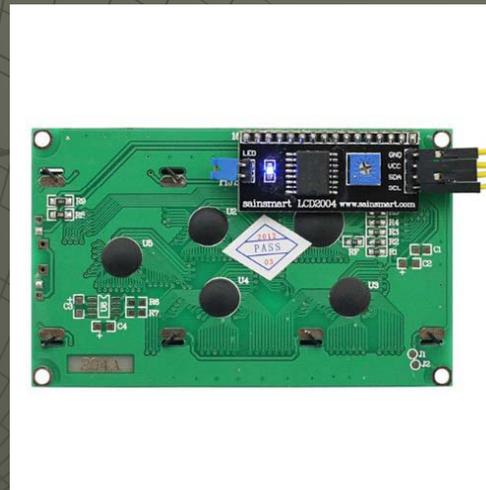


# Hardware Required

In this tutorial we'll become familiar with LCDs (Liquid Crystal Displays) and how to use them in your projects



-LCD Module: Four Line 20 characters/line I2C LCD

- Purchase: <http://www.sainsmart.com/sainsmart-iic-i2c-twi-serial-2004-20x4-lcd-module-shield-for-arduino-uno-mega-t3.html>
- Purchase: <http://www.amazon.com/SainSmart-Serial-Module-Shield-Arduino/dp/B0080DYTZO>

-Arduino Uno: From Arduino, Amazon, Sparkfun, MP3Cars, many more:

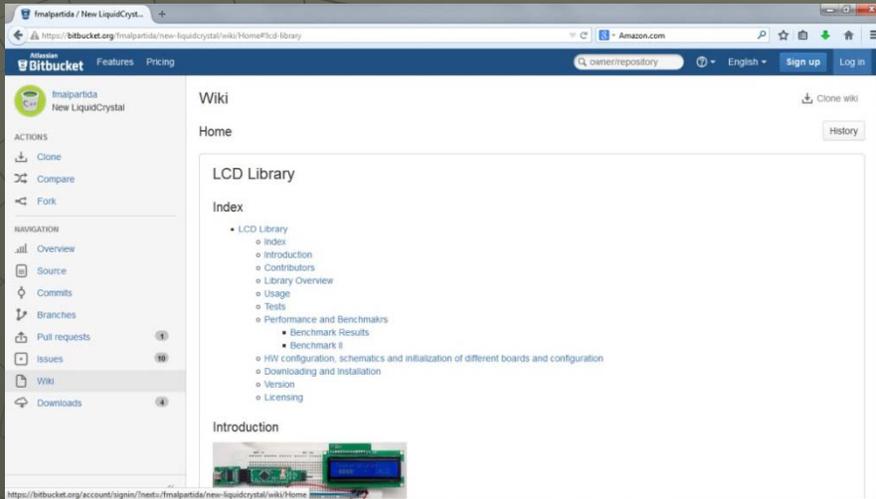
- Purchase: <http://www.amazon.com/Arduino-UNO-board-DIP-ATmega328P/dp/B006H06TVG>
- Schematic: [http://arduino.cc/en/uploads/Main/Arduino\\_Uno\\_Rev3-schematic.pdf](http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf)

- A PC or laptop

# AIAA OC Rocketry

## AIAA OC Section – NAR #718

# Software Required



Arduino Integrated Development Environment (IDE):

<http://arduino.cc/en/main/software#.Uy4WgU1OUpA>

I2C LCD Library – F Malpartida – download the latest .zip file (currently LiquidCrystal\_V1.2.1.zip)

Library: <https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads>

Info: <https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home#I2c-library>

You will need the “LCD” and the “LiquidCrystal\_I2C” libraries (in the “LiquidCrystal” folder)

Sketches: [ftp://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/LCD\\_20x4\\_I2C.zip](ftp://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/LCD_20x4_I2C.zip)

