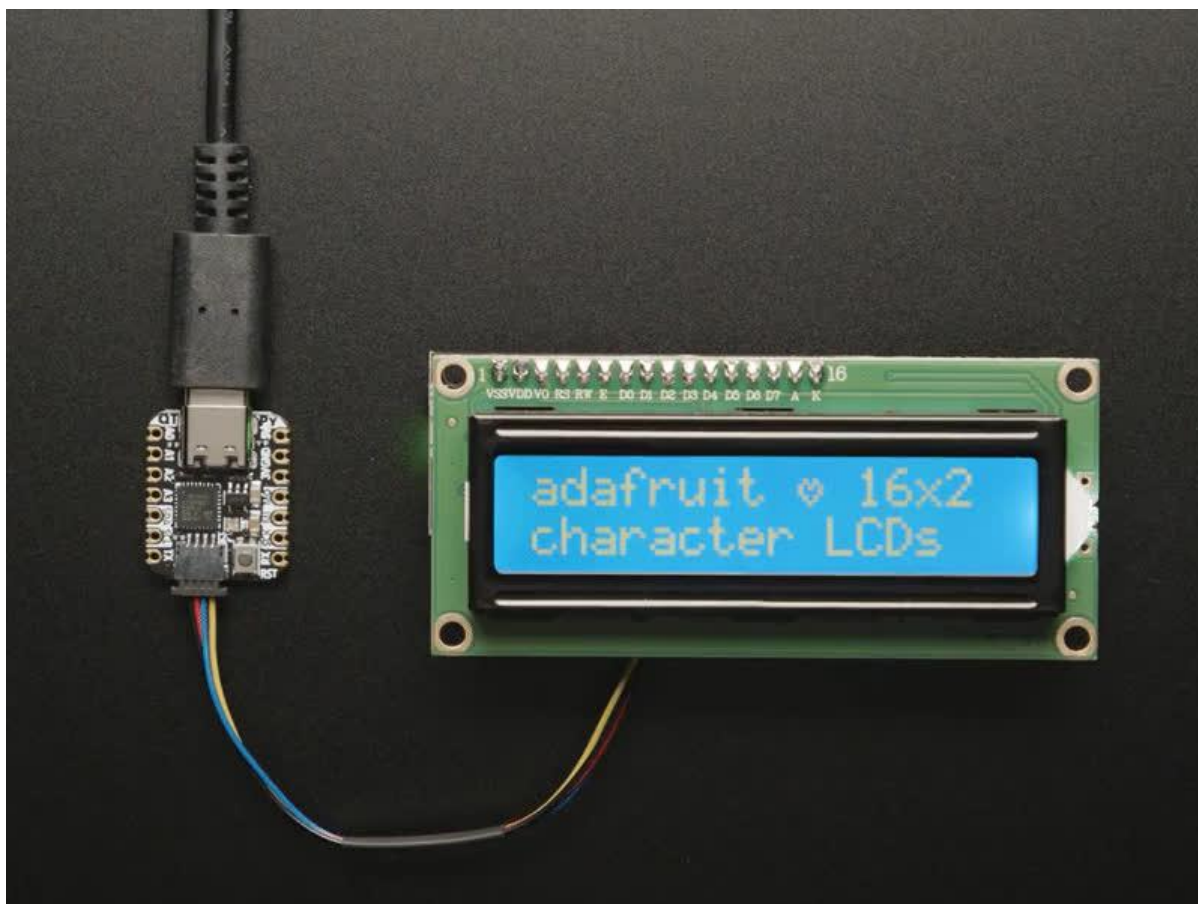




I2C/SPI LCD Backpack

Created by lady ada



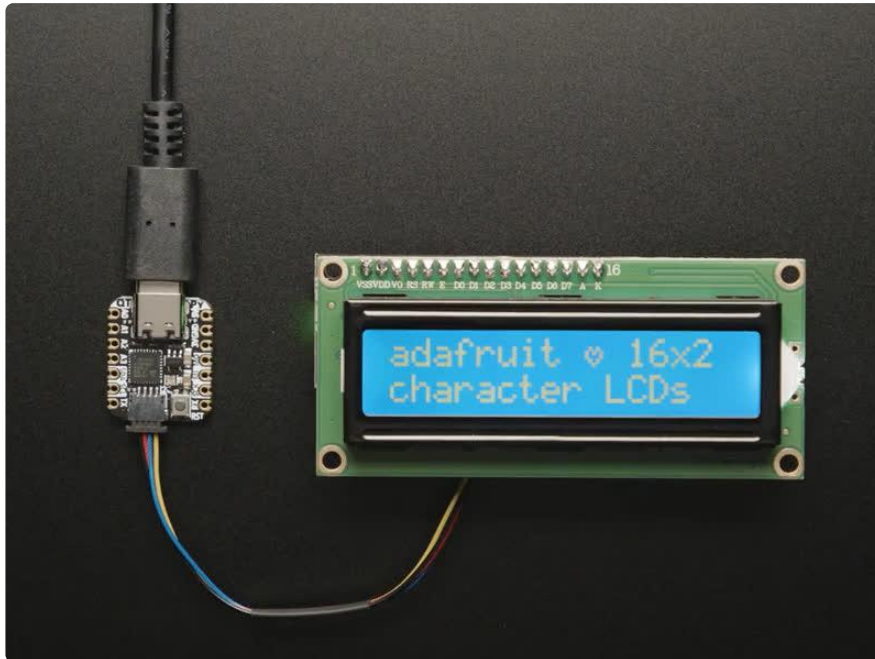
<https://learn.adafruit.com/i2c-spi-lcd-backpack>

Last updated on 2023-02-24 09:20:34 AM EST

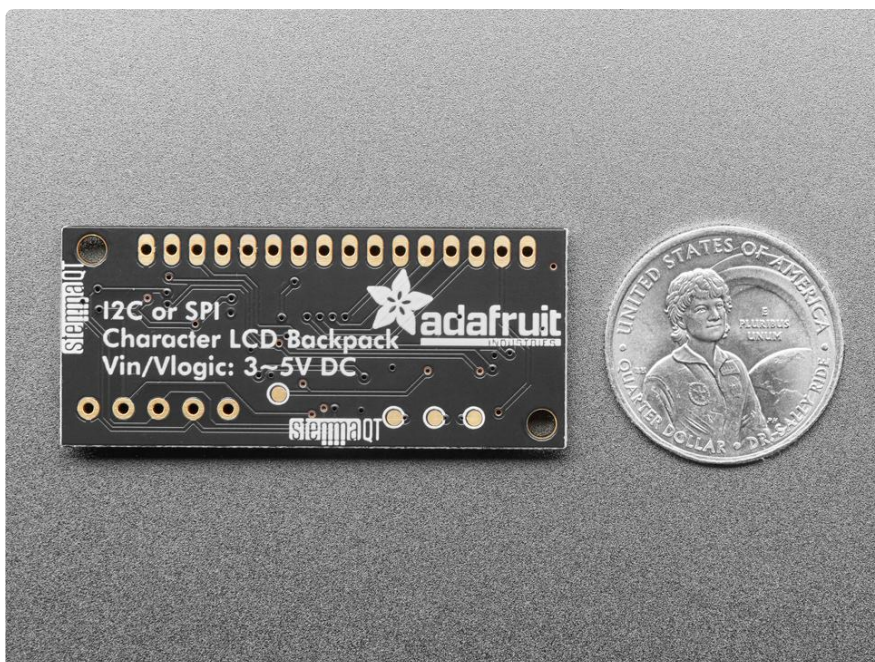
Table of Contents

Overview	3
Pinouts	6
<ul style="list-style-type: none">• Power Pins• I/O Pins• I2C Logic Pins• SPI Enable Jumper• SPI Pins• Address Jumpers• 3-5V Jumper	
Which LCD to Use?	9
<ul style="list-style-type: none">• Wait - the backpack has 16 holes, but my LCD only has 14 pins!	
Assembly	9
<ul style="list-style-type: none">• Parts Check• Terminal Blocks• Prepare LCD• Attach LCD• My LCD only has 14 pins. Which holes do I use?	
Arduino I2C Use	14
<ul style="list-style-type: none">• Install Adafruit_LiquidCrystal• Load Demo• Changing the I2C Address	
Arduino SPI Use	19
<ul style="list-style-type: none">• Install Adafruit_LiquidCrystal• Load Demo	
Python & CircuitPython	23
<ul style="list-style-type: none">• CircuitPython Microcontroller Wiring• Python Computer Wiring• CircuitPython Installation of CharLCD Library• Python Installation of CharLCD Library• Python & CircuitPython Usage• SPI Initialization• Backpack Usage• Full Example Code	
Python Docs	31
Downloads	31
<ul style="list-style-type: none">• STEMMA QT Schematic and Fab Print• Original Schematic and Fab Print	

