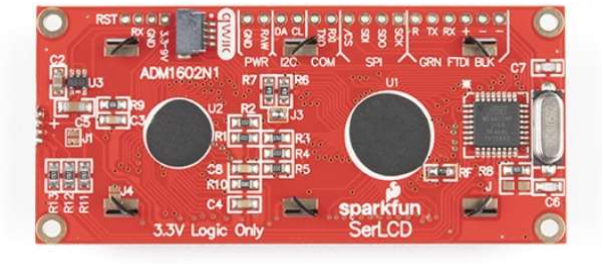
AVR-Based Serial Enabled LCDs Hookup Guide

CONTRIBUTORS:  *QCPETE*

Introduction

Heads up! The latest versions of these LCDs have a Qwiic connector. The previous versions did not. Some of the pictures in this tutorial are from the previous version of the hardware, however because the change is rather minor, they are still relevant. The pinouts along the edge are still the same order and position (except for the left-most cluster of 4 PTH pins).

	
Newer versions have qwiic connector! *Silk will look like "ADM1602N1".*	*Older versions do not have a qwiic connector.* *Silk will look like "ADM1602N_v2.6"*

Also note! Throughout the tutorial, the name "**SerLCD**" will refer to the hardware. **OpenLCD** will refer to the firmware.

The AVR-based serial enabled LCD (a.k.a. SerLCD) is a simple and cost effective solution for adding Liquid Crystal Displays (LCDs) into your project. The PCB design on the back of the screen includes an ATmega328P that handles all of the screen control. It can accept commands via serial, I^2C and SPI. The latest versions also include a Qwiic connector for solder-less single-cable connection setup. This simplifies the number of wires needed and allows your project to display all kinds of text and numbers. We offer three varieties of the AVR-based Serial Enabled LCDs:

SparkFun 16x2 SerLCD - RGB Text (Qwiic)
◉ LCD-16397
$19.95

SparkFun 16x2 SerLCD - RGB Backlight (Qwiic)
◉ LCD-16396
$19.95

SparkFun 20x4 SerLCD - RGB Backlight (Qwiic)
LCD-16398
$24.95
★★★★★ 1

The firmware is fully opensource and available for download at the GitHub repo here:

OPENLCD FIRMWARE GITHUB REPOSITORY

This allows for any customizations you may need. Uploading firmware (custom or updates), is easily done from the Arduino IDE using a Serial Basic. See firmware update instructions in the troubleshooting section of this tutorial for more info.

Also note, the example code used below is all available in the repo (along with many more examples). Before beginning this tutorial, it's a good idea to clone the repository (or download the entire repo as a zip), to grab all of the examples. But if you prefer, you can always use the "COPY CODE" button on each of the examples below.

Note that these all have identical firmware and can accept the same commands. However, you must adjust your display characters and cursor position as necessary for each model. Also note, there is a jumper on the back of each screen, and this "tells" the firmware how to correctly set the lines and columns for each screen.

Product Showcase: SparkFun Qwiic SerLCDs

Required Materials

If you are using Qwiic, then you will only need a Redboard Qwiic, a Qwiic cable, and a Micro-B USB cable for programming.

For non-qwiic setups, you may need the following materials in this wishlist. Depending on what you have, you may not need everything on this list. Add it to your cart, read through the guide, and adjust the cart as necessary.

Wishlist for AVR-based Serial Enabled LCDs Hookup Guide SparkFun Wish List

SparkFun RedBoard - Programmed with Arduino
DEV-13975
At SparkFun we use many Arduinos and we're always looking for the simplest, most stable one. Each board is a bit…

SparkFun Logic Level Converter - Bi-Directional
BOB-12009
If you've ever tried to connect a 3.3V device to a 5V system, you know what a challenge it can be. The SparkFun bi…

Break Away Headers - Straight
PRT-00116
A row of headers - break to fit. 40 pins that can be cut to any size. Used with custom PCBs or general custom head…

Jumper Wires Standard 7" M/M - 30 AWG (30 Pack)
PRT-11026
If you need to knock up a quick prototype there's nothing like having a pile of jumper wires to speed things up, and …

Breadboard - Self-Adhesive (White)
PRT-12002
This is your tried and true white solderless breadboard. It has 2 power buses, 10 columns, and 30 rows - a total of …

Arduino and Breadboard Holder
DEV-11235
We've been prototyping for a long time on these awesome little plastic plates but it's time for an upgrade. This versi…

SparkFun USB Mini-B Cable - 6 Foot
CAB-11301
This is a USB 2.0 type A to Mini-B 5-pin cable. You know, the mini-B connector that usually comes with USB Hubs, …

Wall Adapter Power Supply - 5V DC 2A (Barrel Jack)
TOL-12889
This is a high quality switching 'wall wart' AC to DC 5V 2000mA Barrel Jack wall power supply manufactured specif…

Trimpot 10K Ohm with Knob
COM-09806
There are lots of trimpots out there. Some are very large, some so small they require a screwdriver. Here at SparkF…

SparkFun 16x2 SerLCD - Black on RGB 3.3V
LCD-14072

Tools

You may need a soldering iron, solder, and general soldering accessories, and screw driver depending on your setup.

Solder Lead Free - 100-gram Spool
TOL-09325
$7.95
★★★★★ 7

Pocket Screwdriver Set
TOL-12891
$3.95
★★★½☆ 5

Weller WLC100 Soldering Station
TOL-14228
$44.95
★★☆☆☆ 1

Suggested Reading

If you aren't familiar with the following concepts, we recommend checking out these tutorials before continuing.

How to Solder: Through-Hole Soldering

This tutorial covers everything you need to know about through-hole soldering.

Serial Communication

Asynchronous serial communication concepts: packets, signal levels, baud rates, UARTs and more!

Serial Peripheral Interface (SPI)

SPI is commonly used to connect microcontrollers to peripherals such as sensors, shift registers, and SD cards.

What is an Arduino?

What is this 'Arduino' thing anyway? This tutorials dives into what an Arduino is and along with Arduino projects and widgets.

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

I2C
An introduction to I2C, one of the main embedded communications protocols in use today.

ASCII
A brief history of how ASCII came to be, how it's useful to computers, and some helpful tables to convert numbers to characters.

Note: Click on any of the images in this tutorial for a closer look!

Hardware Overview

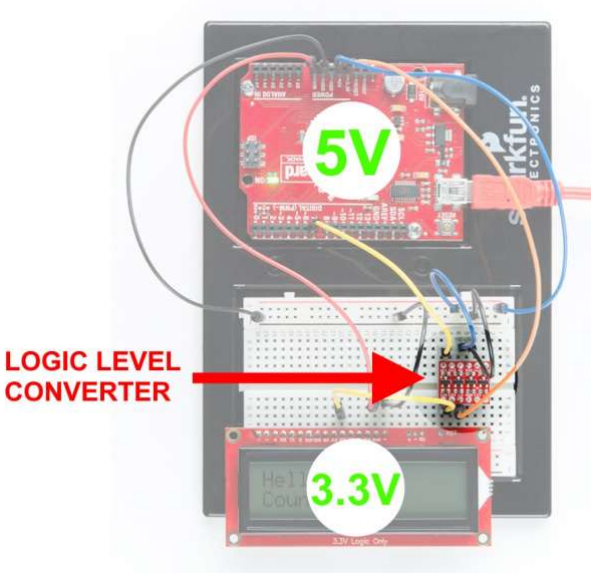
Features

The AVR-based SerLCD has many features that make it powerful and economical:

- Qwiic connector allows for single cable solution that provides connections for both power and I^2C communication lines.
- AVR microcontroller utilizes 11.0592 MHz crystal for greater communication accuracy
    - Adjustable baud rates of 1200, 2400, 4800, **9600 (default)**, 14400, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000
- The AVR ATmega328P (with Arduino-compatible bootloader) is populated on the back of each LCD screen and handles all of the LCD control
- 3 communication options
    - Serial UART, I^2C and SPI
- Adjustable I^2C address controlled via software special commands **(0x72 default)**
- Emergency reset to factory settings (Jumper RX to GND on bootup)
- Operational backspace character
- Incoming buffer stores up to 80 characters
- Pulse width modulation of backlight allows direct control of backlight brightness and current consumption
- Pulse width modulation of contrast allows for software defined contrast amount. The previous backpack versions of this product required adjusting the contrast via a hardware trim-pot which was less precise and less accessible in most enclosed projects.
- User definable splash screen
- Open-sourced firmware and Arduino-compatible bootloader enables updates via the Arduino IDE

AVR-Based Serial Controller (3.3V Logic Only)

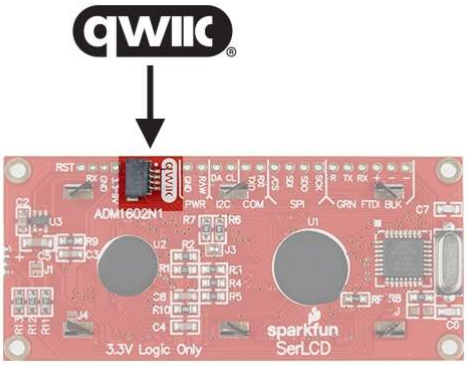
Using these screens, it is easy to connect to any microcontroller that has a serial UART, I^2C, or SPI. If you are using Qwiic to control your screen, then all you need is a qwiic cable. However, if you choose to use something other than Qwiic (with connections located on the PTH headers), then ***please make sure you convert to 3.3V Logic!*** In the example below, we chose to use our 5V RedBoard with a Logic Level Converter.



Example setup including 5V Redboard and Logic Level Converter

Connection Options

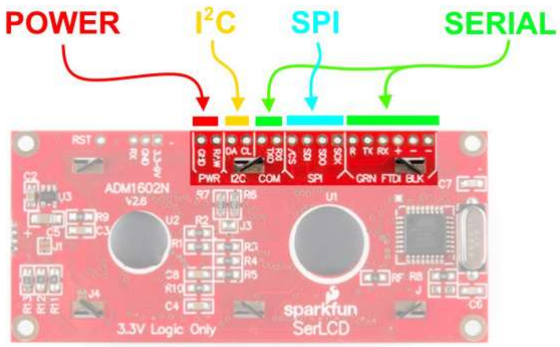
The latest versions of the SerLCD have a qwiic connector on the back.



Qwiic connector is located on the back side of the screens.

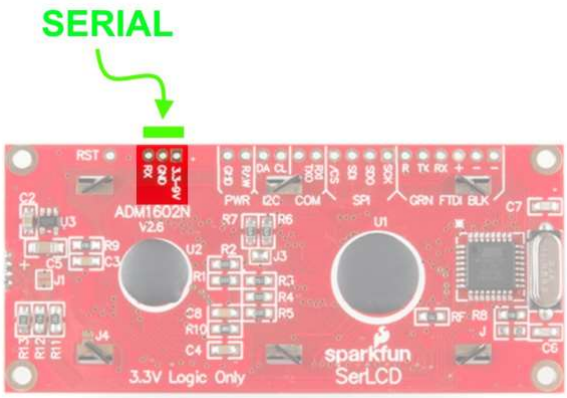
All Qwiic compatible products operate their I^{2}C on the cable at a logic level of 3.3V, and so no conversion is necessary. Also note, the connector is a right-angled syle connector, so that if your project does not have much space on the back side of the screen, then you can exit your cable with a lower profile. We did not put it exiting on the edge of the board (as most Qwiic products do), so that if your project and/or enclosure is tight on the sides of the screen, the Qwiic cable will not conflict.

Both sizes of these screens (16x2 and 20x4) have a row of headers along the top side. This is where you can connect power and your choice of communication protocol (Serial UART, I^{2}C, or SPI). It also has our 6-pin Arduino Serial port available for convenient firmware updates.



