Using a HD44780 Compatible LCD module with an Arduino microcontroller and Programming Environment

Keith Jackson
A37673560

ECE480 Design Team 1
Fall 2010

Abstract

This application note is an introduction to interfacing a Hitachi HD44780 (or compatible) based LCD module with an Arduino development board. Included in this application note is an introduction to LCD and Arduino technologies, a description of physical connections, and a description of the code necessary to display characters on the LCD module. In addition, example code is included to show a complete solution.

Introduction

When visual output is required in a design, there are several different options- color LCDs, monochrome LCDs, and individual LEDs. All of these options have unique advantages over the others. LEDs are the simplest to implement, but can convey limited information. Color LCDs can show a great deal of information, but are the most difficult to implement in addition to being the most expensive. A good compromise between a color LCD and LEDs is a monochrome LCD. The most common controller chip used on monochrome LCD modules is the Hitachi HD44780. The HD44780 is a chip that receives information from a microcontroller, and then uses that information to make characters on the LCD by turning individual pixels on and off.

Choosing an LCD and Arduino board

Choosing hardware carefully will make the job of interfacing the LCD with the Arduino much simpler. There are considerations to make both on the LCD side, and the Arduino side.

Typical monochrome LCD modules based on the HD44780 are in one of several common configurations. Generally, such LCD modules feature 1, 2 or 4 lines, and 16, 20, or 40 characters per line. For simplicity, this application note covers the most common size: 16 characters, 2 lines (16x2).

Arduino development boards come in a variety of different flavors. They greatly range in size and capability. In general, the larger the unit, the more capability the unit has. Smaller units do not have as many I/O ports and have less memory and program storage space. Each LCD will require a minimum of 6 digital I/O pins, which is nearly half of the digital I/O pins on some of the smaller boards. For this reason, this application note covers the Arduino Mega, the largest in the Arduino lineup. The Arduino Mega features a 16 MHz processor clock, 128KB of flash memory, 8KB SRAM, and 54 digital I/O pins.

