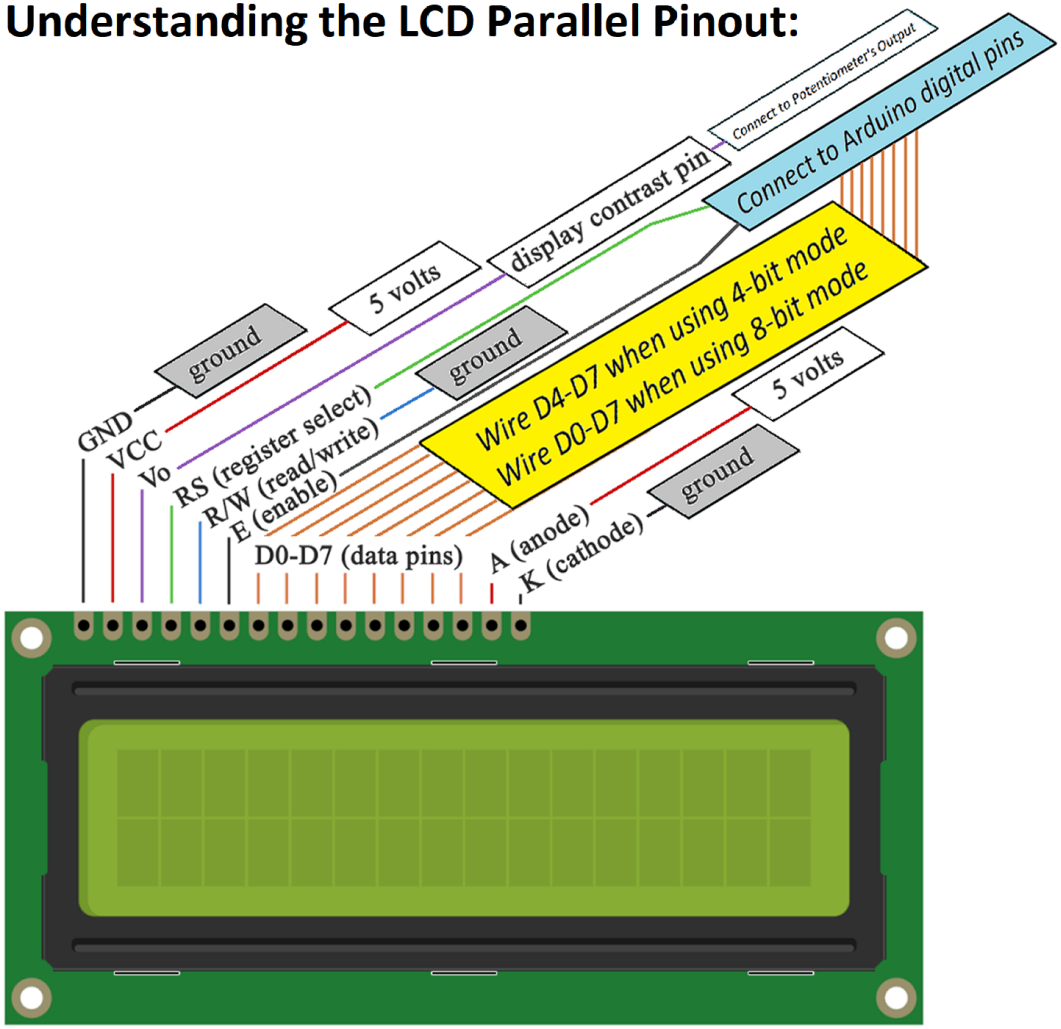
Arduino Guide: Setting Up A Character LCD with Parallel I/O

www.abra-electronics.com

What You Need:

1) 1x Arduino board (Uno or Nano)
2) 1x Character LCD
3) 1x Potentiometer (Variable Resistor)
4) M-M jumper Wires
5) Breadboard
6) Power Supply (7-12V)
   OR
   9V Battery

Understanding the LCD Parallel Pinout:



