the Arduino along with the code for your program.

Now we're ready to get into the programming! I'll go over more interesting things you can do in a moment, but for now lets just run a simple test program. This program will print "hello, world!" to the screen. Enter this code into the Arduino IDE and upload it to the board:

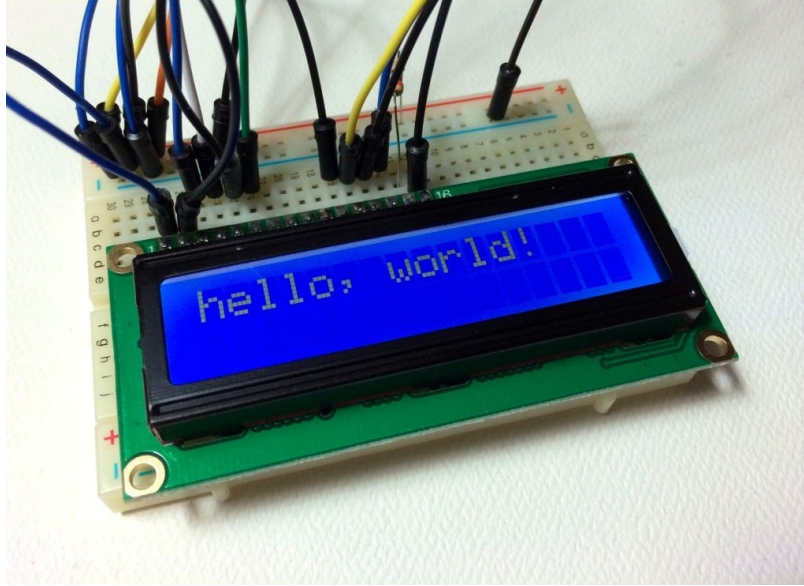
```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
}
```

Your LCD screen should look like this:



LCD DISPLAY OPTIONS

There are 19 different functions in the LiquidCrystal library available for us to use. These functions do things like change the position of the text, move text across the screen, or make the display turn on or off. What follows is a short description of each function, and how to use it in a program.

LiquidCrystal()

The `LiquidCrystal()` function sets the pins the Arduino uses to connect to the LCD. You can use any of the Arduino's digital pins to control the LCD. Just put the Arduino pin numbers inside the parentheses in this order:

```
LiquidCrystal(RS, E, D4, D5, D6, D7)
```

RS, E, D4, D5, D6, D7 are the LCD pins.

For example, say you want LCD pin D7 to connect to Arduino pin 12. Just put "12" in place of D7 in the function like this:

```
LiquidCrystal(RS, E, D4, D5, D6, 12)
```

This function needs to be placed before the `void setup()` section of the program.



lcd.begin()

This function sets the dimensions of the LCD. It needs to be placed before any other LiquidCrystal function in the `void setup()` section of the program. The number of rows and columns are specified as `lcd.begin(columns, rows)`. For a 16×2 LCD, you would use `lcd.begin(16, 2)`, and for a 20×4 LCD you would use `lcd.begin(20, 4)`.

lcd.clear()

This function clears any text or data already displayed on the LCD. If you use `lcd.clear()` with `lcd.print()` and the `delay()` function in the `void loop()` section, you can make a simple blinking text program:

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.print("hello, world!");
  delay(500);
  lcd.clear();
  delay(500);
}
```

lcd.home()

This function places the cursor in the upper left hand corner of the screen, and prints any subsequent text from that position. For example, this code replaces the first three letters of “hello world!” with X’s:

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
```




```
    lcd.print("hello, world!");  
}  
  
void loop() {  
    lcd.home();  
    lcd.print("XXX");  
}
```



lcd.setCursor()

Similar, but more useful than `lcd.home()` is `lcd.setCursor()`. This function places the cursor (and any printed text) at any position on the screen. It can be used in the `void setup()` or `void loop()` section of your program.