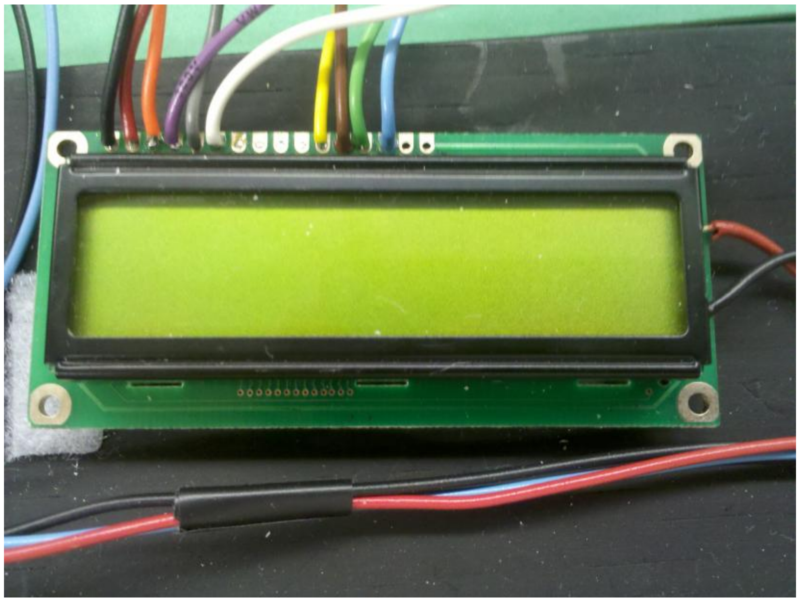


Figure 1: A 16x2 LCD Module

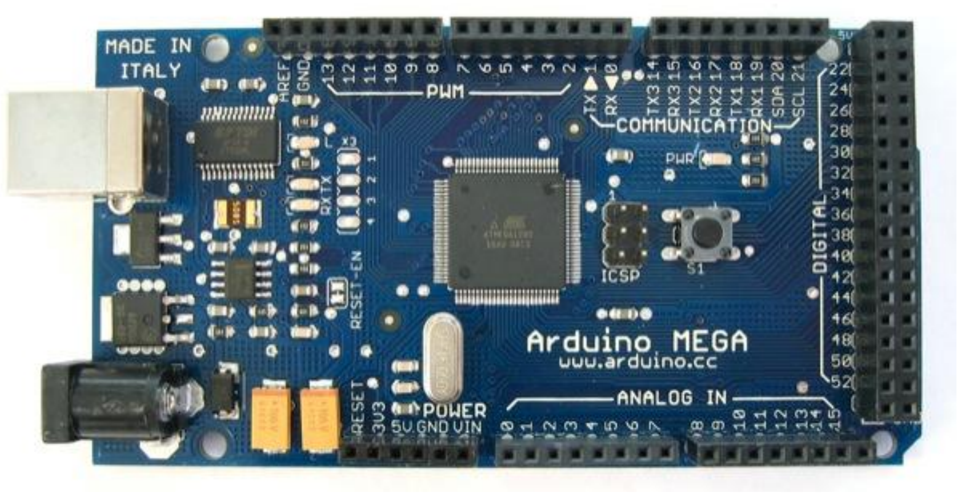


Figure 2: An Arduino Mega Development Board

Connecting the LCD to the Arduino

Once the LCD and Arduino have been picked, the next step is to properly wire them together. There are many ways to do so. There are two decisions to make: which digital I/O pins to use on the Arduino, and whether 8 bit addressing is necessary. When connecting an LCD to an Arduino, any of the Arduino's digital pins can be used, as long as the programming accounts for each used pin, as will be covered later on. There are two addressing modes for HD44780 controllers: 4 bit and 8 bit. In an 8 bit scheme, there will be a total of 10 I/O pins used, whereas 4 bit addressing requires only 6. Therefore, unless LCD refresh time is of the utmost importance, 8 bit addressing is usually unnecessary. In the diagram below in Figure 3, the correct connections are shown for the 4 bit addressing mode. In an 8 bit scheme, four wires would be added to the DB0-DB3 pins, which would connect to four more Arduino pins. R7 should be a 10K Ohm resistor. RP1 is a turnpot that is used to set the contrast of the display. Pins 15 and 16 on the diagram are LCD backlight power connections, which may or may not be present on an LCD.