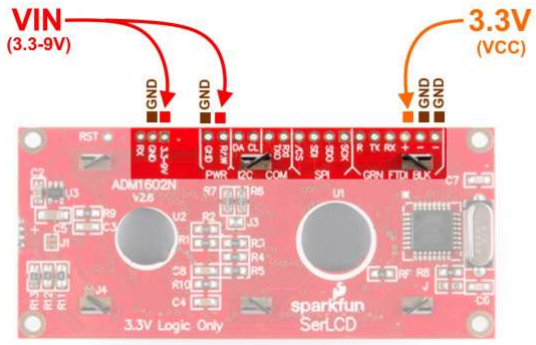
Main header pinouts

Note, if you choose Serial UART, there is a handy 3-pin JST footprint. This includes the minimum connections needed: RX, GND and VIN. Our JST Jumper 3 Wire Assembly is a good way to go:

	
Location on screen	*JST Jumper 3 Wire Assembly*

Input Voltage (VDD) and Logic Levels

All of these screens can be powered by **3.3-9V**. You have two pin options for connecting up power. One is labeled "RAW" and the other (on the 3-pin JST) is labeled "3.3-9V". ***Note, the pin labeled "+" is NOT a power input pin!*** This is connected to the VCC of the ATmega329P (3.3V).



Power Input pin options and ATmega328 VCC pin

Pro tip: If you plan to run 100% brightness on all three colors (red, green, blue), then it would be best to keep your power input voltage low. As close to 3.3V as possible is best. This will keep the on-board *linear* 3.3V voltage regulator nice and cool. It can accept up to 9V and power all three backlights at 100%, but it will get a little warm, and over years of use, potentially damage the vreg. For more information about vregs and why they heat up sometimes, please check out our tutorial: Power and Thermal Dissipation

⚡ **Warning!** ONLY USE **3.3V** LOGIC LEVELS when connecting to any of the data pins on the LCD screens. Connecting directly from a 5V microcontroller (such as the Redboard) will damage your LCD screen permanantly!

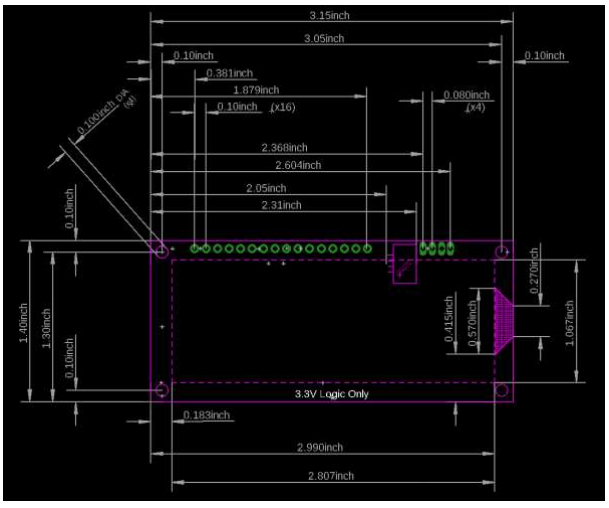
Contrast Control

The on-board ATmega328P controls the contrast of the screen using a PWM signal. This can be adjusted by sending a command via Serial UART, I^2C, or SPI. The screens ship out with the contrast setting set to 10, which we have found works well for most environments. Temperature and supply voltage can affect the contrast of the LCD, so you may need to adjust it accordingly. For more info about contrast and a detailed example, please see the section below called: Serial UART: Example Code - Contrast Control with a Trimpot.

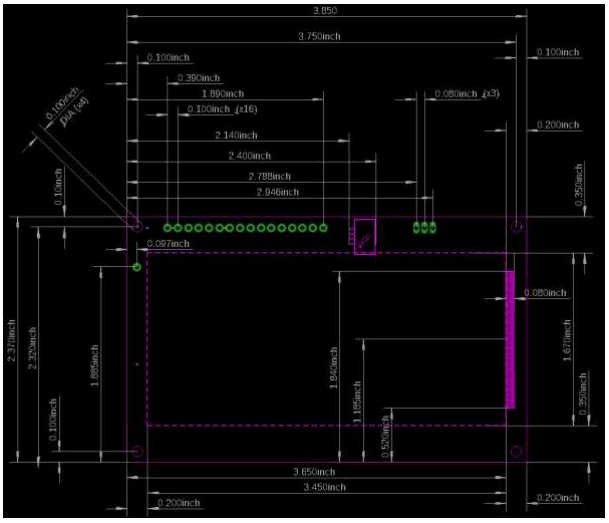
Dimensions

If you plan to mount a SerLCD into your project, please take a look at the following dimensional drawings. We have provided them in both PDF and PNG file format for both the 16x2 and the 20x4 Qwiic versions. Also, note that the tolerances of manufacturing are not always perfect, so please allow for some buffer room if designing around these numbers. Otherwise, your enclosure may be too tight or possibly not fit at all.

DIMENSIONAL PDF FILE (16X2)	**DIMENSIONAL PNG FILE (16X2)**
DIMENSIONAL PDF FILE (20X4)	**DIMENSIONAL PNG FILE (20X4)**



16x2 Dimensional PNG



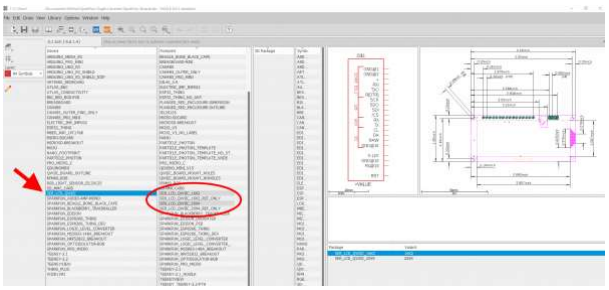
20x4 Dimensional PNG

Also note, the standoff holes on these screens do *not* fit our 4-40 sized screws or hex metal standoffs. You will need to use something slightly smaller in diameter, like our 2-56 screws.

If you would like to include an LCD as a device in your eagle design, then you will find the SparkFun-boards.lbr very useful. It is located in the larger github repository that contains all of the SparkFun Eagle Libraries.

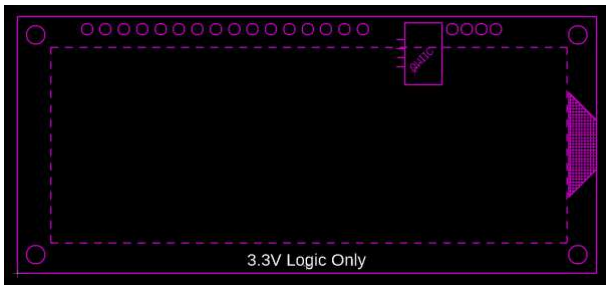
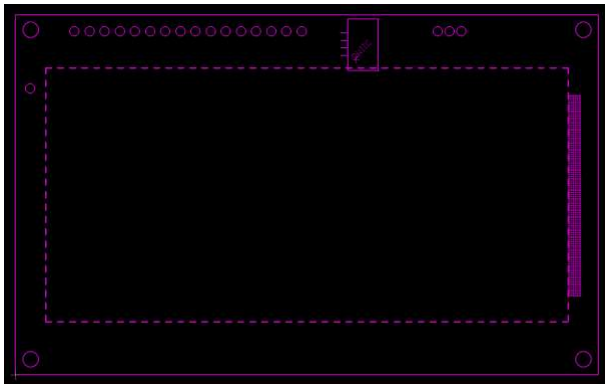
SPARKFUN EAGLE LIBRARIES GITHUB

Inside the library you will find the *SER_LCD_QWIIC* devices that allow you to simply drop this into your design and start netting/routing as desired!



Our Eagle device shown in the Eagle Library Manager.

As another option, we have included footprints that have *only* the reference layers, with no electrical connections (that is, no device with a symbol and linked pins/pads).

	
SER_LCD_QWIIC_16X2_REF_ONLY.fpt	*SER_LCD_QWIIC_20X4_REF_ONLY.fpt*

These "reference only" footprints are useful for dropping into a design and then quickly knowing the location of your standoff holes, the screen, the qwiic connector and such. If you plan to use the qwiic connection, but still want to mount a SerLCD into your PCB design, the "reference only" footprints are the best option. You can add these as a part directly into your board editor, and you'll be ready to drop in standoff holes and start designing.

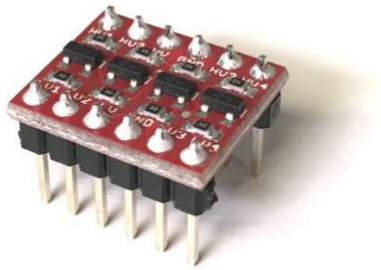
Hardware Hookup - Initial

Heads up! If you are using a Qwiic cable to power/control your SerLCD, then it is not necessary to solder in any headers or do any logic level conversion. Simply jump ahead to the I²C Hardware Hookup and Example Code to get up and running quickly.

Install Headers

For this tutorial, we are going to try out Serial UART, I²C, and SPI. In order to easily follow along using a breadboard, solder some headers to the connection ports along the side of your screen. Also, solder some headers onto either side of the logic level converter. This will allow us to easily plug these into the breadboard and wire each data line up with jumper wires.

Headers on screen



Headers on logic level converter

Note, we are going to use the 16x2 model with RGB backlight during this tutorial. If you are using a different model (RGB text or the 20x4), then the header pin-out and spacing is identical.

Connecting to an Arduino

Warning! Do not use the TX pin from your Arduino (aka "hardware serial TX") to control your LCD.

The TX pin is used for Serial Uploads of new sketches onto your Arduino, and will cause problems for both your Arduino and the LCD. In other words, when you upload new code to your "project" Arduino, it will be confused because the screen is sharing that TX pin. Please only use software serial to control your LCD. For all of the Serial UART examples, we have setup software serial on D7.

Warning! The RX input should be a TTL-level signal of 3.3V. If you are using a 5V Arduino, you will need a logic level converter between the two.

SparkFun Logic Level Converter - Bi-Directional

BOB-12009

$2.95

★★★★½ 106

SparkFun Voltage-Level Translator Breakout - TXB0104

BOB-11771

$4.50

★★★★☆ 7

SparkFun Logic Level Converter - Single Supply

PRT-14765

$14.95

SparkFun Level Translator Breakout - PCA9306

BOB-11955

$6.95

★★★★★ 3

You should **NOT** connect the board directly to RS232-level voltages (which are around +/-10V). This will damage the board. For more information on RS232, see our explanation here). If you do wish to connect this display to RS232 signals, you can use a level-shifting board to translate the RS232 signals to TTL-level signals.

SparkFun Transceiver Breakout - MAX3232

BOB-11189

$5.95

★★★★½ 28

SparkFun RS232 Shifter - SMD

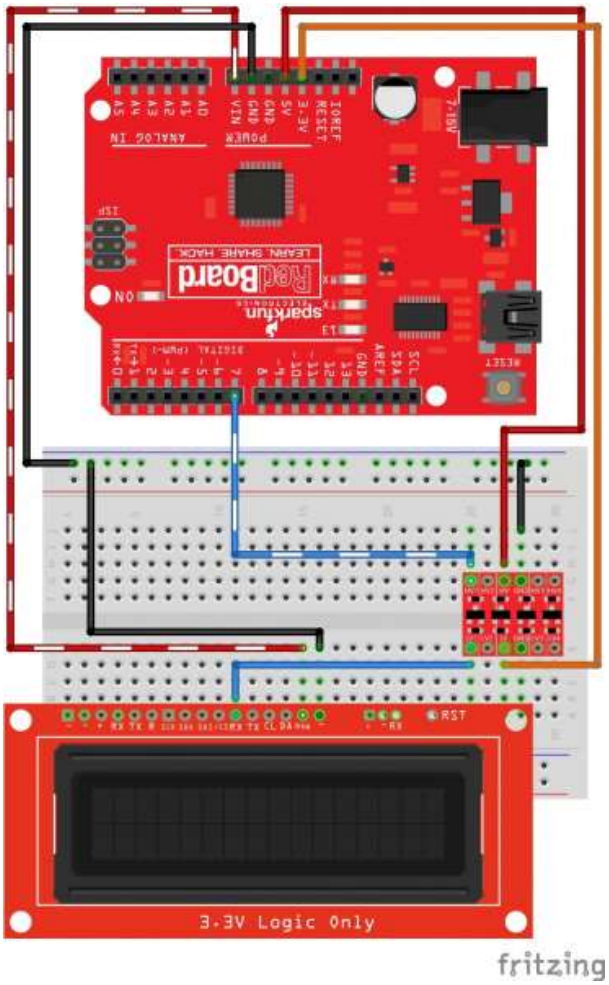
PRT-00449

$15.95

★★★★½ 14

SparkFun RS232 Shifter SMD (No DB9)
PRT-08780
$10.95
★★★½☆ 9

For the first set of examples in this tutorial (SERIAL UART), there are three connections you need to make to the LCD: RX, GND, and RAW. For the other communication protocols that we will explore later, you will need to wire up some other lines. Please see the following Fritzing graphic to see how to wire up your connections through a logic level converter.