Overview

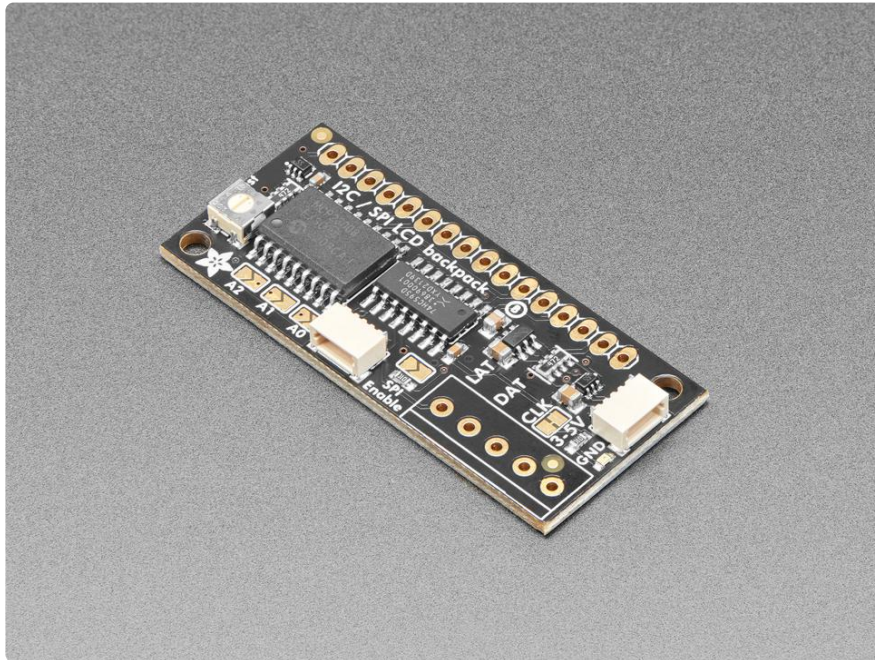


Character LCDs are a fun and easy way to have your microcontroller project talk back to you. They are also common, and easy to get, available in tons of colors and sizes. [We've written tutorials on using character LCDs with an Arduino \(\)](#) (or similar microcontroller) but find that the number of pins necessary to control the LCD can be restrictive, especially with ambitious projects. We wanted to make a 'backpack' (add-on circuit) that would reduce the number of pins without a lot of expense.