Four sample sketches show how to use the I2C LCD and the library

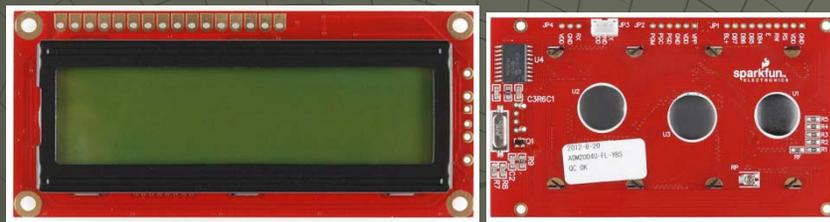
- LCD\_20x4\_I2C>HelloWorld: Displays fixed characters “Hello world” and more
- LCD\_20x4\_I2C>SerialDisplay: Displays characters sent to the Arduino via the serial port
- LCD\_20x4\_I2C>TextDirection: Shows how to display characters left to right and right to left
- LCD\_20x4\_I2C>AutoScroll: Shows how to scroll characters as they appear

Seeit serial communications program (or use your favorite program):

<ftp://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/seeit.zip>



# Liquid Crystal Displays (LCDs)



Many different LCDs are available in many different “flavors”:

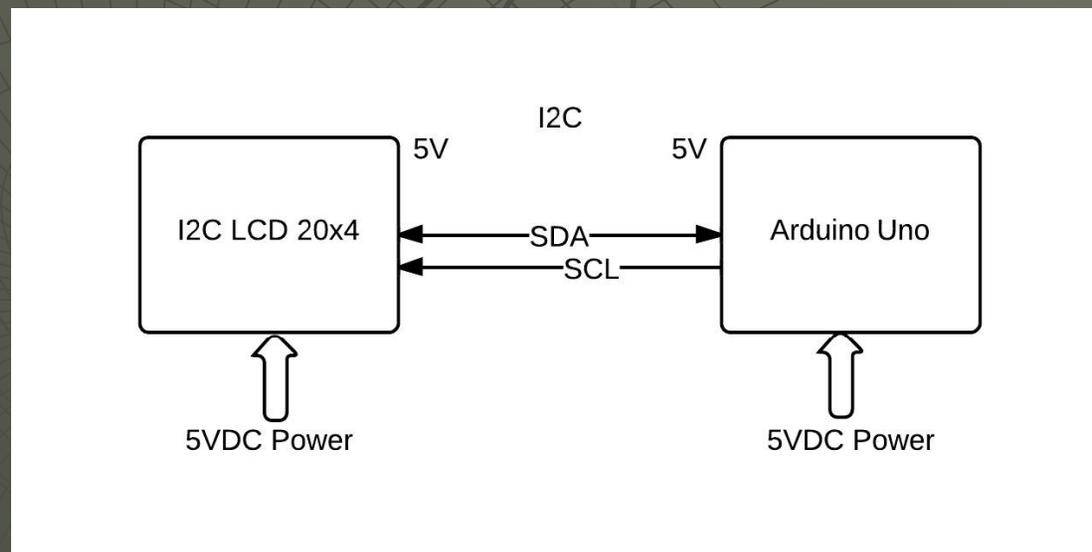
- Display line of text or graphics
- Several sizes (e.g. 16 characters x 2 lines, 20 characters x 4 lines)
- Monochrome colors (e.g. white on blue, black on green)
- Interface (parallel, serial, I2C)

This tutorial uses a text display with 4 lines of text, 20 characters per line and with an I2C interface to minimize the connecting wires



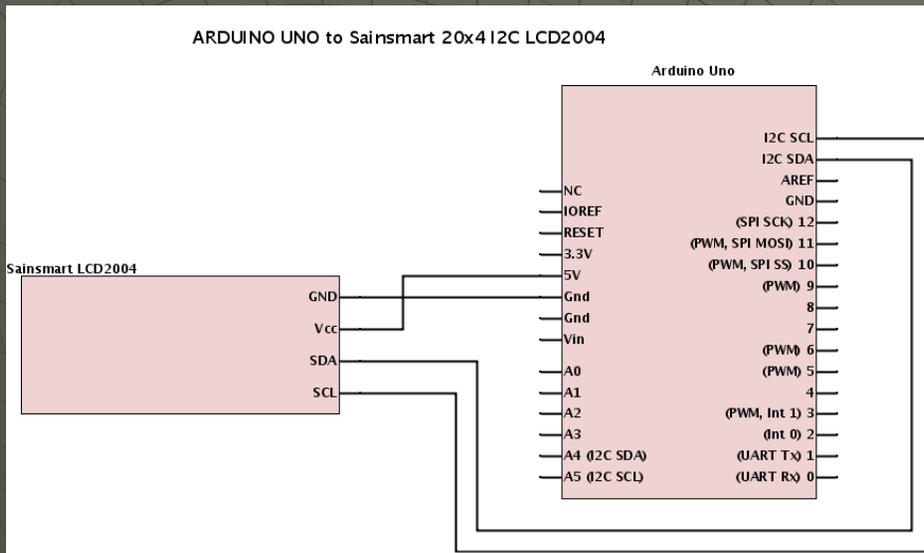
# Connecting the LCD to the Arduino Uno Block Diagram