SparkFun SerLCD Library

Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

The SparkFun SerLCD Arduino library demonstrates all the bells and whistles of the SerLCD. The SparkFun SerLCD Arduino library can be downloaded with the Arduino library manager by searching '**SparkFun SerLCD**' or you can grab the zip here from the GitHub repository:

SPARKFUN SERLCD ARDUINO LIBRARY (ZIP)

Once you have the library installed checkout the various examples.

- **Example1:** Print text to the display over I2C. Want to use serial? Checkout the *Serial* subfolder.
- **Example2:** Change the brightness of the RGB backlight.
- **Example3:** Set the cursor to a specific position. This really increases the speed at which you can update the display because you only need to change the characters that are changing.
- **Example4:** Move the cursor around the screen
- **Example5:** Enable/disable the cursor
- **Example6:** Make the cursor blink
- **Example7:** Enable automatic scrolling
- **Example8:** More scrolling with text
- **Example9:** Load a custom character
- **Example10:** Power down the display
- **Example11:** Change the where the next character will be printed (left-to-right or right-to-left).
- **Example12:** Read characters from serial and display them over I2C
- **Example13:** Rapidly update the backlight color
- **Example14:** Show the devices current firmware version
- **Example15:** Enable/Disable the displaying of system messages
- **Example16:** Create custom splash screen and enable/disable it.
- **Serial** ** **Example1:** Print to the display using serial instead of I2C

This SparkFun SerLCD Library really focuses on I^2C because it's faster than serial and supports daisy-chaining. If you'd prefer serial, it's as simple as

```
lcd.begin(Serial);
```

Checkout the various examples to see the available functions to call for all the features but any and all can be used over serial, I2C, or SPI.

Firmware Overview

Note: The examples in this tutorial assume you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE.

The easiest way to control the display is to use the SparkFun SerLCD Arduino Library. But if you prefer, you can connect using any device that can produce serial UART, I2C, or SPI.

The most basic way to use these screens is to simply send characters to them. They will display on the screen as you send them, and then if you go beyond the last character of the screen, it will begin overwriting from the first spot.

You can choose to send characters over Serial UART, I^2C, or SPI. The best way to learn about each communication protocol is to try out the "basic" example located in the GitHub repository.

GITHUB: OPENLCD/FIRMWARE/EXAMPLES

Heads up! Throughout the tutorial, the name "**SerLCD**" will refer to the hardware throughout the tutorial. **OpenLCD** will refer to the firmware used on the SerLCD.

In addition to basic operation, there are special commands you can send the screen for special operations (like "clear screen" or set cursor position) and configuration settings (like BAUD RATE).

Configuration & Command Set

Note: The following commands are valid for users with **OpenLCD Firmware v1.1+**. The latest Serial Enabled LCDs shipped after February 2019 will have the firmware version. If you are having any issues using the commands listed below, make sure to update the firmware.

The special pipe character '|' (also called the 'or' character) is used to tell the screen to enter "settings mode". You then follow this command with another special character (usually "ctrl+c", ctrl+d, etc.). A complete table of commands are shown below in ASCII, DEC and HEX. Any one of these representations is acceptable when sending a command character.

The HD44780 LCD controller is very common. The extended commands for this chip include but are not limited to those described in table. Please refer to the HD44780 datasheet for more information.

Note, this "cheat sheet" is also located at the top of each example code section for easy reference.

ASCII	DEC	HEX	Description
'|'	124	0x7C	Enter Settings Mode. *Note: When a second `|` ASCII character is sent to the display, the screen will escape the settings mode and display the same second vertical "pipe" as text. This feature is supported on v1.3+ of the SerLCD.*
ctrl+h	8	0x08	Software reset of the system
ctrl+i	9	0x09	Enable/disable splash screen
ctrl+j	10	0x0A	Save currently displayed text as splash
ctrl+k	11	0x0B	Change baud to 2400bps
ctrl+l	12	0x0C	Change baud to 4800bps

ctrl+m	13	0x0D	Change baud to 9600bps
ctrl+n	14	0x0E	Change baud to 14400bps
ctrl+o	15	0x0F	Change baud to 19200bps
ctrl+p	16	0x10	Change baud to 38400bps
ctrl+q	17	0x11	Change baud to 57600bps
ctrl+r	18	0x12	Change baud to 115200bps
ctrl+s	19	0x13	Change baud to 230400bps
ctrl+t	20	0x14	Change baud to 460800bps
ctrl+u	21	0x15	Change baud to 921600bps
ctrl+v	22	0x16	Change baud to 1000000bps
ctrl+w	23	0x17	Change baud to 1200bps
ctrl+x	24	0x18	Change the contrast. Follow Ctrl+x with number 0 to 255. 40 is default.
ctrl+y	25	0x19	Change the TWI address. Follow Ctrl+y with number 0 to 255. 114 (0x72) is default.
ctrl+z	26	0x1A	Enable/disable ignore RX pin on startup (ignore emergency reset)
ESC to ""	27-34	0x1B-0x22	Record line of pixel data to a custom character. Up to eight characters can be recorded. Each character is made up of 7 lines of pixel data. Each line of pixel data must be proceeded by the character position to record within. 0x1B = character 0. 0x22 = character 7. For example, 0x1C 01 1C 03 1C 07 1C 0F 1C 1F 1C 3F 1C 7F will draw a triangle in character location 1. Saved to NVM.
'#' to '*'	35-42	0x23-0x2A	Display custom character 0 through 7. For example, 0x24 will display the character stored in location 1. 0x28 will display character stored in location 5.
'+'	43	0x2B	Set RGB Backlight. Follow this command with three bytes representing Red, Green, Blue, backlight values. Supported on v1.1+ of SerLCD.
','	44	0x2C	Display current SerLCD firmware version. Supported on v1.1+ of SerLCD. If this command doesn't display anything version is v1.0.
'-'	45	0x2D	Clear display. Move cursor to home position.
'.'	46	0x2E	Enable system messages as settings change (ie, 'Contrast: 5'). Supported on v1.2+ of SerLCD.
'/'	47	0x2F	Disable the displaying of system messages. Supported on v1.2+ of SerLCD.

'0'	48	0x30	Enable splash screen at power on. Similar to Ctrl+i command but definitive. Supported on v1.2+ of SerLCD.
'1'	49	0x31	Disable splash screen at power on. Similar to Ctrl+i command but definitive. Supported on v1.2+ of SerLCD.
n/a	128-157	0x80-0x9D	Set the primary backlight brightness. 128 = Off, 157 = 100%.
n/a	158-187	0x9E-0xBB	Set the green backlight brightness. 158 = Off, 187 = 100%.
n/a	188-217	0xBC-0xD9	Set the blue backlight brightness. 188 = Off, 217 = 100%.
ctrl+c to ctrl+g	3-7	0x03-0x07	Depricated command: Send line width.

OpenLCD "Cheat Sheet" Command Set

Clear Screen and Set Cursor Position

Clear display and set cursor position are the two commands that are used frequently. To clear the screen, send the control character '|' followed by '-'. Clearing the screen resets the cursor position back to position 0 (i.e. the first character on the first line).

Here's how you could do it doing a Serial UART write on software serial:

```
OpenLCD.write('|'); //Send setting character
OpenLCD.write('-'); //Send clear display character
```

Note, most of the example sketches in the repo use these two commands during setup(), so you can try any of the examples out to see these commands in action.

To set the active cursor position, send the control character **254** followed by **128 + row + position**. To give this a shot, check out the complete example sketch in the GitHub repo or copy and paste the following into your Arduino IDE:

```
/*
 OpenLCD is an LCD with Serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).

 OpenLCD gives the user multiple interfaces (serial, I2C, and SPI) to control an LCD. SerLCD was
the original
 serial LCD from SparkFun that ran on the PIC 16F88 with only a serial interface and limited fea
ture set.
 This is an updated serial LCD.

 This example shows how to change the position of the cursor. This is very important as this is
 the
 fastest way to update the screen, ie - rather than clearing the display and re-transmitting a h
andful of bytes
 a cursor move allows us to re-paint only what we need to update.

 We assume the module is currently at default 9600bps.

 We use software serial because if OpenLCD is attached to an Arduino's hardware serial port duri
ng bootloading
 it can cause problems for both devices.

 Note: If OpenLCD gets into an unknown state or you otherwise can't communicate with it send 18
 (0x12 or ctrl+r)
 at 9600 baud while the splash screen is active and the unit will reset to 9600 baud.

 Emergency reset: If you get OpenLCD stuck into an unknown baud rate, unknown I2C address, etc,
 there is a
 safety mechanism built-in. Tie the RX pin to ground and power up OpenLCD. You should see the sp
lash screen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset to 9600bps with a I2C address of 0x72. Note: This featu
re can be
 disabled if necessary. See *Ignore Emergency Reset* for more information.

 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
```

```
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.

 For example, to change the baud rate to 115200 send 124 followed by 18.

*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX (not used), TX

int counter = 250;

void setup()
{
  Serial.begin(9600); //Start serial communication at 9600 for debug statements
  Serial.println("OpenLCD Example Code");

  OpenLCD.begin(115200); //Begin communication with OpenLCD

  //Send the reset command to the display - this forces the cursor to return to the beginning of
the display
  OpenLCD.write('|'); //Send setting character
  OpenLCD.write('-'); //Send clear display character

  OpenLCD.print("Hello World!    Counter: "); //For 16x2 LCDs
  //OpenLCD.print("Hello World!        Counter: "); //For 20x4 LCDs
}

void loop()
{
  OpenLCD.write(254); //Send command character
```

```
  OpenLCD.write(128 + 64 + 9); //Change the position (128) of the cursor to 2nd row (64), positi
on 9 (9)

  OpenLCD.print(counter++); //Re-print the counter
  //OpenLCD.print(" "); //When the counter wraps back to 0 it leaves artifacts on the display

  delay(2); //Hang out for a bit if we are running at 115200bps
}
```

The two lines of code that are actually changing the cursor position are as follows:

```
OpenLCD.write(254); //Send command character
OpenLCD.write(128 + 64 + 9); //Change the position (128) of the cursor to 2nd row (64), position
9 (9)
```

The example sets the cursor to the second row, position 9. To set the cursor to the first row, position 0, the command would look like this:

```
OpenLCD.write(254); //Send command character
OpenLCD.write(128 + 0 + 0); //Change the position (128) of the cursor to 1st row (0), position 0
(0)
```

Use the following tables to see row commands and available positions:

16 Character Displays	
Line Number (command)	**Viewable Cursor Positions**
1 (0)	0-15
2 (64)	0-15

20 Character Displays	
Line Number (command)	**Viewable Cursor Positions**
1 (0)	0-19
2 (64)	0-19
3 (20)	0-19
4 (84)	0-19

Setting Up the LCD Size

You should never need to send any screen size configuration commands to setup these screens. During *setup()*, the firmware looks at a specific pin that is either left floating or tied high. From this it can determine which screen size it is populated on (16x2 or 20x4). However, the commands are still available for manually setting the width and lines.