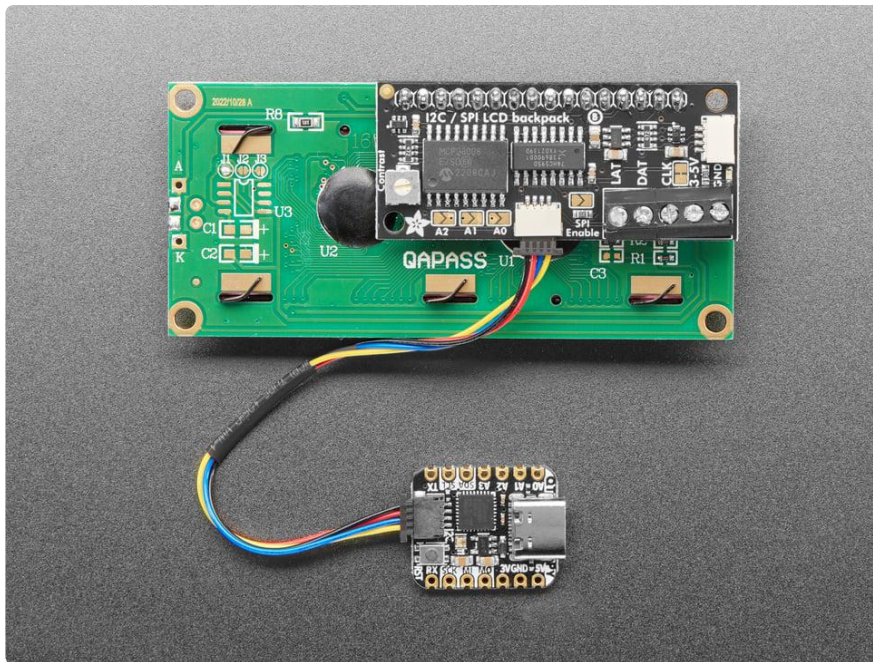


By using simple I2C and SPI input/output expanders we have reduced the number of pins, while still making it easy to interface with the LCD. Only 2 pins are needed for I2C, 3 for SPI. For Arduino and CircuitPython/Python users, we provide an easy-to-use library that is backwards compatible with projects using the '6 pin' wiring.

This backpack comes with a 2-pin and 3-pin terminal block as shown (you can snap it together to make a 5-pin terminal and then solder it to the backpack for easy wiring).



This backpack will work with any 'standard' character LCD, from 8x1 to 20x4 sizes! As long as they have a 16-pin single-line connection header at the top. [We carry a few LCDs that work great \(\)](#). We suggest using our blue white 20x4 or 16x2 LCDs. Note that it does not work with 16x2 OLED displays. You can try to connect our RGB 16x2 or 20x4 LCDs, but this backpack will not control the RGB backlight so you'll have to use the backpack only for the 14 digital IO pins (pins #1-14) and connect the backlight pins (#15-#18) directly to your microcontroller with 4 extra wires for color/PWM control as if they were just an RGB LED.



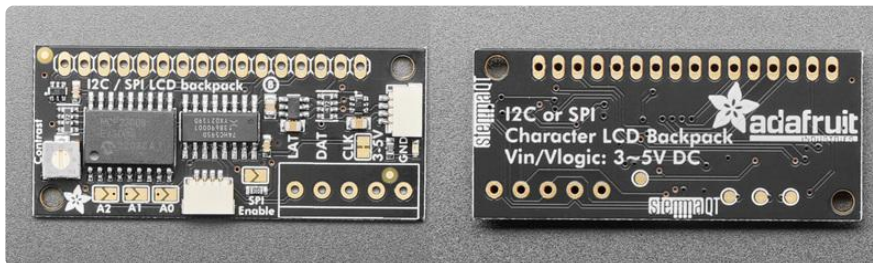
NEW! As of February 8, 2023 - This backpack now comes with a big re-spin that makes lots of improvements:

- We've added a 3-5V boost circuit so you can use this backpack to control 5V LCDs even with 3V power and logic.
- The contrast potentiometer is a lot nicer and easier to twist using a small screwdriver
- Added [SparkFun qwiic \(\)](#) compatible [STEMMA QT \(\)](#) connectors for the I2C bus so you don't even need to solder the I2C and power lines. Just wire up to your favorite micro using a [STEMMA QT adapter cable. \(\)](#) [QT Cable is not included, but we have a variety in the shop \(\)](#).
- Functionality and size/shape are the same - mechanically and code-wise it is a drop-in replacement.
- We've also updated this PCB with [Adafruit Pinguin \(\)](#) to make a lovely and legible silkscreen.

There are two versions of this board - the STEMMA QT version shown above, and the original header-only version shown below. Code works the same on both!



Pinouts



The default I2C address is 0x20.

Power Pins

- 3-5V (VIN) - You can power the I2C/SPI LCD Backpack with 3 to 5V DC. There is a 3-5V boost circuit onboard so you can use this backpack to control 5V LCDs even with 3V power and logic.
- GND- Common ground for data and power.

I/O Pins

Located along the top of the board, above the board label on the silk, are the input and output pins. These pins are for attaching an LCD screen to the backpack and are numbered 1-16. Pin 1 is located furthest to the left when looking at the front of the board.

Advanced users can repurpose these pins for general purpose I/O expansion. The MCP23008 has 8 I/O pins (7 are connected, backpack pins 4, 6, 11, 12, 13, 14 and 15) with optional pull-ups and the SPI 74HC595 has 7 connected outputs (backpack pins 4, 6, 11, 12, 13, 14 and 16).

