Code

LCD Code

```
/*
SparkFun Inventor's Kit
Example sketch 15
LIQUID CRYSTAL DISPLAY (LCD)
  A Liquid Crystal Display (LCD) is a sophisticated module
  that can be used to display text or numeric data. The display
  included in your SIK features two lines of 16 characters, and
  a backlight so it can be used at night.
  If you've been using the Serial Monitor to output data,
  you'll find that a LCD provides many of the same benefits
  without needing to drag a large computer around.
  This sketch will show you how to connect an LCD to your Arduino
  and display any data you wish.
Hardware connections:
  The LCD has a 16-pin male header attached to it along the top
  edge. Pin 1 is the pin closest to the corner of the LCD.
  Pin 16 is the pin closest to the center of the LCD.
  Plug the LCD into your breadboard.
  As usual, you will want to connect the + and – power rails
  on the side of the breadboard to 5V and GND on your Arduino.
  Plug your 10K potentiometer into three unused rows on your
  breadboard. Connect one side of the potentiometer to 5V,
  and the other side to GND (it doesn't matter which). When you
  run this sketch, you'll use the potentiometer to adjust the
  contrast of the LCD so you can see the display.
  Now connect the LCD pins. Remember that pin 1 on the LCD
  is the one closest to the corner. Start there and work your
  way up.
  1 to GND
  2 to 5V
  3 to the center pin on the potentiometer
  4 to Arduino digital pin 12
  5 to GND
  6 to Arduino digital pin 11
  7 (no connection)
  8 (no connection)
  9 (no connection)
  10 (no connection)
  11 to Arduino  digital pin 5
  12 to Arduino  digital pin 4
  13 to Arduino  digital pin 3
  14 to Arduino  digital pin 2
  15 to 5V
  16 to GND
  Once everything is connected, load this sketch into the
  Arduino, and adjust the potentiometer until the display
```

```
  is clear.
Library
  The LCD has a chip built into it that controls all the
  individual dots that make up the display, and obeys commands
  sent to it by the the Arduino. The chip knows the dot patterns
  that make up all the text characters, saving you a lot of work.
  To communicate with this chip, we'll use the LiquidCrystal
  library, which is one of the standard libraries that comes
  with the Arduino. This library does most of the hard work
  of interfacing to the LCD; all you need to pick a location
  on the display and send your data!
Tips
  The LCD comes with a protective film over the display that
  you can peel off (but be careful of the display surface as
  it scratches easily).
  The LCD has a backlight that will light up when you turn on
  your Arduino. If the backlight doesn't turn on, check your
  connections.
  As we said above, the potentiometer adjusts the contrast of
  the display. If you can't see anything when you run the sketch,
  turn the potentiometer's knob until the text is clear.
This sketch was written by SparkFun Electronics,
with lots of help from the Arduino community.
This code is completely free for any use.
Visit https://www.sparkfun.com/products/12060 for SIK information.
Visit http://www.arduino.cc to learn about the Arduino.
Version 1.0 2/2013 MDG
*/
// Load the LiquidCrystal library, which will give us
// commands to interface to the LCD:
#include <LiquidCrystal.h>
// Initialize the library with the pins we're using.
// (Note that you can use different pins if needed.)
// See http://arduino.cc/en/Reference/LiquidCrystal
// for more information:
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  // The LiquidCrystal library can be used with many different
  // LCD sizes. We're using one that's 2 lines of 16 characters,
  // so we'll inform the library of that:
  lcd.begin(16, 2);
  // Data sent to the display will stay there until it's
  // overwritten or power is removed. This can be a problem
  // when you upload a new sketch to the Arduino but old data
  // remains on the display. Let's clear the LCD using the
  // clear() command from the LiquidCrystal library:
  lcd.clear();
  // Now we'll display a message on the LCD!
  // Just as with the Arduino IDE, there's a cursor that
  // determines where the data you type will appear. By default,
  // this cursor is invisible, though you can make it visible
  // with other library commands if you wish.
  // When the display powers up, the invisible cursor starts
```

```
  // on the top row and first column.
  lcd.print("public breastfeeding");

  // Adjusting the contrast (IMPORTANT!)
  // When you run the sketch for the first time, there's a
  // very good chance you won't see anything on the LCD display.
  // This is because the contrast likely won't be set correctly.
  // Don't worry, it's easy to set, and once you set it you won't
  // need to change it again.
  // Run the sketch, then turn the potentiometer until you can
  // clearly see the "hello, world!" text. If you still can't
  // see anything, check all of your connections, and ensure that
  // the sketch was successfully uploaded to the Arduino.
}
void loop()
{
  // You can move the invisible cursor to any location on the
  // LCD before sending data. Counting starts from 0, so the top
  // line is line 0 and the bottom line is line 1. Columns range
  // from 0 on the left side, to 15 on the right.
  // In additon to the "hello, world!" printed above, let's
  // display a running count of the seconds since the Arduino
  // was last reset. Note that the data you send to the display
  // will stay there unless you erase it by overwriting it or
  // sending a lcd.clear() command.
  // Here we'll set the invisible cursor to the first column
  // (column 0) of the second line (line 1):
  lcd.setCursor(0,1);
  // Now we'll print the number of seconds (millis() / 1000)
  // since the Arduino last reset:
  lcd.print(millis()/1000);
  // TIP: Since the numeric data we're sending is always growing
  // in length, new values will always overwrite the previous ones.
  // However, if you want to display varying or decreasing numbers
  // like a countdown, you'll find that the display will leave
  // "orphan" characters when the new value is shorter than the
  // old one.