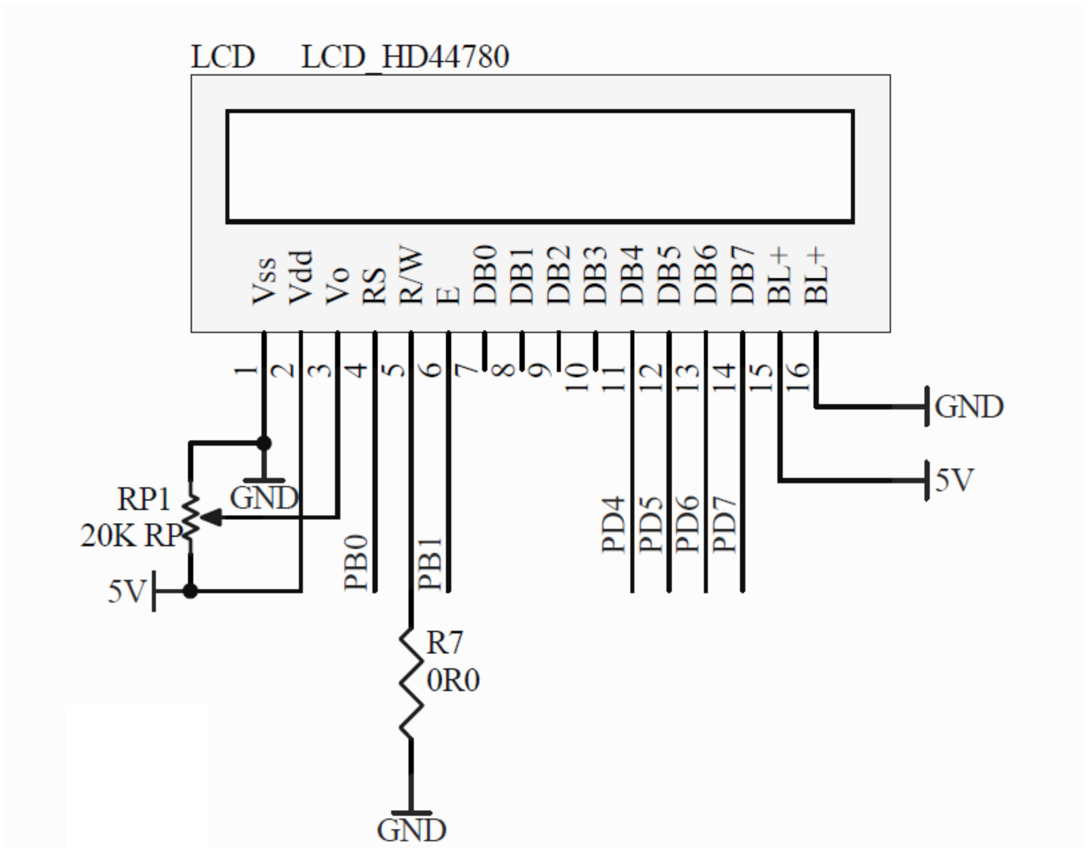
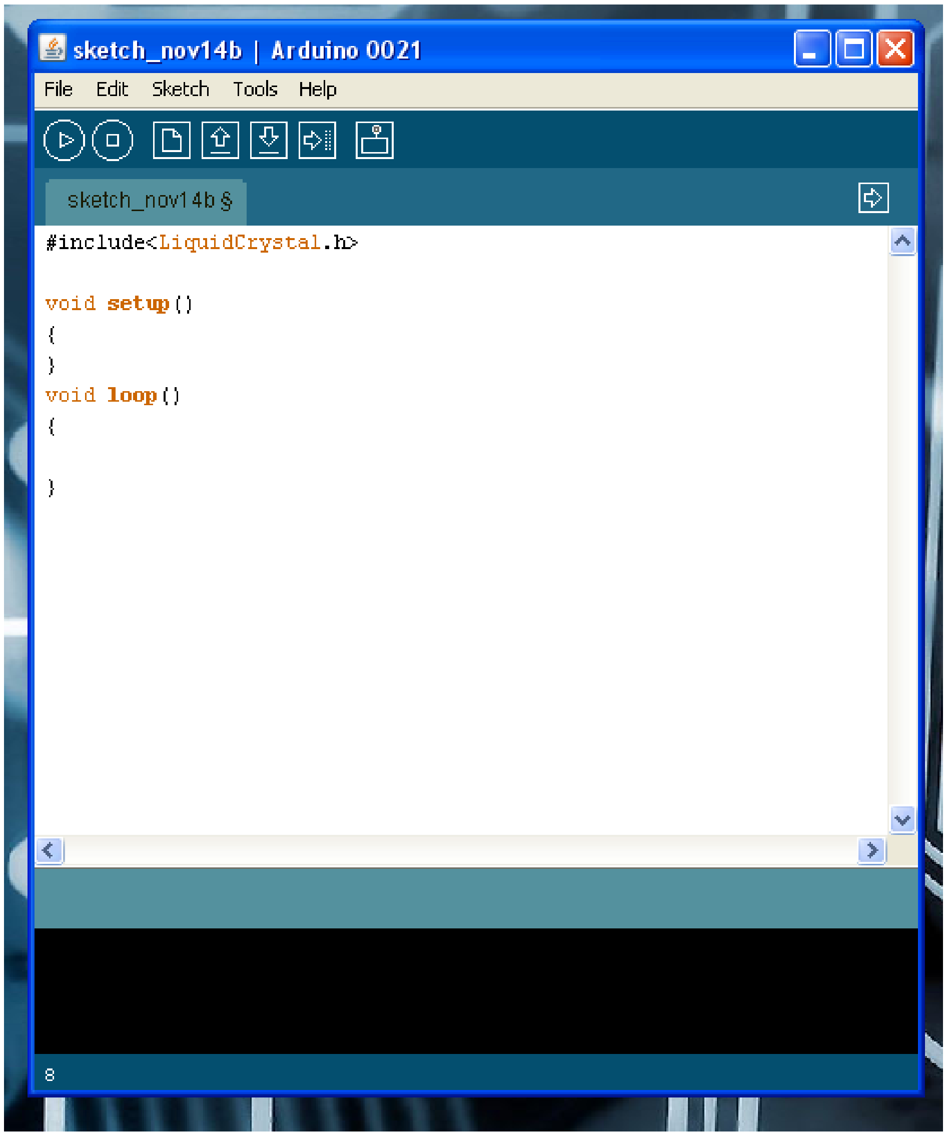


Figure 3: Wiring Diagram for the Arduino / LCD Interface

Application Code

To begin coding the actual program to control the LCD output, first download the Arduino software package from www.arduno.cc. This software package is able to compile and upload written code to the Arduino board. Starting a new project, we can begin to code the LCD output. Code for the Arduino is a derivative of the ubiquitous C programming language, and thus has all the standard C libraries and many others as well. The Arduino software includes a library (LiquidCrystal.h) that can interface with HD44780 compatible LCDs; this keeps LCD control at a high level, avoiding any involvement with machine code or assembly language code.



```
#include<LiquidCrystal.h>

void setup()
{
}
void loop()
{

}
```

Figure 4: The Arduino Programming Environment

The functions used in this application note are:

.begin(Rows, Characters);

.print("MESSAGE");

.setCursor(Column, Row);

.clear();

These functions are called by appending the function after the name of the LCD module. To initialize and set the name of the LCD module, the user first puts the line:

LiquidCrystal LCD(PB0, PB1, PD4, PD5, PD6, PD7);

Where PB0, PD4, etc. are the numbers corresponding to the pins on the Arduino (as shown in Fig. 3), and LCD is the name of the display.

The next step is to assign a size to the LCD by calling the *.begin()* command. If the LCD is a 16x2 unit, the function call would be made as follows:

LCD.begin(16, 2);

Once this is complete, the LCD is now ready to output character data.