LCDs with an I2C interface require fewer wires between the LCD and the Arduino Uno. Both the LCD and the Arduino Uno run at 5VDC so no level shifters are required





# Connecting the LCD to the Arduino Uno Schematic



The Sainsmart LCD2004 20x4 I2C module uses the HD44780 controller from Hitachi and a PCF8574 I2C 8 bit I/O expander

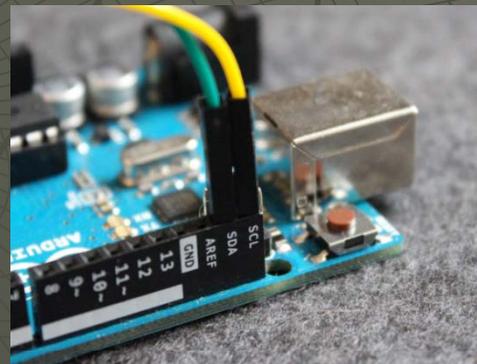
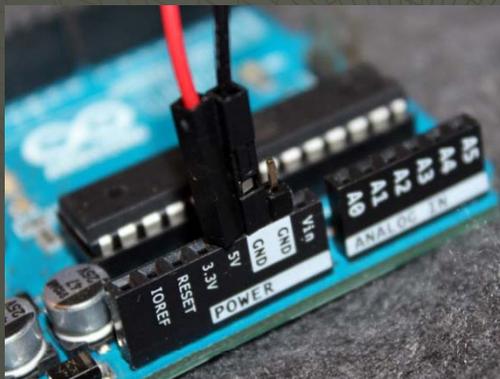
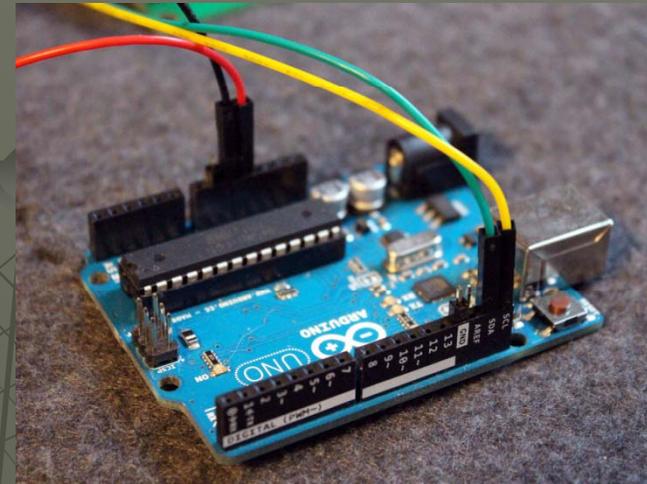
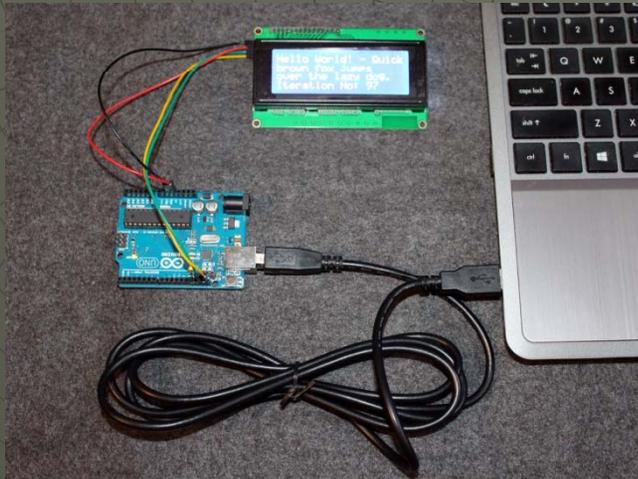
Datasheets:

Hitachi HD44780 controller <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

PCF 8574 I2C to parallel [http://www.nxp.com/documents/data\\_sheet/PCF8574.pdf](http://www.nxp.com/documents/data_sheet/PCF8574.pdf)

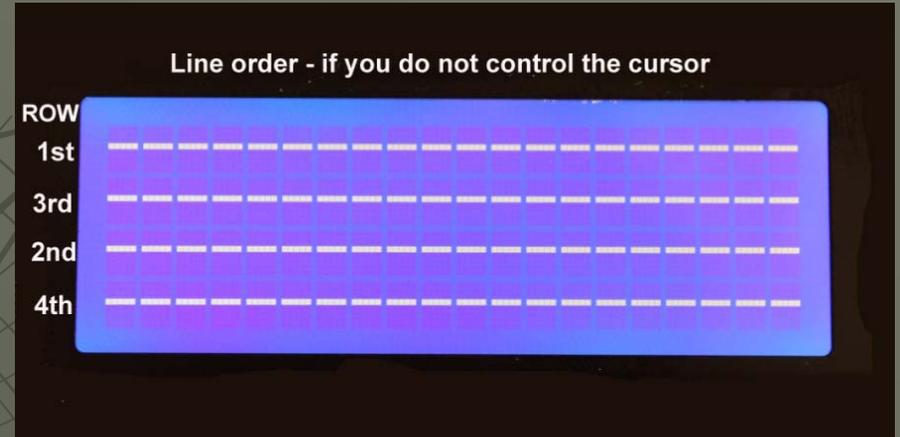
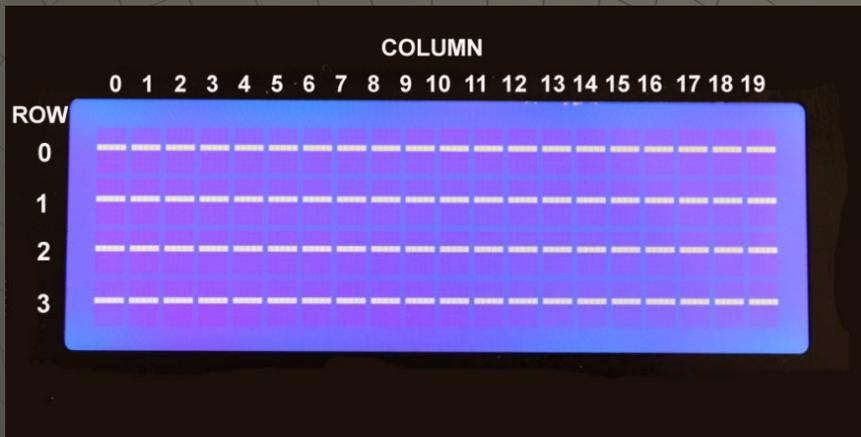


# Connecting the LCD to the Arduino Uno





# Lines and character positions



Line ordering and character positions on a multi line display with the Hitachi HD44780 controller are not straightforward. Lines are 0 – 3 when using the setCursor call. If you do not set the cursor, it will wrap from lines 1 to 3 to 2 to 4



# Lines and character positions

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	← Character position (dec.)
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	← Row0 DDRAM address (hex)
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	← Row1 DDRAM address (hex)
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	← Row2 DDRAM address (hex)
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	← Row3 DDRAM address (hex)

When directly addressing characters, or when using autoscroll or right to left, you may observe unanticipated behavior with character positions beyond the first line. The Hitachi controller is used for many different displays with varied characters per lines and number of lines, so the characters are not all in a sequential order as you would expect . The diagram above is from a tutorial on Rickey’s World on LCD interfacing:

<http://www.8051projects.net/lcd-interfacing/basics.php>



# Install Arduino IDE and Libraries

Working Directory Tree:  
(My) Documents  
  Arduino  
    libraries  
      LiquidCrystal  
      <more...>

Note: You can skip this step if you have already installed the IDE

## STEP 1: INSTALL THE ARDUINO IDE

- Download the IDE (currently named "arduino-1.0.5-r2-windows.exe") and click on it to install
- It will be installed in the Program Files directory
- Your sketches and libraries will go in the Arduino folder in (my) Documents
- Also allow the installation of the device driver software

## STEP 2: INSTALL THE LIBRARY

- Download and install the I2C LCD Library from F Malpartida (currently LiquidCrystal\_V1.2.1.zip)  
<https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads>
- Library install instructions are here:  
<http://arduino.cc/en/Guide/Libraries#.Uy4bYU1OUpA>
- To automatically install a downloaded .zip file library, start the Arduino IDE and click on: SKETCH->IMPORT LIBRARY->ADD LIBRARY then navigate to where the libraries were downloaded and click on the library (.zip file or folder) – check that the library has been added under "contributed" in the list under SKETCH->IMPORT LIBRARY->ADD LIBRARY
- To manually install a downloaded library, unzip it and move the folder containing all files into the "libraries" folder in the (My) Documents\Arduino\libraries, then restart the IDE



Download the LCD sketches from the AIAA OC Rocketry web sites:

Sketches: [http://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/LCD\\_20x4\\_I2C.zip](http://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/LCD_20x4_I2C.zip)

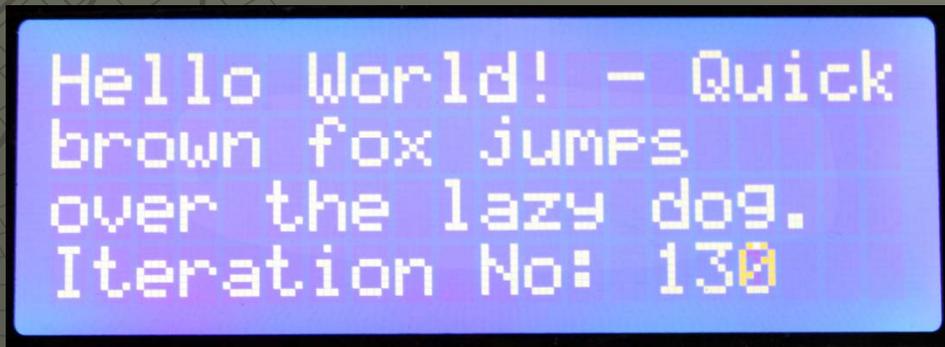
Unzip the folders with the .ino file into the Sketches folder under Arduino in (My)Documents. The first sketch we will use is the LCD\_20x4\_I2C>HelloWorld. This sketch will display fixed on “canned” text on the LCD, and increment a counter. It is a good beginning to make certain everything is connected properly and shows how to initialize the LCD and move the cursor

```
void setup()
{
  lcd.begin (20,4);

  // Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home ();           // go home

  lcd.print("Hello World! - Quick");
  lcd.setCursor ( 0, 1 ); // go to the 2nd line
  lcd.print("brown fox jumps");
  lcd.setCursor ( 0, 2 ); // go to the third line
  lcd.print("over the lazy dog.");
  lcd.setCursor ( 0, 3 ); // go to the fourth line
  lcd.print("Iteration No: ");
}

void loop()
{
  // Advances iteratino count every 1 second
  lcd.setCursor (14,3); // go col 14 of line 3
  lcd.print(n++,DEC);
  delay(1000);
}
```