I2C Logic Pins

- CLK (SCL) - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- DAT (SDA) - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- [STEMMA QT \(\)](#) - These connectors allow you to connect to dev boards with STEMMA QT connectors or to other things with [various associated accessories \(\)](#).

SPI Enable Jumper

- SPI Enable - The SPI Enable Jumper is located on the back of the board, between the lower STEMMA QT connector and the terminal block pins. You can solder this jumper closed to control the backpack over SPI.

SPI Pins

The I2C/SPI LCD Backpack can be controlled over SPI after soldering the SPI Enable jumper closed.

- CLK (SCK) - SPI clock input pin
- DAT (MOSI) - This is the SPI MOSI (Microcontroller Out / Serial In) pin. It is used to send data from the microcontroller to the backpack.
- LAT - This is the latch input, which transfers the received data to the output lines on the shift register for the LCD.

Address Jumpers

On the back of the board are three address jumpers, labeled A0, A1, and A2, between the Adafruit logo on the board silk and the lower STEMMA QT port. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

The default I2C address is 0x20. The other address options can be calculated by adding the A0/A1/A2 to the base of 0x20.

A0 sets the lowest bit with a value of 1, A1 sets the next bit with a value of 2 and A2 sets the next bit with a value of 4. The final address is $0x20 + A2 + A1 + A0$ which would be 0x27.

So for example if A2 is soldered closed and A0 is soldered closed, the address is $0x20 + 4 + 1 = 0x25$.

If only A0 is soldered closed, the address is $0x20 + 1 = 0x21$

If only A1 is soldered closed, the address is $0x20 + 2 = 0x22$

If only A2 is soldered closed, the address is $0x20 + 4 = 0x24$

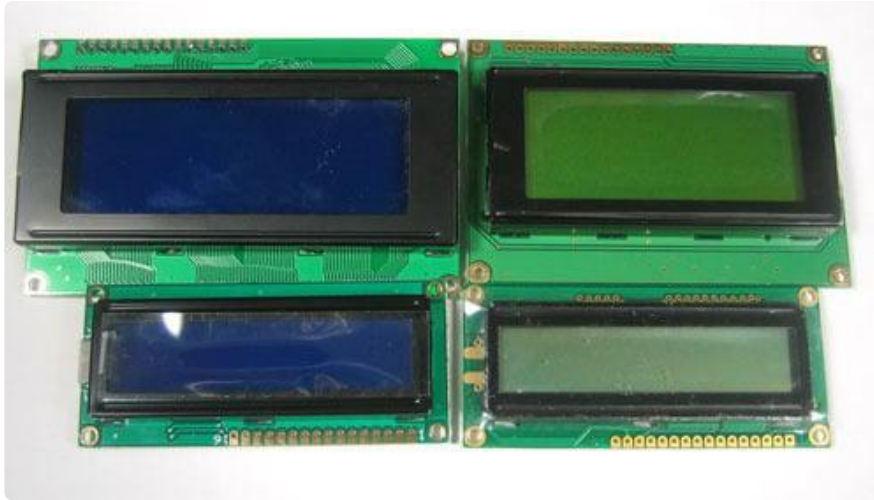
The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

ADDR	A0	A1	A2	ADDR	A0	A1	A2
0x20	L	L	L	0x24	L	L	H
0x21	H	L	L	0x25	H	L	H
0x22	L	H	L	0x26	L	H	H
0x23	H	H	L	0x27	H	H	H

3-5V Jumper

- 3-5V - The 3-5V jumper is located on the back of the board between the CLK and 3-5V labels on the silk. If you cut this jumper, it will disconnect the 5V boost circuit from VCC, causing the backpack to output 3V to a connected LCD.

Which LCD to Use?



This backpack will work with any 'standard'/'classic' character LCD. It does not work with graphic LCDs. Character LCDs come in sizes ranging from 8x1 (8 characters, one line) to 40x4 (40 characters, four lines). The backpack will also only fit LCDs that have a single line of pins at the top, not the ones that have a 2x10 or 2x8 connector on the side. Those are much rarer these days but just keep a look out for that!

The backpack will work with RGB LCDs (but won't control the RGB backlight, you can do that separately from the LCD control) and it won't work with 40x4 LCDs because they have a second Enable pin.