The cursor position is defined with `lcd.setCursor(column, row)`. The column and row coordinates start from zero (0-15 and 0-1 respectively). For example, using `lcd.setCursor(2, 1)` in the `void setup()` section of the “hello, world!” program above prints “hello, world!” to the lower line and shifts it to the right two spaces:

```
#include <LiquidCrystal.h>  
  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
  
void setup() {  
    lcd.begin(16, 2);  
    lcd.setCursor(2, 1);  
    lcd.print("hello, world!");  
}
```

```
void loop() {  
}
```

lcd.write()

You can use this function to write different types of data to the LCD, for example the reading from a temperature sensor, or the coordinates from a GPS module. You can also use it to print custom characters that you create yourself (more on this below). Use `lcd.write()` in the `void setup()` or `void loop()` section of your program.

lcd.print()

This function is used to print text to the LCD. It can be used in the `void setup()` section or the `void loop()` section of the program.

To print letters and words, place quotation marks (" ") around the text. For example, to print *hello, world!*, use `lcd.print("hello, world!")`.

To print numbers, no quotation marks are necessary. For example, to print 123456789, use `lcd.print(123456789)`.

`lcd.print()` can print numbers in decimal, binary, hexadecimal, and octal bases. For example:



- `lcd.print(100, DEC)` prints "100";
- `lcd.print(100, BIN)` prints "1100100"
- `lcd.print(100, HEX)` prints "64"
- `lcd.print(100, OCT)` prints "144"

lcd.cursor()

This function creates a visible cursor. The cursor is a horizontal line placed below the next character to be printed to the LCD.

The function `lcd.noCursor()` turns the cursor off. `lcd.cursor()` and `lcd.noCursor()` can be used together in the `void loop()` section to make a blinking cursor similar to what you see in many text input fields:

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
  lcd.cursor();
  delay(500);
  lcd.noCursor();
  delay(500);
}
```

This places a blinking cursor after the exclamation point in "hello, world!"

Cursors can be placed anywhere on the screen with the `lcd.setCursor()` function. This code places a blinking cursor directly below the exclamation point in "hello, world!":

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
```



```
    lcd.print("hello, world!");
}

void loop() {
    lcd.setCursor(12, 1);
    lcd.cursor();
    delay(500);
    lcd.setCursor(12, 1);
    lcd.noCursor();
    delay(500);
}
```

lcd.blink()

This function creates a block style cursor that blinks on and off at approximately 500 milliseconds per cycle. Use it in the `void loop()` section. The function `lcd.noBlink()` disables the blinking block cursor.

lcd.display()

This function turns on any text or cursors that have been printed to the LCD screen. The function `lcd.noDisplay()` turns off any text or cursors printed to the LCD, without clearing it from the LCD's memory.

These two functions can be used together in the `void loop()` section to create a blinking text effect. This code will make the "hello, world!" text blink on and off:

```
#include <LiquidCrystal.h>

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

void setup() {
    lcd.begin(16, 2);
    lcd.print("hello, world!");
}

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



lcd.scrollDisplayLeft()

This function takes anything printed to the LCD and moves it to the left. It should be used in the `void loop()` section with a delay command following it. The function will move the text 40 spaces to the left before it loops back to the first character. This code moves the "hello, world!" text to the left, at a rate of one second per character:

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
  lcd.scrollDisplayLeft();
  delay(1000);
}
```

Text strings longer than 40 spaces will be printed to line 1 after the 40th position, while the start of the string will continue printing to line 0.

lcd.scrollDisplayRight()

This function behaves like `lcd.scrollDisplayLeft()`, but moves the text to the right.

lcd.autoscroll()

This function takes a string of text and scrolls it from right to left in increments of the character count of the string. For example, if you have a string of text that is 3 characters long, it will shift the text 3 spaces to the left with each step:

```
#include <LiquidCrystal.h>

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