ASCII	DEC	HEX	Description
'|'	124	0x7C	Enter Settings Mode
ctrl+c	3	0x03	Change width to 20
ctrl+d	4	0x04	Change width to 16
ctrl+e	5	0x05	Change lines to 4
ctrl+f	6	0x06	Change lines to 2
ctrl+g	7	0x07	Change lines to 1

Changing the Baud Rate

The screens ship out with a default baud rate setting to **9600 baud**, but they can be set to a variety of baud rates. The 11.0592 MHz crystal provides greater clock accuracy and allows for baud rates up to 1MHz!

To change the baud rate, you will need to send two commands: First, send the "|" command to enter settings mode. Second, send the desired baud rate command. (For example, to set it to 57600, you'd send `|`, then `Ctrl + q`)

⚠ **Caution!** Once you change the baud rate, you need to change the baud rate of your controlling device to match this. To change the LCD's baud rate from 9600 to 57600, first enter the control character **0x7C** control and **0x11**. Then adjust your microcontroller's code to match the baud rate of 57600.

Here is a table will all of the available baud rate settings commands:

ASCII	DEC	HEX	Description
'|'	124	0x7C	Enter Settings Mode
ctrl+k	11	0x0B	Change baud to 2400bps
ctrl+l	12	0x0C	Change baud to 4800bps
ctrl+m	13	0x0D	Change baud to 9600bps
ctrl+n	14	0x0E	Change baud to 14400bps
ctrl+o	15	0x0F	Change baud to 19200bps
ctrl+p	16	0x10	Change baud to 38400bps
ctrl+q	17	0x11	Change baud to 57600bps
ctrl+r	18	0x12	Change baud to 115200bps
ctrl+s	19	0x13	Change baud to 230400bps

ctrl+t	20	0x14	Change baud to 460800bps
ctrl+u	21	0x15	Change baud to 921600bps
ctrl+v	22	0x16	Change baud to 1000000bps

OpenLCD Baud Rate Command Set

If your screen enters into an unknown state or you otherwise can't communicate with it, try sending a `Ctrl + r` (**0x12**) character at 9600 baud while the splash screen is active (during the first 500 ms of boot-up) and the unit will reset to 9600 baud.

Backlight Brightness

These screens provide you with control of the backlight to one of 30 different brightness levels on each of the colors (Red, Green and Blue). With this, you can mix colors together to get almost any custom color you'd like. This is also handy when power consumption of the unit must be minimized. By reducing the brightness, the overall backlight current consumption is reduced.

> **Note:** The LCD screens that have RGB text on black background, the control of the "backlight" is actually controlling the RGB values of the text brightness.

To change the backlight brightness, you will need to send two commands: First, send the "|" command to enter settings mode. Second, send a number that corresponds to the color and brightness you'd like to set. (For example, to set the Green backlight to 100%, you'd send `|`, then `187`). For some good example code on how to adjust any and all of the colors, check out the example sketch here or copy and paste the code below into your Arduino IDE:

```
/*
 OpenLCD is an LCD with serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).
 OpenLCD gives the user multiple interfaces (serial, I2C, and SPI) to control an LCD. SerLCD was
the original
 serial LCD from SparkFun that ran on the PIC 16F88 with only a serial interface and limited fea
ture set.
 This is an updated serial LCD.

 This example shows how to change the backlight brightness. We assume the module is currently at
default 9600bps.
 We use software serial because if OpenLCD is attached to an Arduino's hardware serial port duri
ng bootloading
 it can cause problems for both devices.
 Note: If OpenLCD gets into an unknown state or you otherwise can't communicate with it send 18
 (0x12 or ctrl+r)
 at 9600 baud while the splash screen is active and the unit will reset to 9600 baud.
 Emergency reset: If you get OpenLCD stuck into an unknown baud rate, unknown I2C address, etc,
 there is a
 safety mechanism built-in. Tie the RX pin to ground and power up OpenLCD. You should see the sp
lash screen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset to 9600bps with a I2C address of 0x72. Note: This featu
re can be
 disabled if necessary. See *Ignore Emergency Reset* for more information.
 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
```

```
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.

 For example, to change the baud rate to 115200 send 124 followed by 18.
*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX (not used), TX

byte counter = 0;

void setup()
{
  Serial.begin(9600); //Begin local communication for debug statements

  OpenLCD.begin(9600); //Begin communication with OpenLCD

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 0); //Set green backlight amount to 0%

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 0); //Set blue backlight amount to 0%
}

void loop()
{
  //Control red backlight
  Serial.println("Mono/Red backlight set to 0%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128); //Set white/red backlight amount to 0%

  delay(2000);

  //Control red backlight
  Serial.println("Mono/Red backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128 + 15); //Set white/red backlight amount to 51%

  delay(2000);

  //Control red backlight
```

```
  Serial.println("Mono/Red backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128 + 29); //Set white/red backlight amount to 100%

  delay(2000);

  //The following green and blue backlight control only apply if you have an RGB backlight enabl
ed LCD

  all_off(); // turn off all backlights - see function below

  //Control green backlight
  Serial.println("Green backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 15); //Set green backlight amount to 51%

  delay(2000);

  //Control green backlight
  Serial.println("Green backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 29); //Set green backlight amount to 100%

  delay(2000);

  all_off(); // turn off all backlights - see function below

  //Control blue backlight
  Serial.println("Blue backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 15); //Set blue backlight amount to 51%

  delay(2000);

  //Control blue backlight
  Serial.println("Blue backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 29); //Set blue backlight amount to 100%

  delay(2000);

  all_off(); // turn off all backlights - see function below

}

void all_off(void)
{
  // Set all colors to 0

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128); //Set white/red backlight amount to 0%

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 0); //Set green backlight amount to 0%
```

```
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 0); //Set blue backlight amount to 0%

  delay(2000);
}
```

And here is a table showing all of the available backlight settings and commands:

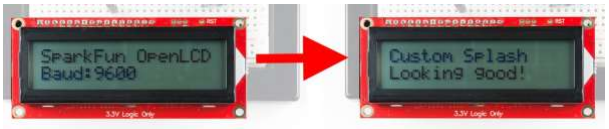
ASCII	DEC	HEX	Description
n/a	128-157	0x80-0x9D	Set the primary backlight brightness. 128 = Off, 157 = 100%.
n/a	158-187	0x9E-0xBB	Set the green backlight brightness. 158 = Off, 187 = 100%.
n/a	188-217	0xBC-0xD9	Set the blue backlight brightness. 188 = Off, 217 = 100%.

OpenLCD Backlight Command Set

Splash Screen

These LCD screens have a splash screen by default that reads "SparkFun OpenLCD Baud:9600". This splash screen verifies that the unit is powered, working correctly, and that the connection to the LCD is correct. The splash screen is displayed for 500 ms during boot-up and may be turned off if desired.

Setting Splash Screen



To make a custom splash screen, you need to send the characters you want to display (above we sent the characters "Custom Splash Looking good!") followed by `|`, then `Ctrl + j`). You will see a quick pop-up message display "Flash Recorded". Cycle power to test. You can also try out the example sketch located in the GitHub repo or copy and paste the code below into your Arduino IDE:

```
/*
 OpenLCD is an LCD with serial/I2C/SPI interfaces.
 By: Nathan Seidle, Pete Lewis
 SparkFun Electronics
 Date: 7/26/2018
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).
 OpenLCD gives the user multiple interfaces (serial, I2C, and SPI) to control an LCD. SerLCD was
the original
 serial LCD from SparkFun that ran on the PIC 16F88 with only a serial interface and limited fea
ture set.
 This is an updated serial LCD.

 This example shows how to change the Splash Screen contents. We assume the module is currently
 at default 9600bps.
 We use software serial because if OpenLCD is attached to an Arduino's hardware serial port duri
ng bootloading
 it can cause problems for both devices.
 Note: If OpenLCD gets into an unknown state or you otherwise can't communicate with it send 18
 (0x12 or ctrl+r)
 at 9600 baud while the splash screen is active and the unit will reset to 9600 baud.
 Emergency reset: If you get OpenLCD stuck into an unknown baud rate, unknown I2C address, etc,
 there is a
 safety mechanism built-in. Tie the RX pin to ground and power up OpenLCD. You should see the sp
lash screen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset to 9600bps with a I2C address of 0x72. Note: This featu
re can be
 disabled if necessary. See *Ignore Emergency Reset* for more information.
 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
```

```
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.

 For example, to change the baud rate to 115200 send 124 followed by 18.
*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX (not used), TX

void setup()
{
  Serial.begin(9600); //Begin local communication for debug statements

  OpenLCD.begin(9600); //Begin communication with OpenLCD

  delay(1000);

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write('-'); // clear screen

  delay(1000);

  OpenLCD.print("Custom Splash   Looking good!"); // Send our new content to display - this will
soon become our new splash screen.

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(10); //Set current contents to splash screen memory (this is also a "ctrl-j", if
you are doing it manually)

}

void loop()
{
  // nothing here, just doing this example in setup()
}
```

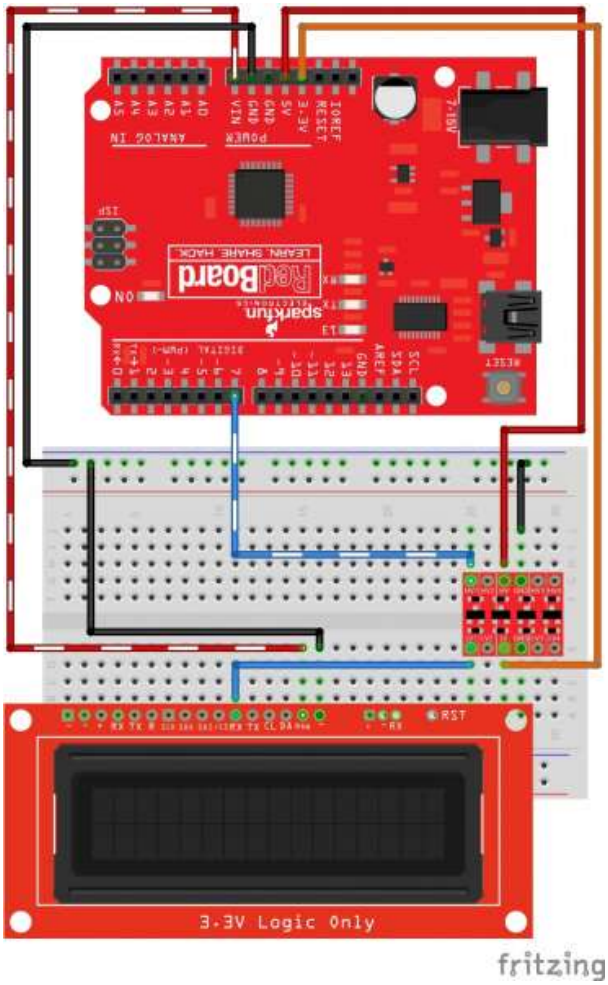
Turning Splash Screen On/Off

To disable the splash screen, send `|`, then `Ctrl + i`. Every time this command is sent to the unit, the splash screen display option will toggle. If the splash screen is currently being displayed, sending `|`, then `Ctrl + i` will disable the splash screen during the next boot, and sending `|`, then `Ctrl + i` characters again will enable the

splash screen.

Serial UART: Hardware Hookup

Here's how to setup your hardware for most of the Serial UART example code.



Note, some other examples require additional wiring (for example the contrast example requires a trimpot for variable control). We will show Fritzing graphics for each example covered in this tutorial. For the remaining tutorials, please look at the comments at the top of the example code for info on how to hookup the hardware.