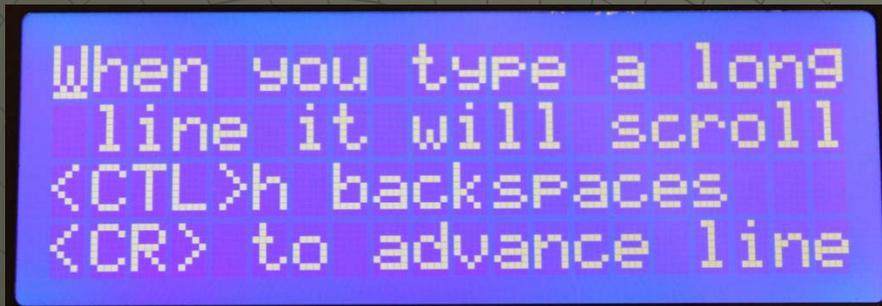
# AIAA OC Rocketry

## AIAA OC Section – NAR #718

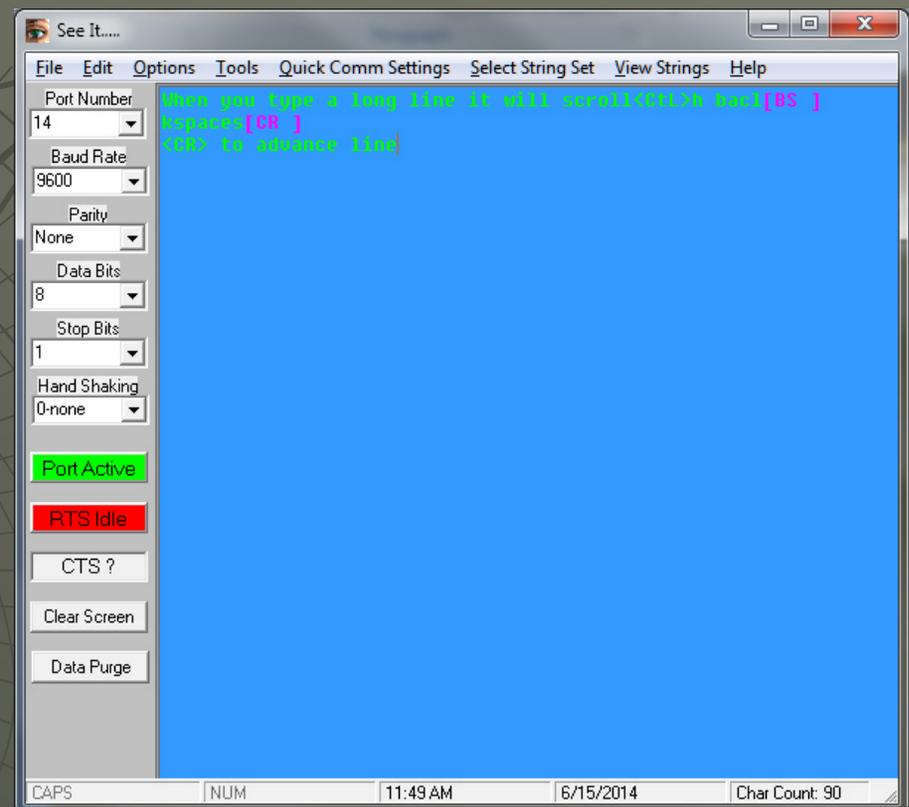
The next sketch we will use is the LCD\_20x4\_I2C\_Serial Display. This sketch will display the characters received on the serial port. You can install the SeeIt :

<http://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/seeit.zip>

serial communications program (set the proper COM port and BAUD rate to 9600). Or you can use the Serial Monitor Tool built in to the Arduino IDE, but it likes to send out a string at a time. This sketch shows how to handle the multiple line displays as well as how to receive a characters and then display them.