The position of the cursor can be set by calling the *.set()* function. To set the position, the row and column will need to be put in as follows:

To set the cursor to the first character in the first row, the call would be:

LCD.setCursor(0, 0);

As another example, to set the cursor to the last character in the second row of a 16x2 LCD, the call would be:

LCD.setCursor(15, 1);

By using this command, the user can write a character to any point on the screen at any time.

To output to the screen, the *.print()* function will need to be used. It can output up to 16 characters at once. These characters will remain until they are overwritten, or cleared from the screen by the *.clear()* command as is shown below.

LCD.clear();

There are other advanced commands not included here, because they are not necessary for a basic understanding of the interfacing process. A full listing can be found at:

http://arduino.cc/en/Reference/LiquidCrystal

Code Example

```
/*****************************************************************************

#include <LiquidCrystal.h>
//Include the Library used to interface with the LCD

LiquidCrystal LCD(38, 40, 47, 49, 51, 53);
//Declare the pin numbers on the Arduino used for the LCD

unsigned long int loops = 0;
//Keep track of the amount of times the loop has run

void setup()
//Runs tasks that only need to be run at startup, not each loop
{
  LCD.begin(16, 2);
  //Sets the size of the LCD; in Columns, then Rows
}

void loop()
//Runs continuously, running until power is lost or the board is reset
{
  loops++;
  //Increase the loop counter

  LCD.setCursor(0, 0);
  //Set cursor to first line, first character

  LCD.print("Hello World!");
  //Print Hello World to the first line

  LCD.setCursor(0, 1);
  //Set cursor to the second line, first character

  LCD.print(loops);
  //Print out the value of loops

  long int temp = millis();
  //Set temp to millis; millis increase with time

  while ((temp + 1000) > millis())
  //Loop until one second has passed
  {
  }

  LCD.clear();
  //Clear the LCD screen

  temp = millis();
  //Set temp to millis; millis increase with time

  while ((temp + 1000) > millis())
  //Loop until one second has passed
  {
  }
}
*****************************************************************************/
```

The example code will display Hello World and the loop counter on the screen for one second, then clear the screen for one second. The loop will run continuously. While running, the screen will look as shown in Figure 5.

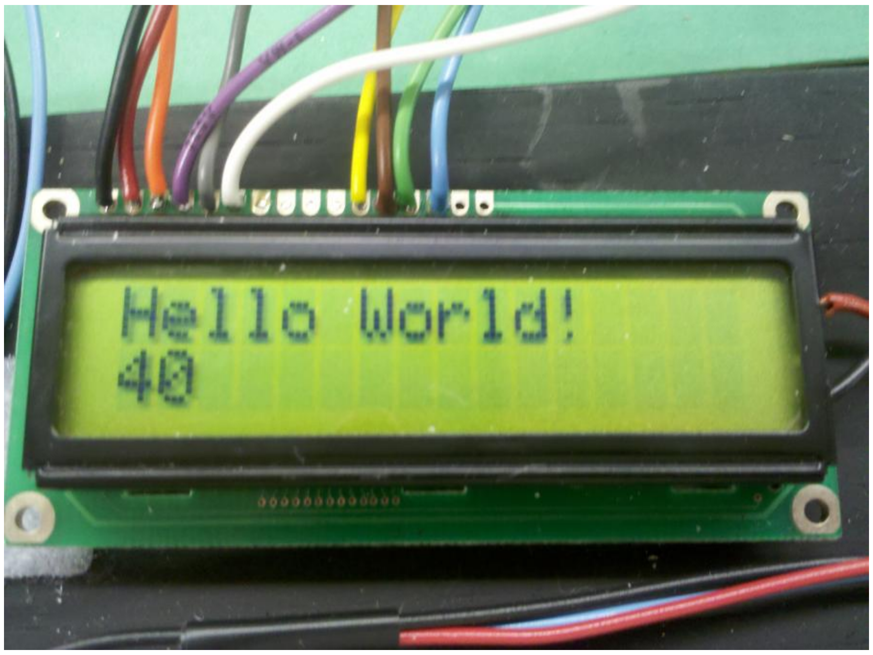


Figure 5: Completed Program

Conclusion

Both the Arduino development board and LCDs based on the Hitachi HD44780 controller chip have numerous possibilities for products. They are low-cost and highly capable. Using the Arduino programming interface, this can be undertaken by many, and is useful for LCD display outputs in almost any project or product.

References

- http://arduino.cc/en/
- http://arduino.cc/en/Reference/LiquidCrystal
- http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1206025987
- http://www.nuelectronics.com/download/projects/LCDshield_v1_1.pdf
- http://arduino.cc/en/Main/ArduinoBoardMega