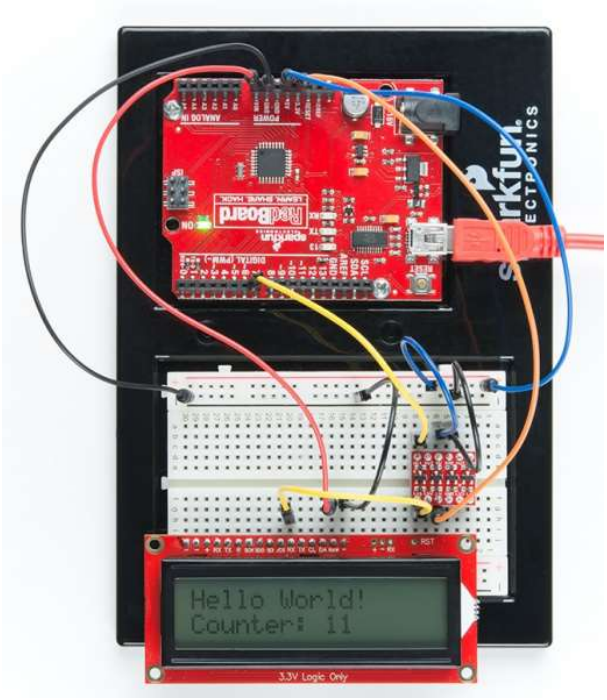
Serial UART: Example Code - Basic

You can download the latest example code for this experiment from the GitHub repo or you can copy and paste the following code into your Arduino IDE:

```
/*
 OpenLCD is an LCD with serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).

 This example shows how to display a counter on the display over serial. We use software serial
 because if
 OpenLCD is attached to an Arduino's hardware serial port during bootloading  it can cause probl
ems for both devices.

 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX, TX

byte counter = 0;
byte contrast = 2; //Lower is more contrast. 0 to 5 works for most displays.

void setup()
{
  Serial.begin(9600); //Start serial communication at 9600 for debug statements
  Serial.println("OpenLCD Example Code");

  OpenLCD.begin(9600); //Start communication with OpenLCD

  //Send contrast setting
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(24); //Send contrast command
  OpenLCD.write(contrast);

}

void loop()
{
  //Send the clear command to the display - this returns the cursor to the beginning of the disp
lay
  OpenLCD.write('|'); //Setting character
  OpenLCD.write('-'); //Clear display

  OpenLCD.print("Hello World!    Counter: "); //For 16x2 LCD
  //OpenLCD.print("Hello World!        Counter: "); //For 20x4 LCD
  OpenLCD.print(counter++);
```

```
  delay(250); //Hang out for a bit
}
```

Here's what you should see after uploading the code to your Arduino. Try changing the text with a different message!



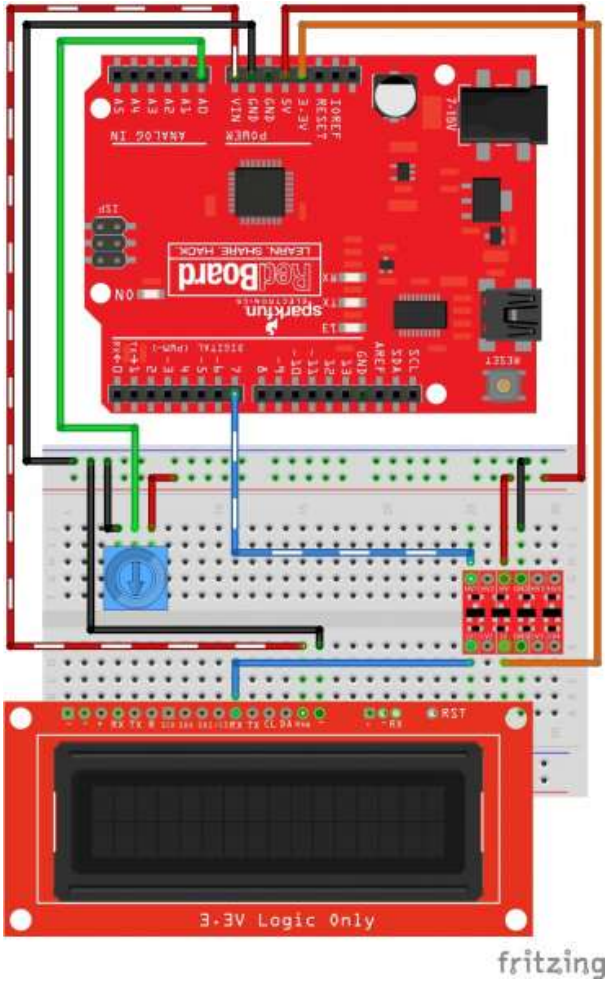
To send text to the board, wait 1/2 second (500ms) after power up for the splash screen to clear, then send text to the display through your serial port. The display understands all of the standard ASCII characters (upper and lowercase text, numbers, and punctuation), plus a number of graphic symbols and Japanese characters. See the HD44780 datasheet for the full list of supported characters.

If you send data that goes past the end of the first line, it will skip to the start of the second line. If you go past the end of the second line, the display will jump back up to the beginning of the first line.

Tip: You can simulate a scrolling window in software by copying the second line to the first line, and clearing the second line.

Serial UART: Example Code - Contrast Control with a Trimpot

For this contrast control example, you will need to wire up a trimpot. This will allow you to adjust the contrast in real time and find the best setting for your environment. Wire things up like this:



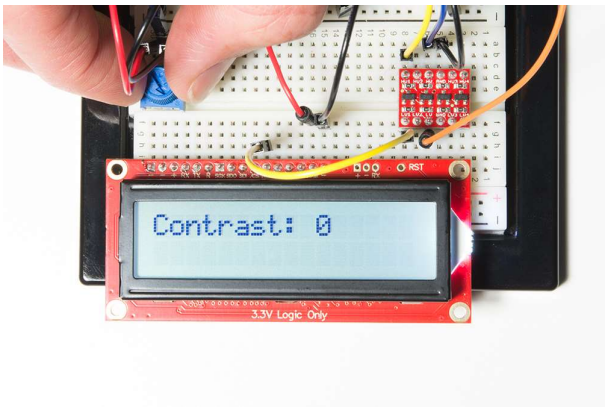
You can download the latest example code for this experiment from the GitHub repo or you can copy and paste the following code into your Arduino IDE:

```
 /*
 OpenLCD is an LCD with Serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).

 OpenLCD gives the user multiple interfaces (serial, I2C, and SPI) to control an LCD. SerLCD was
the original
 serial LCD from SparkFun that ran on the PIC 16F88 with only a serial interface and limited fea
ture set.
 This is an updated serial LCD.

 This example shows how to change the contrast using a trimpot. We assume the module is currentl
y at
 default 9600bps.

 We use software serial because if OpenLCD is attached to an Arduino's hardware serial port duri
ng bootloading
 it can cause problems for both devices.

 Note: If OpenLCD gets into an unknown state or you otherwise can't communicate with it send 18
 (0x12 or ctrl+r)
 at 9600 baud while the splash screen is active and the unit will reset to 9600 baud.

 Emergency reset: If you get OpenLCD stuck into an unknown baud rate, unknown I2C address, etc,
 there is a
 safety mechanism built-in. Tie the RX pin to ground and power up OpenLCD. You should see the sp
lash screen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset to 9600bps with a I2C address of 0x72. Note: This featu
re can be
 disabled if necessary. See *Ignore Emergency Reset* for more information.

 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

 Hook a trimpot up:
 Pin 1 - 5V
 Pin 2 - A0
 Pin 3 - GND

*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX (not used), TX

byte counter = 0;
```

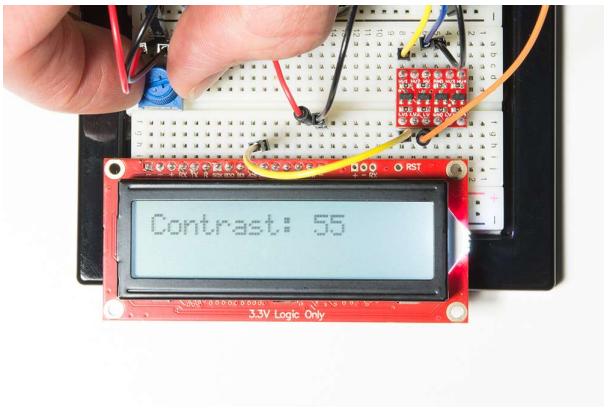
```
void setup()
{
  Serial.begin(9600); //Start serial communication at 9600 for debug statements
  Serial.println("OpenLCD Example Code");

  OpenLCD.begin(9600); //Begin communication with OpenLCD

  //Send the reset command to the display - this forces the cursor to return to the beginning of
the display
  OpenLCD.write('|'); //Send setting character
  OpenLCD.write('-'); //Send clear display character
  OpenLCD.print("Contrast test");

  pinMode(A0, INPUT);
}

int oldContrast = 0;
long startTime = 0;
bool settingSent = false;

void loop()
{
  int trimpot = averageAnalogRead(A0);
  int newContrast = map(trimpot, 0, 1023, 0, 255); //Map this analog value down to 0-255

  //Only send new contrast setting to display if the user changes the trimpot
  if(newContrast != oldContrast)
  {
    Serial.print("nc: ");
    Serial.println(newContrast);

    oldContrast = newContrast; //Update

    startTime = millis();
    settingSent = false;
  }

  //Wait at least 100ms for user to stop turning trimpot
  //OpenLCD displays the contrast setting for around 2 seconds so we can't send constant updates
  if(millis() - startTime > 500 && settingSent == false)
  {
    //Send contrast setting
    OpenLCD.write('|'); //Put LCD into setting mode
    OpenLCD.write(24); //Send contrast command
    OpenLCD.write(newContrast);

    settingSent = true;

    Serial.print("New contrast: ");
    Serial.println(newContrast);
  }

  delay(100); //Hang out for a bit
```

```
}

//Takes an average of readings on a given pin
//Returns the average
int averageAnalogRead(byte pinToRead)
{
  byte numberOfReadings = 8;
  unsigned int runningValue = 0;

  for(int x = 0 ; x < numberOfReadings ; x++)
    runningValue += analogRead(pinToRead);
  runningValue /= numberOfReadings;

  return(runningValue);
}
```

After uploading your sketch, you can now try adjusting the trimpot and watch the contrast change in real time. Here are a few examples that I see:

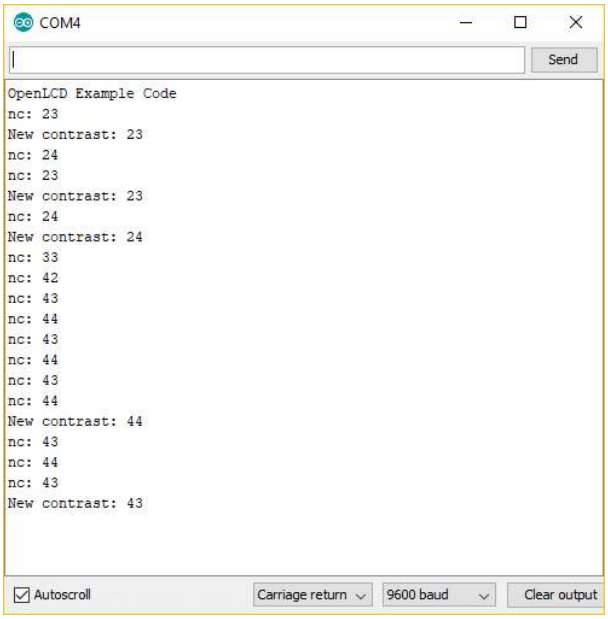


Contrast Setting: 0



Contrast Setting: 55

Note, if you are not seeing any text in the LCD, make sure and try rotating to either extreme. If you are up above 100, then in some cases you may not see any text. Watching your serial monitor from your Arduino can be helpful as well. It will tell you the settings as you are sending them to the LCD. Here is some example serial debug that I see while I adjust the trimpot:



Serial UART: Example Code - Backlight Control

For this next example, you can use the same exact hardware setup as in the previous two examples (Serial Basic or Serial Contrast). To jump right in and start playing with backlight control, you can get the latest example code from the Github repo or you can copy and paste the following code into your Arduino IDE:

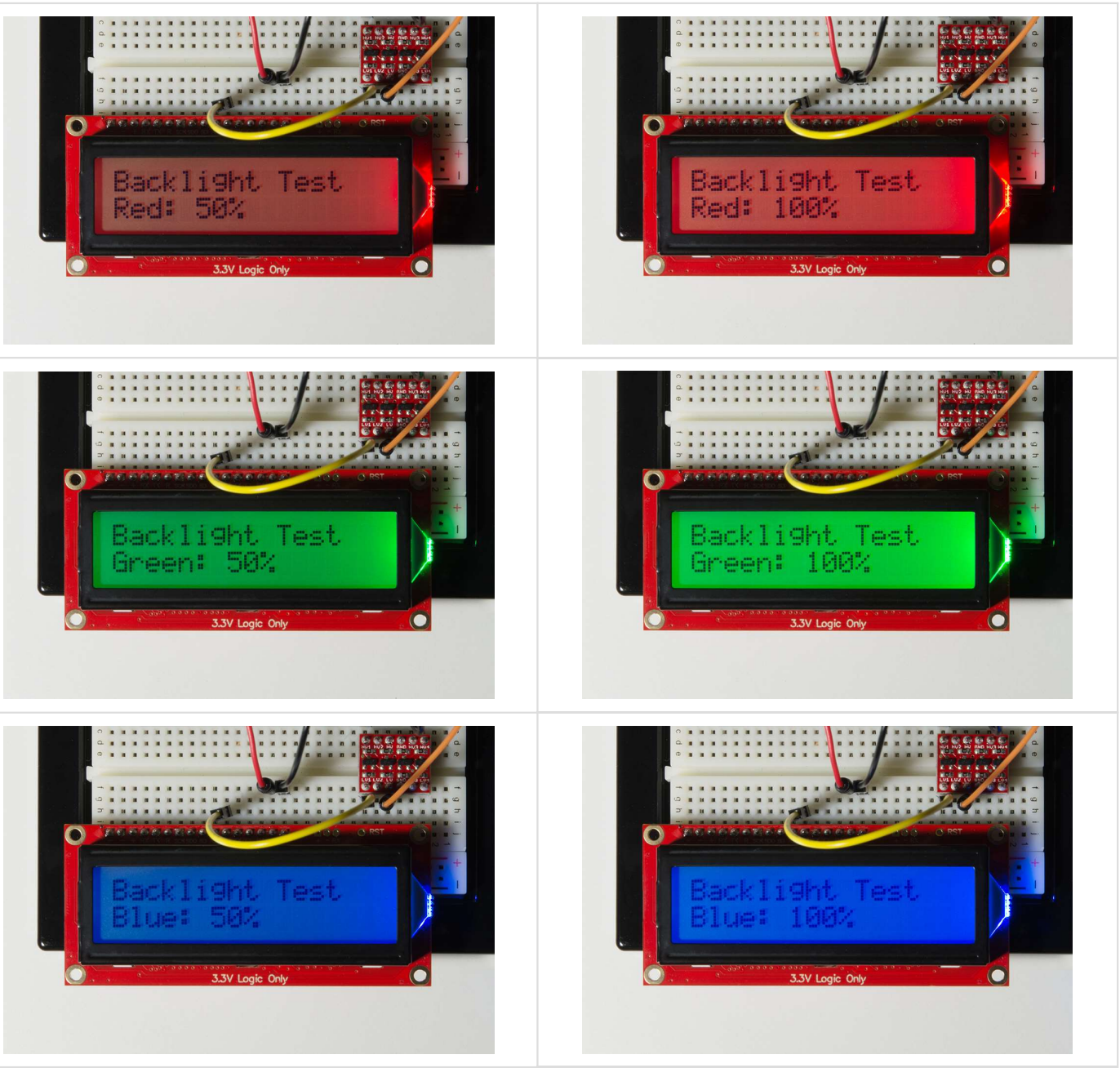
```
/*
 OpenLCD is an LCD with serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).
 OpenLCD gives the user multiple interfaces (serial, I2C, and SPI) to control an LCD. SerLCD was
the original
 serial LCD from SparkFun that ran on the PIC 16F88 with only a serial interface and limited fea
ture set.
 This is an updated serial LCD.

 This example shows how to change the backlight brightness. We assume the module is currently at
default 9600bps.
 We use software serial because if OpenLCD is attached to an Arduino's hardware serial port duri
ng bootloading
 it can cause problems for both devices.
 Note: If OpenLCD gets into an unknown state or you otherwise can't communicate with it send 18
 (0x12 or ctrl+r)
 at 9600 baud while the splash screen is active and the unit will reset to 9600 baud.
 Emergency reset: If you get OpenLCD stuck into an unknown baud rate, unknown I2C address, etc,
 there is a
 safety mechanism built-in. Tie the RX pin to ground and power up OpenLCD. You should see the sp
lash screen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset to 9600bps with a I2C address of 0x72. Note: This featu
re can be
 disabled if necessary. See *Ignore Emergency Reset* for more information.
 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 RX (OpenLCD) to Pin 7 (Arduino)
 VIN to 5V
 GND to GND

 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
```

```
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.

 For example, to change the baud rate to 115200 send 124 followed by 18.
*/

#include <SoftwareSerial.h>

SoftwareSerial OpenLCD(6, 7); //RX (not used), TX

byte counter = 0;

void setup()
{
  Serial.begin(9600); //Begin local communication for debug statements

  OpenLCD.begin(9600); //Begin communication with OpenLCD

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 0); //Set green backlight amount to 0%

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 0); //Set blue backlight amount to 0%
}

void loop()
{
  //Control red backlight
  Serial.println("Mono/Red backlight set to 0%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128); //Set white/red backlight amount to 0%

  delay(2000);

  //Control red backlight
  Serial.println("Mono/Red backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128 + 15); //Set white/red backlight amount to 51%

  delay(2000);

  //Control red backlight
```

```
  Serial.println("Mono/Red backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128 + 29); //Set white/red backlight amount to 100%

  delay(2000);

  //The following green and blue backlight control only apply if you have an RGB backlight enabl
ed LCD

  all_off(); // turn off all backlights - see function below

  //Control green backlight
  Serial.println("Green backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 15); //Set green backlight amount to 51%

  delay(2000);

  //Control green backlight
  Serial.println("Green backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 29); //Set green backlight amount to 100%

  delay(2000);

  all_off(); // turn off all backlights - see function below

  //Control blue backlight
  Serial.println("Blue backlight set to 51%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 15); //Set blue backlight amount to 51%

  delay(2000);

  //Control blue backlight
  Serial.println("Blue backlight set to 100%");
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 29); //Set blue backlight amount to 100%

  delay(2000);

  all_off(); // turn off all backlights - see function below

}

void all_off(void)
{
  // Set all colors to 0

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(128); //Set white/red backlight amount to 0%

  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(158 + 0); //Set green backlight amount to 0%
```

```
  OpenLCD.write('|'); //Put LCD into setting mode
  OpenLCD.write(188 + 0); //Set blue backlight amount to 0%

  delay(2000);
}
```

With this code running, you should see your backlight colors cycle through a pattern of 0%, then 50%, then 100%. It will show this for each color individually (Red, Green, and Blue). Here are some shots of what it looks like for me when I run the code. Note, yours may vary slightly due to your RAW input voltage and the temperature of your environment.



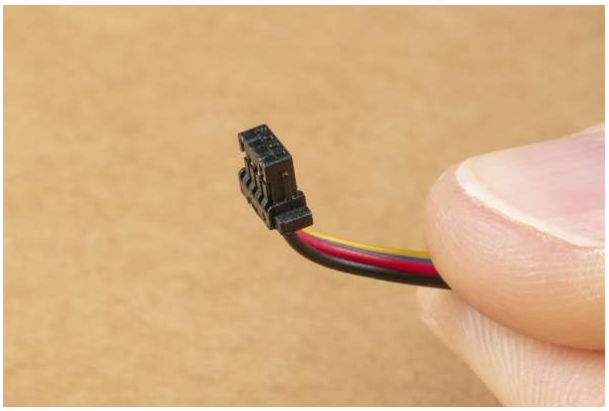
Tip: By mixing the amounts of each backlight color, you can create virtually any color you like. In the examples above, we are trying out each backlight on its own. For other colors, try combining different values

of each backlight. For example, to make a purple, you can try RED:29, GREEN:5, and BLUE:25. To make a yellow, try RED:22, GREEN:29, and BLUE:5. To make white, turn them all on to 29 (aka 100%).

I2C: Hardware Hookup & Example Code - Basic

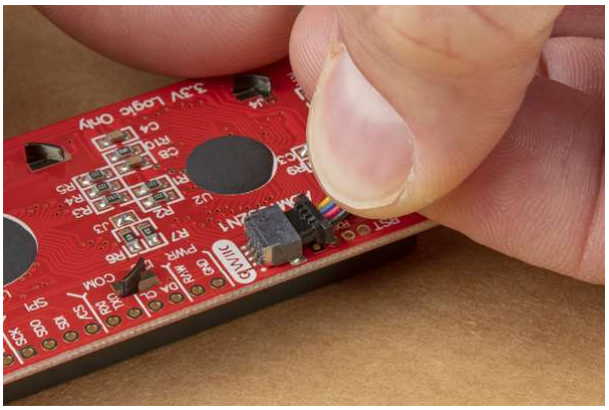
Qwiic connection

If you are using Qwiic, then you simply need to plug in a qwiic cable from your controller to your SerLCD. It helps to bend/curl your Qwiic cable a bit before inserting it into the right-angle connector on the SerLCD.



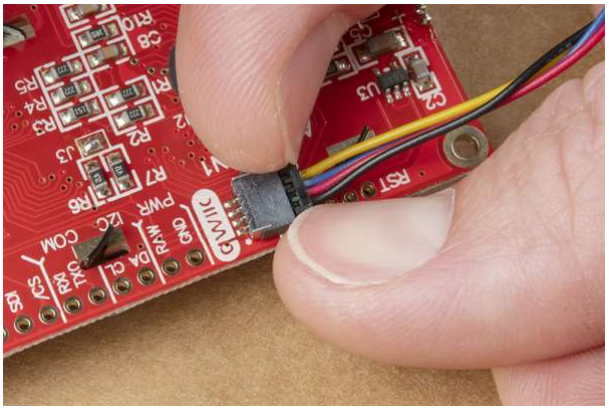
Qwiic Cable with a bend.

With the bend in place, you can better align it before inserting it all the way.



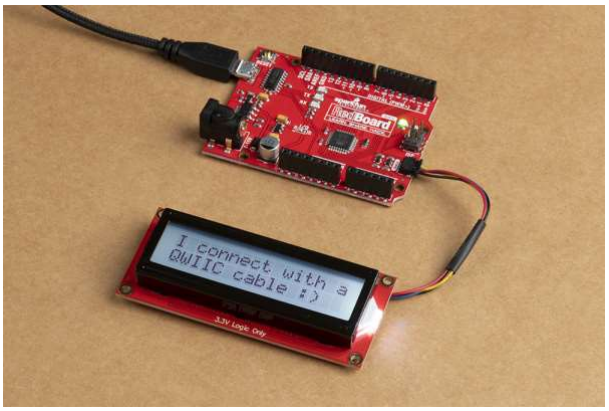
Initial alignment.

To avoid stressing the cable wires, it is best to push the Qwiic connector using your fingernails on the sides of the connector plastic. Tweezers can also do the trick.



Pressing on the sides of the plastic is ideal.

