Arduino LCD Tutorial

LCDs are a popular option for creating user interfaces for embedded systems. This is because LCDs are compact, relatively cheap, and can be driven effectively by a microcontroller with limited computational power. In the BME 354L final project, you will make use of an LCD to enhance the user interface of your device and gain experience interfacing with a ubiquitous technology.

The LCD that you are working with is called a shield for the Arduino. The term, shield, refers to a circuit board that plugs into the standard Arduino Board to provide some additional function, such as user display or wireless communication. Many shields are available for the Arduino Uno platform which can simplify and accelerate project development.

The LCD provided on this shield is a 16 X 2 character display which consumes 6 digital pins to operate and one analog pin to read button presses. These six digital pins allow the Arduino platform to communicate with the chip that controls the LCD. The nice thing about the way these shields are designed is that access is provided to all of the remaining pins on the board (It is a good idea to take an inventory of the pins you have available to you).

Arduino Libraries and References

Arduino provides a large set of well documented code that will help you quickly get up and running with your LCD. In fact, you can find functions that others have written which will do just about any basic task that you could imagine with the LCD. As with other programming languages, the Arduino Wiring language provides the ability to draw on these prewritten functions, which are packaged as libraries. A library is a collection of functions which you can draw from as you write your code. A link to the Arduino LiquidCrystal library is provided below. Keep in mind that there are other libraries out there.

Arduino LiquidCrystal.h library:

`http://arduino.cc/en/Reference/LiquidCrystal?from=Tutorial.LCDLibrary`

How to Install a Library

You can install a library for use in your programs by downloading the associated library and placing it in the libraries folder in your Arduino Directory. If you have visited the libraries folder previously, you will note that there are a number of default libraries that are provided when you download the Arduino IDE. Lucky for you, the LiquidCrystal library which contains functions for controlling an LCD is provided by default. If you wanted to install additional libraries to use in your programs, however, you need only to drop the library files in the libraries folder and then put an include statement in your program like this:

To use a library in your program you must declare it:

```
#include <NameOfArbitraryLib.h>
```

Exercise 1

The first part of this lab will expose you to the LiquidCrystal.h library. Lets make use of the built in tutorial sketches that are packaged with the Arduino IDE. The first thing that we will do is to test that your LCD is functional. In keeping with tradition, we will test the LCD with a Hello World program. Locate the Scroll example in the LiquidCrystal Library folder and open in the Arduino IDE. This example will show how to animate Hello World so that it scrolls across the screen in a ticker tape fashion.

We need to make a few changes to this program to configure it for our LCD shield from DFRobot. Locate the line in the Scroll.ino program which initializes the LCD. We will be using pins 4,5,6, and 7 for data, rather than 5,4,3 and 2. The order of these is significant, since the synax of the lcd command lists the pins on the Arduino that communicate with data channels D4, D5, D6, and D7 on the LCD in that order. Additionally, we must set RS to pin 8 and Enable to pin 9 for our shield. Check the LiquidCrystal library page to confirm that you have specified the pins correctly in the lcd command.

Objective 1: Modify the scroll.ino program to automatically accelerate or decelerate the scrolling as it pushes text off the screen. Demonstrate this function to your TA.

Lets add some more interesting elements to the animation. The LiquidCrystal library has a cool function called *createChar* which can be useful for adding custom characters to your display. The code for creating a smiley character and printing this character to the LCD is provided to you at the end of this document. Use this code and scroll a set of smiley faces across your LCD.

Objective 2: Create your own character using the *creatChar* function and demonstrate this new character in action! If you feel limited by using a 5X8 pixel character, you could consider how to make a special, multi digit custom character.

Exercise 2

Now, lets work on using the pushbuttons to accept user input. The pushbuttons on this shield (with the exception of the RESET button) are read using a single analog pin. Reading five different button presses with a single pin is done with a voltage divider. The button press, therefore, yields a different voltage on the analog pin 0 depending on which one is pressed. In the exercise below, we will explore this feature and use it to read user input.

Lets begin by figuring out the voltage divider that we are working with for reading button presses. This voltage divider is hardwired to pin A0 because the shield is plugged directly into the Arduino board.

Objective 3: Write a simple program that will read the analog pin A0 and print the voltage to the LCD screen. We will use these thresholds in the next step to create a program that accepts user input. Read the Arduino reference for the following commands if you are not yet familiar with them: *analogRead*,*lcd.print*, *lcd.setCursor*. Remember that you are now writing your own program rather than modifying a previous piece of code. In any Arduino program, including this one, you must write a setup function and a loop function. These tell the Arduino what it should be doing when it first turns on, and then what is should do for the rest of time.

The next thing we will do is to utilize the thresholds you have calculated based on your results of Objective 3. The template for your next program is shown in Appendix 1. This template includes an interesting and important feature of the Arduino language called the Switch Case. A switch case is a programming structure that allows you to handle different cases of the value of a variable. You could easily imagine how to implement this using a more basic programming element such as if statements and for loops, but what you are doing here is essentially equivalent. The compiler decides how to implement your Switch case optimally, rather than letting you do all of the dirty work. An efficient way to read the button press is to use the Switch case. Observe the program listed in this handout carefully.

Objective 4: Write a program that will utilize the pushbuttons and LCD to act as a stopwatch. Your program should accept user inputs to set the desired length of time in Min:Sec format. Next, your program should count down to zero and blink the LCD once the time has reached zero. As a final step, improve your stopwatch by allowing the user to start and stop the time.

```
\subsection*{Appendix 1 - Tutorial Code}

#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
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() {}
```

```
//Sample using LiquidCrystal library
#include <LiquidCrystal.h>

/*********************************************************

This program will test the LCD panel and the buttons
Mark Bramwell, July 2010

*********************************************************/

// select the pins used on the LCD panel
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

// define some values used by the panel and buttons
int lcd_key     = 0;
int adc_key_in  = 0;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5

#define V1     500
#define V2     500
#define V3     500
#define V4     500
#define V5     500
#define VNONE  500
```

```
// read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);      // read the value from the sensor
 // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
 // we add approx 50 to those values and check to see if we are close
 if (adc_key_in > VNONE) return btnNONE; // We make this the 1st option for speed reasons s
 if (adc_key_in < V1)   return btnRIGHT;
 if (adc_key_in < V2)  return btnUP;
 if (adc_key_in < V3)  return btnDOWN;
 if (adc_key_in < V4)  return btnLEFT;
 if (adc_key_in < V5)  return btnSELECT;
 return btnNONE;  // when all others fail, return this...
}

void setup()
{
 lcd.begin(16, 2);              // start the library

}

void loop()
{
 lcd_key = read_LCD_buttons();  // read the buttons

 switch (lcd_key)               // depending on which button was pushed, we perform an acti
 {
   case btnRIGHT:
     {
     //Instructions for what to do on RIGHT button press
     break;
     }
   case btnLEFT:
     {
     //Instructions for what to do on LEFT button press
     break;
     }
   case btnUP:
     {
     //Instructions for what to do on UP button press
     break;
     }
   case btnDOWN:
     {
     //Instructions for what to do on DOWN button press
     break;
     }
   case btnSELECT:
     {
     //Instructions for what to do on SELECT button press
```

```
      break;
      }
      case btnNONE:
      {
      //Instructions for what to do if no button is pressed.
      break;
      }
  }

}
```