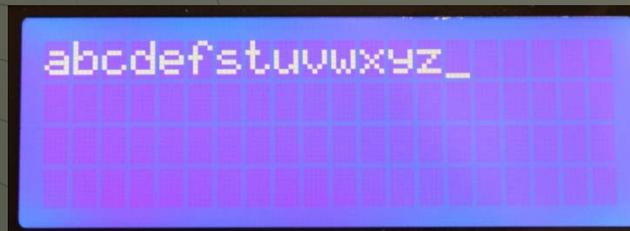
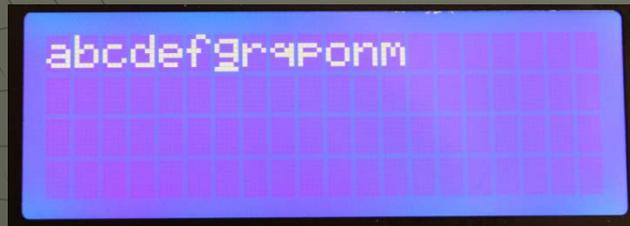
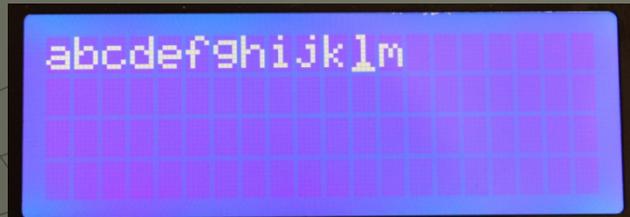
```
while (Serial.available() > 0) {
  newChar = Serial.read();
  if (newChar == 0x08){
    if (charCount == 0 && lineCount != 0){
      charCount = 20;
      lineCount = lineCount-1;
    }
  }
  if (charCount != 0 || lineCount != 0) {
    lcd.setCursor (--charCount, lineCount);
    lcd.write(0x20);
    lcd.setCursor (charCount, lineCount);
  }
}
if (newChar >= 0x20) {
  lcd.write(newChar);
  charCount++;
}
if ((charCount == 20) || (newChar == 0x0d)){
  charCount = 0;
  if (++lineCount > 3)
    lineCount = 0;
  lcd.setCursor (0,lineCount);
}
}
```





The next sketch we will use is the LCD\_20x4\_I2C\_TextDirection. This sketch will display a-z from left to right beginning with "a". When "m" is reached, it will reverse direction displaying characters from right to left. When "s" is reached it will reverse direction again. This is adapted from the examples in the Liquid Crystal library examples. This may or may not prove to be useful in your sketches, but now you know the library support it and how to use it

```
void loop() {  
  // reverse directions at 'm':  
  if (newChar == 'm') {  
    // go right for the next letter  
    lcd.rightToLeft();  
  }  
  // reverse again at 's':  
  if (newChar == 's') {  
    // go left for the next letter  
    lcd.leftToRight();  
  }  
  // reset at 'z':  
  if (newChar > 'z') {  
    // go to (0,0):  
    lcd.clear();  
    // start again at 0  
    newChar = 'a';  
  }  
  // print the character  
  lcd.write(newChar);  
  // wait a second:  
  delay(500);  
  // increment the letter:  
  newChar++;  
}
```



The final sketch we will use is the LCD\_20x4\_I2C\_TextDirection. This sketch will scroll numbers 0 – 9 in from the left side of the display, erase and scroll numbers 0 – 9 in from the right side of the display. You will need to carefully control the cursor position to use this feature past line #1. This is adapted from the examples in the Liquid Crystal library. Again, this may or may not prove to be useful in your sketches, but now you know the library support it and how to use it

```
void loop() {  
  // set the cursor to (0,0):  
  lcd.setCursor(0, 0);  
  lcd.cursor();  
  // print from 0 to 9:  
  for (int thisChar = 0; thisChar < 10; thisChar++) {  
    lcd.print(thisChar);  
    delay(500);  
  }  
  
  // set the cursor to (20,0):  
  lcd.clear();  
  lcd.setCursor(20,0);  
  lcd.noCursor();  
  // set the display to automatically scroll:  
  lcd.autoscroll();  
  // print from 0 to 9:  
  for (int thisChar = 0; thisChar < 10; thisChar++) {  
    lcd.print(thisChar);  
    delay(500);  
  }  
  // turn off automatic scrolling  
  lcd.noAutoscroll();  
  
  // clear screen for the next loop:  
  lcd.clear();  
}
```





# APPENDIX I

## Hardware and Libraries

In this appendix is background information on hardware and the libraries that control that hardware. You don't need to completely understand this information, but you do need to understand how to use the libraries