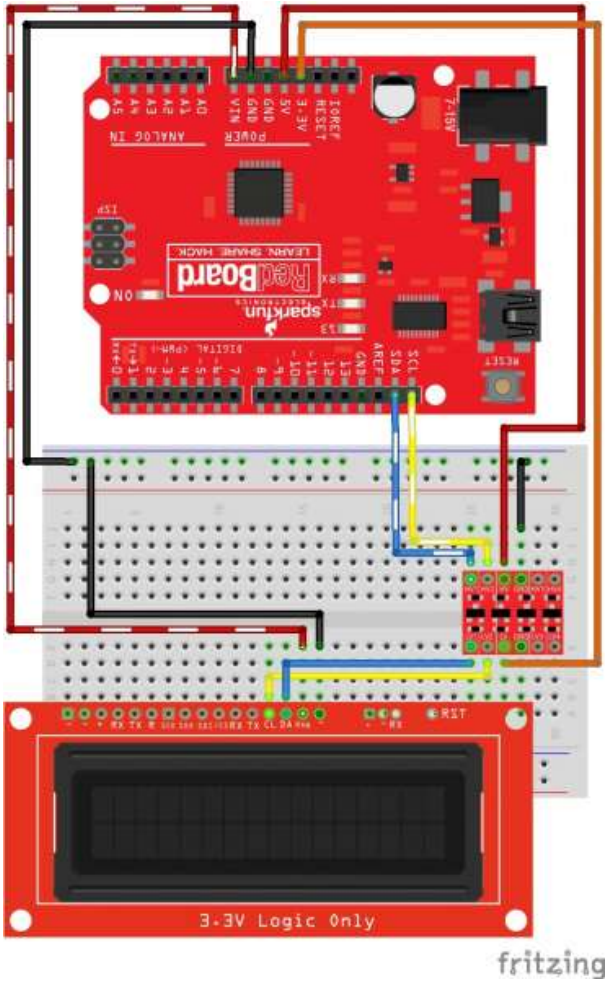
Connect the other end of your Qwiic cable to you controller of choice, and you will be ready to go! Here we are showing the Redboard Qwiic:



Redboard Qwiic connected to SerLCD via Qwiic cable.

Non-Qwiic I^{2}C connection

For I^{2}C, there are only 2 communication lines you need to connect: SDA and CLK. But remember, these *must* be 3.3V logic. So if you are using a 5V Redboard like we are, then you'll need to convert SDA and SCL from 5V to 3.3V. See the following Fritzing diagram for how you can wire this up:



Example Code

After you've got your I^2C lines wired up properly (via Qwiic or through logic level converter), you can get the latest example code from the GitHub repo or you can copy and paste the following code into your Arduino IDE:

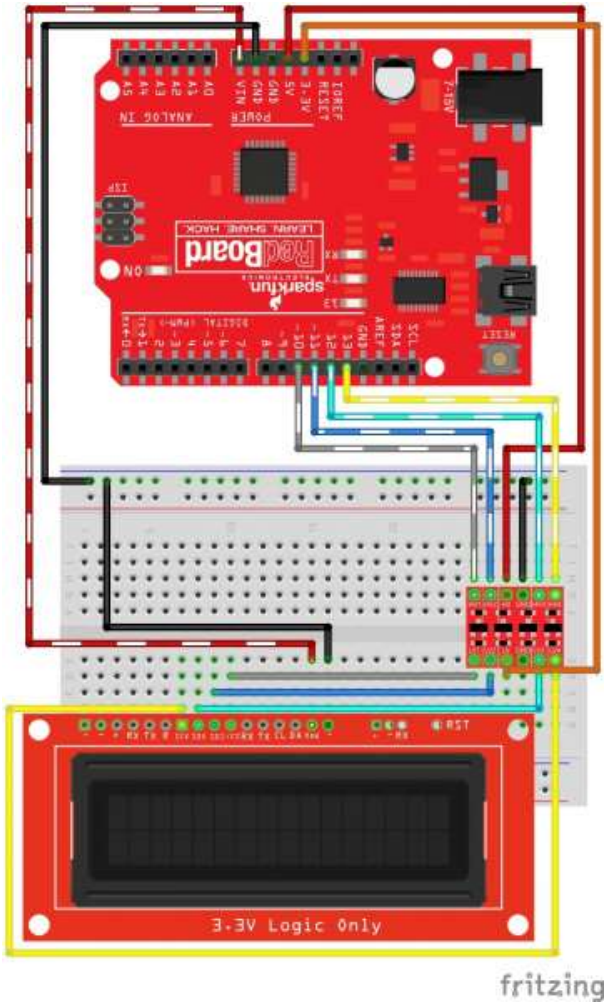
```
/*
 OpenLCD is an LCD with Serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).
 This is example code that shows how to send data over I2C to the display.
 Note: This code expects the display to be listening at the default I2C address. If your display
is not at 0x72, you can
 do a hardware reset. Tie the RX pin to ground and power up OpenLCD. You should see the splash s
creen
 then "System reset Power cycle me" and the backlight will begin to blink. Now power down OpenLC
D and remove
 the RX/GND jumper. OpenLCD is now reset.
 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 SCL (OpenLCD) to A5 (Arduino)
 SDA to A4
 VIN to 5V
 GND to GND
 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.
 For example, to change the baud rate to 115200 send 124 followed by 18.
```

```
*/

#include <Wire.h>

#define DISPLAY_ADDRESS1 0x72 //This is the default address of the OpenLCD

int cycles = 0;

void setup()
{
  Wire.begin(); //Join the bus as master

  //By default .begin() will set I2C SCL to Standard Speed mode of 100kHz
  //Wire.setClock(400000); //Optional - set I2C SCL to High Speed Mode of 400kHz

  Serial.begin(9600); //Start serial communication at 9600 for debug statements
  Serial.println("OpenLCD Example Code");

  //Send the reset command to the display - this forces the cursor to return to the beginning of
the display
  Wire.beginTransmission(DISPLAY_ADDRESS1);
  Wire.write('|'); //Put LCD into setting mode
  Wire.write('-'); //Send clear display command
  Wire.endTransmission();
}

void loop()
{
  cycles++; //Counting cycles! Yay!
  //  Serial.print("Cycle: "); //These serial.print statements take multiple miliseconds
  //  Serial.println(cycles);

  i2cSendValue(cycles); //Send the four characters to the display

  delay(50); //The maximum update rate of OpenLCD is about 100Hz (10ms). A smaller delay will ca
use flicker
}

//Given a number, i2cSendValue chops up an integer into four values and sends them out over I2C
void i2cSendValue(int value)
{
  Wire.beginTransmission(DISPLAY_ADDRESS1); // transmit to device #1

  Wire.write('|'); //Put LCD into setting mode
  Wire.write('-'); //Send clear display command

  Wire.print("Cycles: ");
  Wire.print(value);

  Wire.endTransmission(); //Stop I2C transmission
}
```

You may notice that this is very similar to the example above, Serial Basic. Well, that's because it is doing the exact same thing but instead of Serial UART communication, it is sending the commands over I^2C. If you've got it wired up correctly and the example code running, then you should see the "Hello World Counter:XX" displaying in your LCD screen.

SPI: Hardware Hookup & Example Code - Basic

Here's how to wire up your Redboard to talk SPI to your LCD screen. Remember, convert those logic levels to 3.3Vs! Also note, you could choose a different output pin for the csPin, but in this example we are using D10.



With your hardware now hooked up, the following code is the SPI basic example - it simply writes some characters to the screen over SPI. It has a counter that will increment on each cycle of your main loop. It clears the screen at the top of each loop, so you simply see "Cycles: 1", "Cycles: 2" and so on.

You can get the latest example code from the GitHub repo or you can copy and paste the following code into your Arduino IDE:

```
/*
 OpenLCD is an LCD with Serial/I2C/SPI interfaces.
 By: Nathan Seidle
 SparkFun Electronics
 Date: April 19th, 2015
 License: This code is public domain but you buy me a beer if you use this and we meet someday
 (Beerware license).

 This is example code that shows how to send data over SPI to the display.

 To get this code to work, attached an OpenLCD to an Arduino Uno using the following pins:
 CS (OpenLCD) to 10 (Arduino)
 SDI to 11
 SDO to 12 (optional)
 SCK to 13
 VIN to 5V
 GND to GND

 Command cheat sheet:
 ASCII / DEC / HEX
 '|'    / 124 / 0x7C - Put into setting mode
 Ctrl+c / 3 / 0x03 - Change width to 20
 Ctrl+d / 4 / 0x04 - Change width to 16
 Ctrl+e / 5 / 0x05 - Change lines to 4
 Ctrl+f / 6 / 0x06 - Change lines to 2
 Ctrl+g / 7 / 0x07 - Change lines to 1
 Ctrl+h / 8 / 0x08 - Software reset of the system
 Ctrl+i / 9 / 0x09 - Enable/disable splash screen
 Ctrl+j / 10 / 0x0A - Save currently displayed text as splash
 Ctrl+k / 11 / 0x0B - Change baud to 2400bps
 Ctrl+l / 12 / 0x0C - Change baud to 4800bps
 Ctrl+m / 13 / 0x0D - Change baud to 9600bps
 Ctrl+n / 14 / 0x0E - Change baud to 14400bps
 Ctrl+o / 15 / 0x0F - Change baud to 19200bps
 Ctrl+p / 16 / 0x10 - Change baud to 38400bps
 Ctrl+q / 17 / 0x11 - Change baud to 57600bps
 Ctrl+r / 18 / 0x12 - Change baud to 115200bps
 Ctrl+s / 19 / 0x13 - Change baud to 230400bps
 Ctrl+t / 20 / 0x14 - Change baud to 460800bps
 Ctrl+u / 21 / 0x15 - Change baud to 921600bps
 Ctrl+v / 22 / 0x16 - Change baud to 1000000bps
 Ctrl+w / 23 / 0x17 - Change baud to 1200bps
 Ctrl+x / 24 / 0x18 - Change the contrast. Follow Ctrl+x with number 0 to 255. 120 is default.
 Ctrl+y / 25 / 0x19 - Change the TWI address. Follow Ctrl+x with number 0 to 255. 114 (0x72) is
 default.
 Ctrl+z / 26 / 0x1A - Enable/disable ignore RX pin on startup (ignore emergency reset)
 '-'    / 45 / 0x2D - Clear display. Move cursor to home position.
        / 128-157 / 0x80-0x9D - Set the primary backlight brightness. 128 = Off, 157 = 100%.
        / 158-187 / 0x9E-0xBB - Set the green backlight brightness. 158 = Off, 187 = 100%.
        / 188-217 / 0xBC-0xD9 - Set the blue backlight brightness. 188 = Off, 217 = 100%.

 For example, to change the baud rate to 115200 send 124 followed by 18.
*/
```

```
#include <SPI.h>

int csPin = 10; //You can use any output pin but for this example we use 10

int cycles = 0;

void setup()
{
  pinMode(csPin, OUTPUT);
  digitalWrite(csPin, HIGH); //By default, don't be selecting OpenLCD

  SPI.begin(); //Start SPI communication
  //SPI.beginTransaction(SPISettings(100000, MSBFIRST, SPI_MODE0));
  SPI.setClockDivider(SPI_CLOCK_DIV128); //Slow down the master a bit
}

void loop()
{
  cycles++; //Counting cycles! Yay!

  //Send the clear display command to the display - this forces the cursor to return to the begi
nning of the display
  digitalWrite(csPin, LOW); //Drive the CS pin low to select OpenLCD
  SPI.transfer('|'); //Put LCD into setting mode
  SPI.transfer('-'); //Send clear display command
  digitalWrite(csPin, HIGH); //Release the CS pin to de-select OpenLCD

  char tempString[50]; //Needs to be large enough to hold the entire string with up to 5 digits
  sprintf(tempString, "Cycles: %d", cycles);
  spiSendString(tempString);

  //25ms works well
  //15ms slight flickering
  //5ms causes flickering
  delay(250);
}

//Sends a string over SPI
void spiSendString(char* data)
{
  digitalWrite(csPin, LOW); //Drive the CS pin low to select OpenLCD
  for(byte x = 0 ; data[x] != '\0' ; x++) //Send chars until we hit the end of the string
    SPI.transfer(data[x]);
  digitalWrite(csPin, HIGH); //Release the CS pin to de-select OpenLCD
}
```

Troubleshooting

Random Character

If the display is powered up without the RX line connected to anything, the display may fill with strange characters. This is because the display is receiving random noise on the disconnected line. If you connect the RX line to a true TX port, this will not happen.