```
void setup() {  
  lcd.begin(16, 2);  
}  
  
void loop() {  
  lcd.setCursor(0, 0);  
  lcd.autoscroll();  
  lcd.print("ABC");  
  delay(500);  
}
```

Like the `lcd.scrollDisplay()` functions, the text can be up to 40 characters in length before repeating. At first glance, this function seems less useful than the `lcd.scrollDisplay()` functions, but it can be very useful for creating animations with custom characters.

`lcd.noAutoscroll()`

`lcd.noAutoscroll()` turns the `lcd.autoscroll()` function off. Use this function before or after `lcd.autoscroll()` in the `void loop()` section to create sequences of scrolling text or animations.

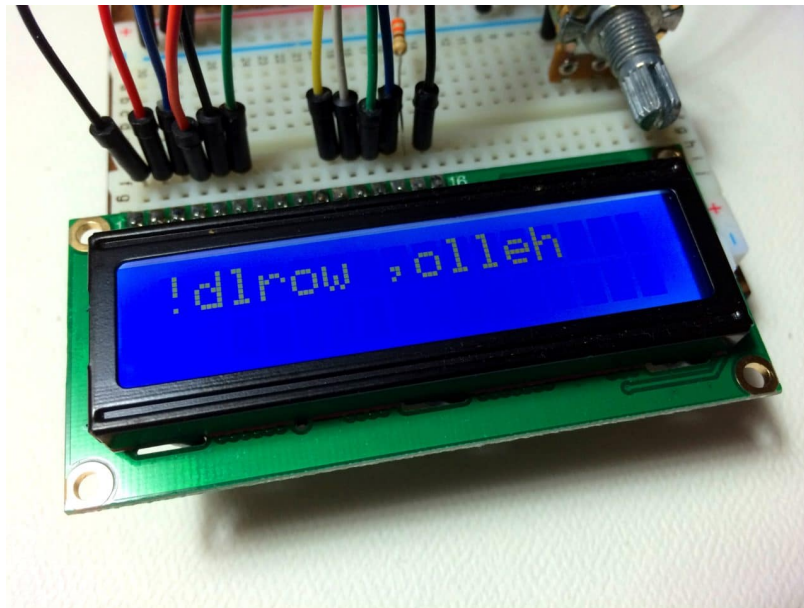
`lcd.rightToLeft()`

This function sets the direction that text is printed to the screen. The default mode is from left to right using the command `lcd.leftToRight()`, but you may find some cases where it's useful to output text in the reverse direction:

```
#include <LiquidCrystal.h>  
  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
  
void setup() {  
  lcd.begin(16, 2);  
  lcd.setCursor(12, 0);  
  lcd.rightToLeft();  
  lcd.print("hello, world!");  
}  
  
void loop() {  
}
```



This code prints the “hello, world!” text as “!dlrow ,olleh”, unless you specify the placement of the cursor with `lcd.setCursor()`, the text will print from the (0, 1) position and only the first character of the string will be visible.



`lcd.createChar()`

This command allows you to create your own custom characters. Each character of a 16x2 LCD has a 5 pixel width and an 8 pixel height. Up to 8 different custom characters can be defined in a single program. To design your own characters, you'll need to make a binary matrix of your custom character from an [LCD character generator](#) or map it yourself. This code creates a degree symbol (°):

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
byte customChar[8] = {
    0b00110,
    0b01001,
    0b01001,
    0b00110,
    0b00000,
    0b00000,
    0b00000,
    0b00000
};

void setup() {
    lcd.createChar(0, customChar);
    lcd.begin(16, 2);
    lcd.write((uint8_t)0);
}
```



```
}  
  
void loop() {  
}
```

There are a lot of cool things you can make happen with these 16×2 LCDs! Try combining some of these functions and see what happens.

Here's a video version of this tutorial so you can see what each function does on the LCD in real time:

How to Set Up and Program an LCD on the



If you found this article useful, subscribe via email to get notified when we publish of new posts! And as always, if you are having trouble with anything, just leave a comment and I'll try to help you out.

JLPCB - Only \$2 for PCB Prototype (Any Color)
Great Quality Approved by 600,000+ Customers, 10,000+ PCB Orders Per Day.