# LCD Controller Hardware

The LCD Library provides function calls to manipulate the LCD to display your data the way you want it displayed. Without the library, you would need to understand how to talk to the LCD controller ICs directly. With the library, you just need to make calls to functions that are already written for you that in turn talk to the LCD controller ICs. The HD44780 LCD controller has a parallel input to the microcontroller which uses many lines. Some LCD screens for the Arduino have an 8bit I/O Expander for I2C that converts the I2C interface to the microcontroller to the parallel interface required by the LCD. I2C uses fewer lines from your Arduino so they can be used for other things. You don't need to understand how to talk to these ICs, but here are the links to their data sheets if you are interested

HD44780 Data Sheet: <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

PCF8547 Data Sheet: [http://www.nxp.com/documents/data\\_sheet/PCF8574.pdf](http://www.nxp.com/documents/data_sheet/PCF8574.pdf)



# LCD Library I

The following routines make up the LCD library – you will be using these:

LiquidCrystal() – sets up the hardware interface

<http://arduino.cc/en/Reference/LiquidCrystalConstructor>

begin() – tells the library the number of rows and columns in your LCD

<http://arduino.cc/en/Reference/LiquidCrystalBegin>

clear() – clears the LCD screen and positions the cursor in the upper left

<http://arduino.cc/en/Reference/LiquidCrystalClear>

home() – positions the cursor in the upper left

<http://arduino.cc/en/Reference/LiquidCrystalHome>

setCursor() – positions the cursor to the specified row and column

<http://arduino.cc/en/Reference/LiquidCrystalSetCursor>

write() – writes a character to be displayed on the LCD

<http://arduino.cc/en/Reference/LiquidCrystalWrite>

print() – writes multiple characters to be displayed on the LCD

<http://arduino.cc/en/Reference/LiquidCrystalPrint>



# LCD Library II

Cursor() – turns display of the cursor ON

<http://arduino.cc/en/Reference/LiquidCrystalCursor>

noCursor() – turns display of the cursor OFF

<http://arduino.cc/en/Reference/LiquidCrystalNoCursor>

blink() – turns on the display of a blinking cursor

<http://arduino.cc/en/Reference/LiquidCrystalBlink>

noBlink() – turns off the blinking cursor

<http://arduino.cc/en/Reference/LiquidCrystalNoBlink>

display() – turns on the LCD after it was turned off – restores the text and cursor

<http://arduino.cc/en/Reference/LiquidCrystalDisplay>

noDisplay() – turns off the LCD without losing the text and cursor

<http://arduino.cc/en/Reference/LiquidCrystalNoDisplay>

scrollDisplayLeft() – scrolls the contents of the display one space to the left

<http://arduino.cc/en/Reference/LiquidCrystalScrollDisplayLeft>

scrollDisplayRight() – scrolls the contents of the display one space to the right

<http://arduino.cc/en/Reference/LiquidCrystalScrollDisplayRight>



# LCD Library III

Autoscroll() – automatically scrolls the characters on the display – each new character is in the same location: <http://arduino.cc/en/Reference/LiquidCrystalAutoscroll>

leftToRight() – sets the direction for text to be written to the LCD – left to right is the default <http://arduino.cc/en/Reference/LiquidCrystalLeftToRight>

rightToLeft() – sets the direction for text to be written to the LCD as right to left <http://arduino.cc/en/Reference/LiquidCrystalRightToLeft>

createChar() – allows you to enter up to eight 5x8 pixel custom characters <http://arduino.cc/en/Reference/LiquidCrystalCreateChar>

To use these functions precede them with "lcd". You declared "lcd" as an instance of this library when you added the line:

```
"LiquidCrystal_I2C          lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin)"
```

in your sketch. For example, to set up a four line by 20 character per line display, you would include "lcd.begin(20,4)" at the beginning of your sketch. The best way to become familiar with these functions is to look at the examples and try using them yourself. The complete current listing of all function calls and examples can be found here: [http://arduino.cc/en/Reference/LiquidCrystal?from=Tutorial\\_LCDLibrary](http://arduino.cc/en/Reference/LiquidCrystal?from=Tutorial_LCDLibrary)

Object oriented programming has the concept of classes that contains functions as members. An Object is an instantiation (to create the code that runs in your program) of a class. You access members of an object by specifying the object, then ".", then the member. In the above example, "lcd" is the object and "begin()" is the member, so to access begin() you need to put lcd.begin() in your code.