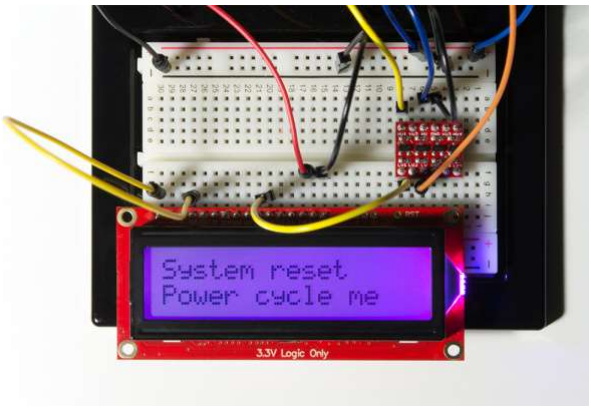
Faded Characters on Display

If the display is unreadable or washed out, the contrast may need to be adjusted. This is done in software, so you will need to send your display some contrast control commands via Serial UART, I^2C or SPI. There is a specific example for each of these communication types inside the github repository here:

- Example2_Serial_Contrast
- Example2_I2C_Contrast
- You can also follow along with the example in this tutorial above: click here.

Emergency Reset

If your LCD screen has entered an unknown state, or you are unable to communicate with it, it's probably a good idea to try resetting everything back to default settings. The OpenLCD firmware has a built-in "emergency reset" feature. When the screen first boots up, the AVR on the back will watch its RX pin. If that pin is held LOW (aka tied to ground), for 2 seconds, then it will reset all settings to default. Most importantly, your baud rate will be set back to 9600. After the reset is complete, the screen will display the message "System Reset Power Cycle Me", and flicker the backlight on and off repeatedly until you cycle power.



To perform an emergency reset, please be sure to follow these exact steps in this order:

- Ensure your screen is OFF.
- Tie RX to GND.
- Power your screen.
- Wait 2 seconds. Verify "System Reset" message.
- Remove connection from RX to GND.
- Cycle Power.

Now, please enjoy your default settings (including 9600 baud).

Firmware Update

Heads up! Remember, the name "**SerLCD**" will refer to the hardware throughout the tutorial. **OpenLCD** will refer to the firmware used on the SerLCD. The board definition (i.e. "**SparkFun SerLCD**") refers to the board type flashed on the ATmega328P.

To update the firmware on your LCD, you can use an FTDI Basic 3.3V - beefy model and the Arduino IDE software. The AVR on the back of your LCD actually has an Arduino-compatible bootloader. That said, it is slightly custom in that it was compiled for the 11.0592 crystal. This means you will need to install the *SparkFun AVR Boards*, and then select "**Sparkfun SerLCD**" as your board type.

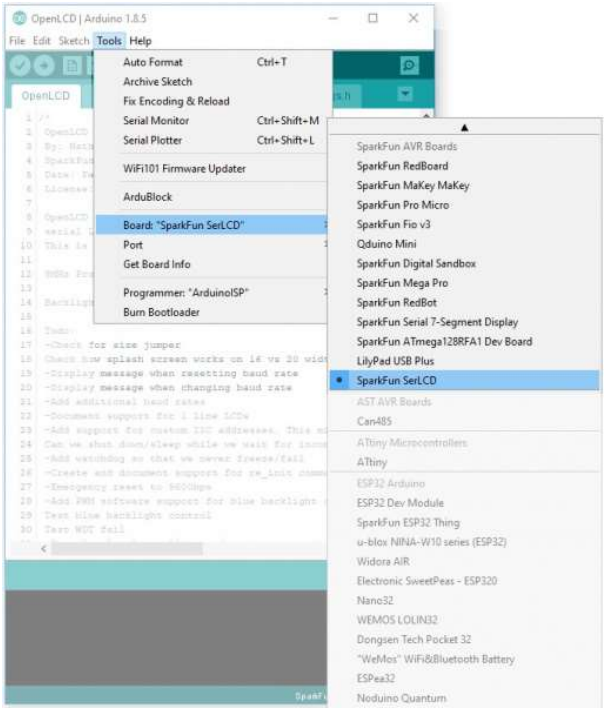
Install the Board Package

> **Note:** The firmware has been tested with Arduino IDE v1.8.5.

If you've made it this far, presumably you have the latest version of the Arduino IDE on your desktop. If you do not, please review our tutorial on installing the Arduino IDE for board add-ons. Here's the tutorial for installing custom board packages:

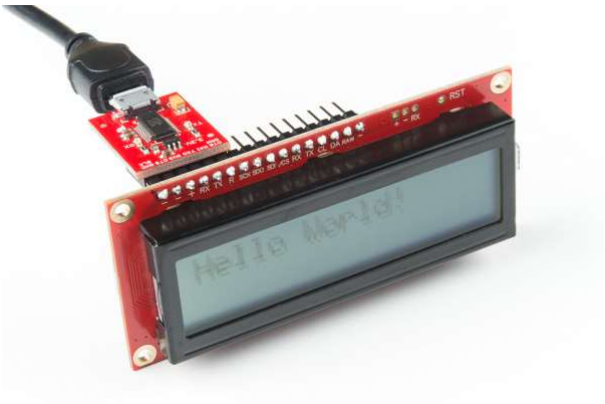
INSTALLING CUSTOM BOARD PACKAGES

Once you have that installed, you should see "**SparkFun SerLCD**" as a board option from the drop down menu here:

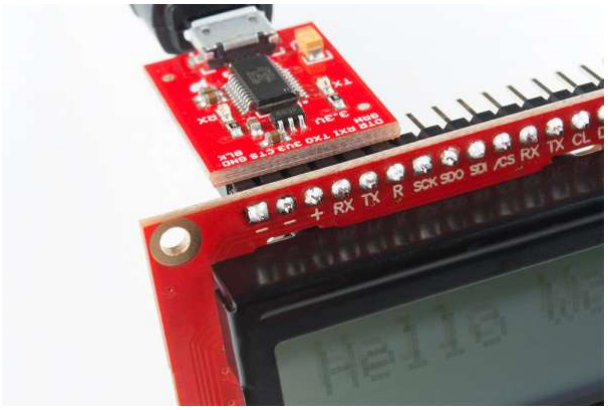


Hardware Hookup

Now plug in your serial FTDI basic into your LCD screen.



Make sure to line it up properly so the "-" is lined up withe the "GND", and the "R" is lined up with "DTR".



Note: For advanced users, you can also flash the hex file by connecting an AVR programmer if you are still having issues uploading through the Arduino IDE. Just make sure to power the SerLCD if the AVR programmer does not provide power for your target and the logic levels are 3.3V.

AVR Programmer Pins	SerLCD Pins	External Power Source for Target (assuming the AVR Programmer is not able to power the target)
	RAW	5V
Vcc	Vcc	
GND	GND	GND
COPI	SDI	
CIPO	SDO	
SCK	SCK	
RESET	RST	

Once connected, grab the *.hex file from the GitHub repo with the Arduino bootloader to re-flash the ATmega328P with AVR Studio or command line. Once the bootloader is re-installed, try following the directions below to install the latest firmware version on the SerLCD. At the time of writing this note, the *.hex file used firmware v1.2.

Default Firmware

Download and unzip the latest firmware located in the OpenLCD GitHub repository:

OPENLCD DEFAULT FIRMWARE

Not that you will need to include all of the files in the "**... > OpenLCD-master > firmware > OpenLCD**" directory. When you open the *OpenLCD.ino* sketch in arduino, the other necessary files will open as tabs.

- *OpenLCD.ino*
- *Setting_Control.ino*

- *System_Functions.ino*
- *settings.h*

SoftPWM and LiquidCrystalFast Arduino Libraries

You will also need to install the SoftPWM and LiquidCrystalFast libraries. You will need to download and manually install the libraries the following libraries before you are able to upload the default firmware. To use these libraries, you can add the library from the Arduino IDE by selecting **Sketch > Include Library > Add .ZIP Library…** and selecting the **.zip** file from wherever you store your file downloads.

SOFTPWM LIBRARY (ZIP)

LIQUIDCRYSTALFAST LIBRARY (ZIP)

Uploading Default Firmware

Once the board add-on and associated libraries are installed, click the **UPLOAD** button in the IDE (the right facing arrow) to get the latest code onto your LCD!

> **Note:** Any of the settings stored in EEPROM memory (like baud, splash screen contents, backlight settings, etc.) will **not** be overwritten. If you are using a fresh IC (or you erased your EEPROM), then all of the defaults will return. To reset these settings to default, you must perform an "**Emergency Reset**". The instructions for this are in another part of this troubleshooting section.

Using the Serial Enabled LCD on an Atmega32U4's Hardware UART

If you are using the serial enabled LCD with an Atmega32U4-based Arduino (like a Pro Micro, Arduino Leonardo, Arduino LilyPad USB, etc), you might need to add a small delay in the setup before you can get it working with the hardware UART (pins 0 and 1). Here's an example:

```
///test example using ATmega32U4's hardware UART and delay
void setup() {
  delay(2000);//add delay so the ATmega32U4 can have a second before sending serial data to the
 LCD
  Serial1.begin(9600);//set up the hardware UART baud
}

void loop() {
  Serial1.print("print something");//send something to the serial enabled LCD
  delay(50);
}
```

Software Serial for Arduino Due

Unfortunately, you are not able to use the serial enabled LCDs with an Arduino Due due the differences in how change interrupts are used for the ARM processor. The software serial library is not included in the Arduino Due's tree:

ARDUINO.CC FORUMS - SOFTWARE SERIAL FOR ARDUINO DUE?

Try using the other hardware serial UARTs that are not connected to the Arduino Due's programming pins for uploading. Make sure to adjust the code for the hardware serial UART.

Resources and Going Further

Now that you've successfully got your OpenLCD up and running, it's time to incorporate it into your own project! When it is complete (or even during the design and build phases) please share in comments section of this tutorial, we'd love to hear about it! We also like doing project highlights, so please don't hesitate to reach out when it's finished. Maybe we could even feature your project with a blog post and video!

Also, if you ran into any issues during this hookup guide, or something wasn't crystal clear the first time you read it, please let us know in the comments section of this tutorial. We strive to make the best documentation possible, and really want to hear about any pain points you discovered. Thanks in advance!

For more information, check out the resources below:

- OpenLCD GitHub Repo - OpenLCD design files, default firmware, and example code.
  - SerLCD Arduino Library
  - Qwiic_SerLCD_Py Python Module
  - SoftPWM Arduino Library
  - LiquidCrystalFast Arduino Library
- HD44780
  - Datasheet
  - LCD User-Defined Graphics - If you would like to create a custom character, you would need to send a command byte before controlling the individual pixels in the character square.

Need some inspiration for your next project? Check out some of these related tutorials:

Using OpenSegment

How to hook up and use the OpenSegment display shield. The OpenSegment is the big brother to the Serial 7-Segment Display. They run on the same firmware, however the OpenSegment is about twice as big.

Blynk Board Project Guide

A series of Blynk projects you can set up on the Blynk Board without ever re-programming it.

How to Run a Raspberry Pi Program on Startup

In this tutorial, we look at various methods for running a script or program automatically whenever your Raspberry Pi (or other Linux computer) boots up.

TFT LCD Breakout 1.8in 128x160 Hookup Guide

This TFT LCD Breakout is a versatile, colorful, and easy way to experiment with graphics or create a user interface for your project.