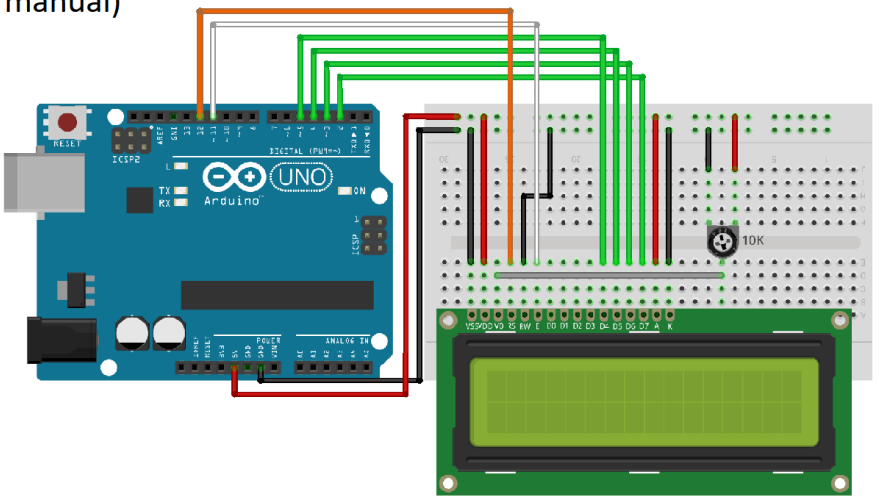
Wiring Diagram:

- Below is an example of how you would hook up a character LCD in 4-bit mode to an Arduino Uno, the process will be the same for other boards. Just make sure that you are using the digital pins.

- If you are using a transflective LCD (usually black on green or black on yellow), then using the backlight LED is optional.

- To get a better performance, use an external power supply with an output rated @ 7-12 Volts.

- Take note of the pins used as you will need to specify them in the code. (Check out the example on the back side of this manual)



Parallel I/O vs. Serial (I^2C, UART and SPI) Comparison:

1- When the number of peripherals in an embedded system increases, using serial might cause annoying latencies or delays in data delivery especially at low baud rates. Using parallel interface provides a fast response in 4-bit mode and even faster in 8-bit mode.

2- Serial signal frames get ridiculously large when the data for multiple devices (microcontrollers, LCDs, sensors, etc.) is passed on to an I^2C or SPI bus; therefore in the case of implementing a character LCD in your project, parallel I/O can be used to create a well organized and modular system by taking out the LCD operation out of the equation. Also, it is much easier to code for a parallel LCD.

3- Serial communication uses the asynchronous UART/USART or the synchronous SPI or I^2C bus which as a result uses less pins and decreases the amount of wiring required while parallel communication uses more pins and requires more wiring.

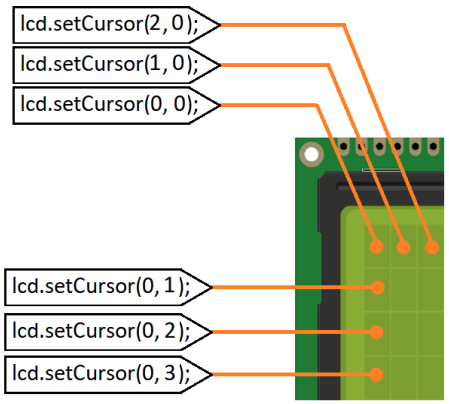
You can read more here: ***https://www.quantil.com/content-delivery-insights/content-acceleration/data-transmission/***

Arduino Sketch Example:

```
/*
  LiquidCrystal Library

  The LiquidCrystal library works with all LCD displays that are compatible
  with the Hitachi HD44780 driver. This also includes the SPLC780D,
  ST7066U and KS0066 driven LCDs. There are many of them out there,
  and you can usually tell them by the 16-pin interface.

  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * potentiometer's output to LCD VO pin (pin 3)
*/

// include the library code:
#include <LiquidCrystal.h>


// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
// rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);  // (colomns, rows)
  // other examples (8, 2), (16, 1), (20, 4), (40, 4)
  // set the cursor to column 0, line 1
  // (note: line 0 is the first row, since counting begins with 0):
  lcd.setCursor(0, 0);
  // Print a message to the LCD.
  lcd.print("LCD test OK! ");
}
void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

How does the "lcd.setCursor( )" function work in the LiquidCrystal Library?

Simply follow the instruction provided in the adjacent figure.

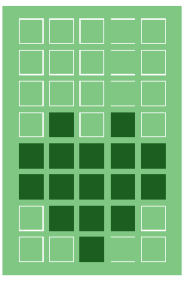


Want to create custom characters?

Integrate the following code snippet into your sketch:

```
#include <LiquidCrystal.h>

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

byte heart[8] = {
  B00000,
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
};

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

Example:
In order to create a heart shaped character, the following pixel combination is used:



```
byte customChar[] = {
  B00000,
  B00000,
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100
};
```

For more details about the LiquidCrystal library and its functions visit:

https://www.arduino.cc/en/Reference/LiquidCrystal