  // To prevent this, you'll need to erase the old number before
  // writing the new one. You can do this by overwriting the
  // last number with spaces. If you erase the old number and
  // immediately write the new one, the momentary erase won't
  // be noticeable. Here's a typical sequence of code:
  // lcd.setCursor(0,1);   // Set the cursor to the position
  // lcd.print("       "); // Erase the largest possible number
  // lcd.setCursor(0,1);   // Reset the cursor to the original position
  // lcd.print(millis()/1000); // Print our value
  // NEXT STEPS:
  // Now you know the basics of hooking up an LCD to the Arduino,
  // and sending text and numeric data to the display!
  // The LCD library has many commands for turning the
  // cursor on and off, scrolling the screen, etc. See:
  // http://arduino.cc/en/Reference/LiquidCrystal
  // for more information.
  // Arduino also comes with a number of built-in examples
```

```
  // showing off the features of the LiquidCrystal library.
  // These are locted in the file/examples/LiquidCrystal menu.
  // Have fun, and let us know what you create!
  // Your friends at SparkFun.
}
```

Blink Code

```
/*
SparkFun Inventor's Kit
Example sketch 01

BLINKING A LED

  Turn an LED on for one second, off for one second,
  and repeat forever.

Hardware connections:

  Most Arduinos already have an LED and resistor connected to
  pin 13, so you may not need any additional circuitry.

  But if you'd like to connect a second LED to pin 13, or use
  a different pin, follow these steps:

    Connect the positive side of your LED (longer leg) to Arduino
    digital pin 13 (or another digital pin, don't forget to change
    the code to match).

    Connect the negative side of your LED (shorter leg) to a
    330 Ohm resistor (orange-orange-brown). Connect the other side
    of the resistor to ground.

    pin 13 _____ + LED - _____ 330 Ohm _____ GND

    (We always use resistors between the Arduino and and LEDs
    to keep the LEDs from burning out due to too much current.)

This sketch was written by SparkFun Electronics,
```

```
with lots of help from the Arduino community.
This code is completely free for any use.
Visit http://learn.sparkfun.com/products/2 for SIK information.
Visit http://www.arduino.cc to learn about the Arduino.

Version 2.0 6/2012 MDG
*/
// Welcome to Arduino!
// If you're brand-new to this, there will be some new things to
// learn, but we'll jump right in and explain things as we go.

// The Arduino is a tiny computer that runs programs called
// "sketches". These are text files written using instructions
// the computer understances. You're reading a sketch right now.

// Sketches have computer code in them, but also (hopefully)
// "comments" that explain what the code does. Comments and code
// will have different colors in the editor so you can tell them
// apart.

// This is a comment - anything on a line after "//" is ignored
// by the computer.

/* This is also a comment - this one can be multi-line, but it
must start and end with these characters */

// A "function" is a named block of code, that performs a specific,
// well, function. Many useful functions are already built-in to
// the Arduino; others you'll name and write yourself for your
// own purposes.

// All Arduino sketches MUST have two specific functions, named
// "setup()" and "loop()". The Arduino runs these functions
// automatically when it starts up or if you press the reset
// button. You'll typically fill these function "shells" with your
// own code. Let's get started!
```

```
// The setup() function runs once when the sketch starts.
// You'll use it for things you need to do first, or only once:


void setup()
{
  // The Arduino has 13 digital input/output pins. These pins
  // can be configured as either inputs or outputs. We set this
  // up with a built-in function called pinMode().

  // The pinMode() function takes two values, which you type in
  // the parenthesis after the function name. The first value is
  // a pin number, the second value is the word INPUT or OUTPUT.

  // Here we'll set up pin 13 (the one connected to a LED) to be
  // an output. We're doing this because we need to send voltage
  // "out" of the Arduino to the LED.

  pinMode(13, OUTPUT);

  // By the way, the Arduino offers many useful built-in functions
  // like this one. You can find information on all of them at the
  // Arduino website: http://arduino.cc/en/Reference
}
// After setup() finishes, the loop() function runs over and over
// again, forever (or until you turn off or reset the Arduino).
// This is usually where the bulk of your program lives:


void loop()
{
  // The 13 digital pins on your Arduino are great at inputting
  // and outputting on/off, or "digital" signals. These signals
  // will always be either 5 Volts (which we call "HIGH"), or
```

```
  // 0 Volts (which we call "LOW").

  // Because we have an LED connected to pin 13, if we make that
  // output HIGH, the LED will get voltage and light up. If we make
  // that output LOW, the LED will have no voltage and turn off.

  // digitalWrite() is the built-in function we use to make an
  // output pin HIGH or LOW. It takes two values; a pin number,
  // followed by the word HIGH or LOW:

  digitalWrite(13, HIGH);   // Turn on the LED

  // delay() is a function that pauses for a given amount of time.
  // It takes one value, the amount of time to wait, measured in
  // milliseconds. There are 1000 milliseconds in a second, so if
  // you delay(1000), it will pause for exactly one second:

  delay(1000);              // Wait for one second

  digitalWrite(13, LOW);    // Turn off the LED

  delay(1000);              // Wait for one second
  // All together, the above code turns the LED on, waits one
  // second, turns it off, and waits another second.

  // When the computer gets to the end of the loop() function,
  // it starts loop() over again. So this program will continue
  // blinking the LED on and off!

  // Try changing the 1000 in the above delay() functions to
  // different numbers and see how it affects the timing. Smaller
  // values will make the loop run faster. (Why?)
}
```