Wait - the backpack has 16 holes, but my LCD only has 14 pins!

As long as your LCD has a HD44780 driver chip, you're OK. The extra two pins are for the backlight. Your LCD doesn't have a backlight.

We'll show you which holes to ignore in the Assembly section of this guide.

Assembly

Putting together the backpack onto an LCD is a quick process, and should take only a few minutes with a soldering iron.

Parts Check

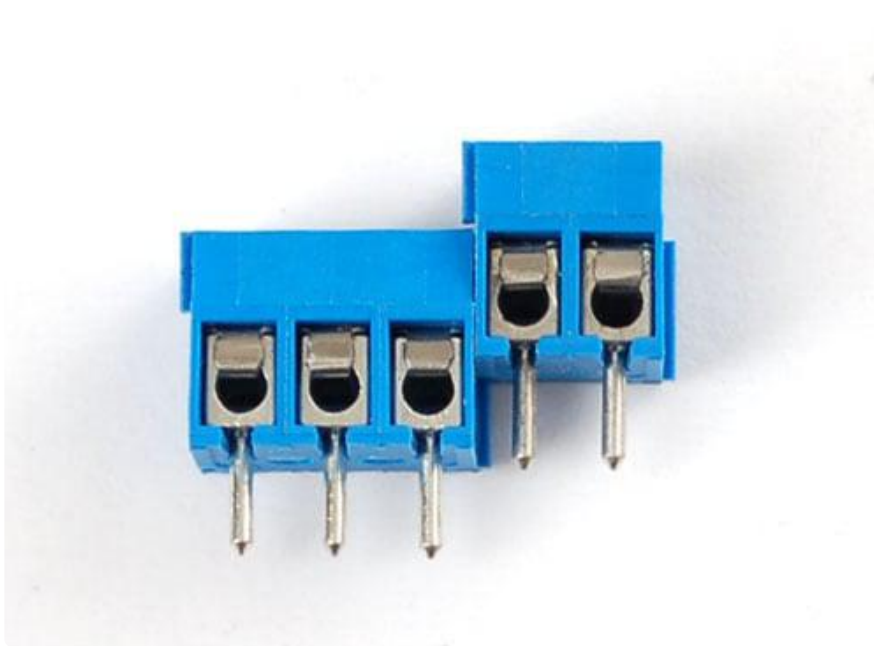
Verify you have everything in the bag, there should be an assembled and tested PCB, a 2-pin and 3-pin 3.5mm terminal block. The backpack does not come with header or

an LCD.

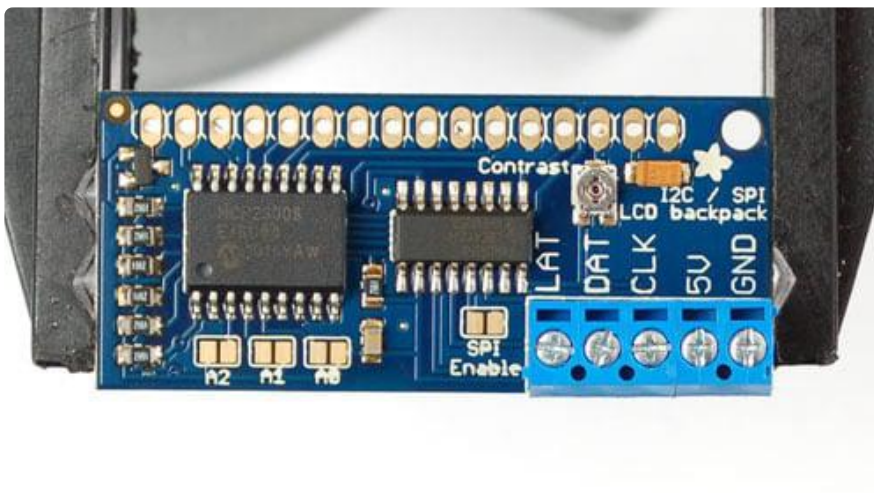
Terminal Blocks

The [terminal blocks](#) () allow you to easily attach and remove the LCD from your wiring, which we think is awfully handy. If you dont want the terminal blocks (they stick out a bit) you can always skip this step.

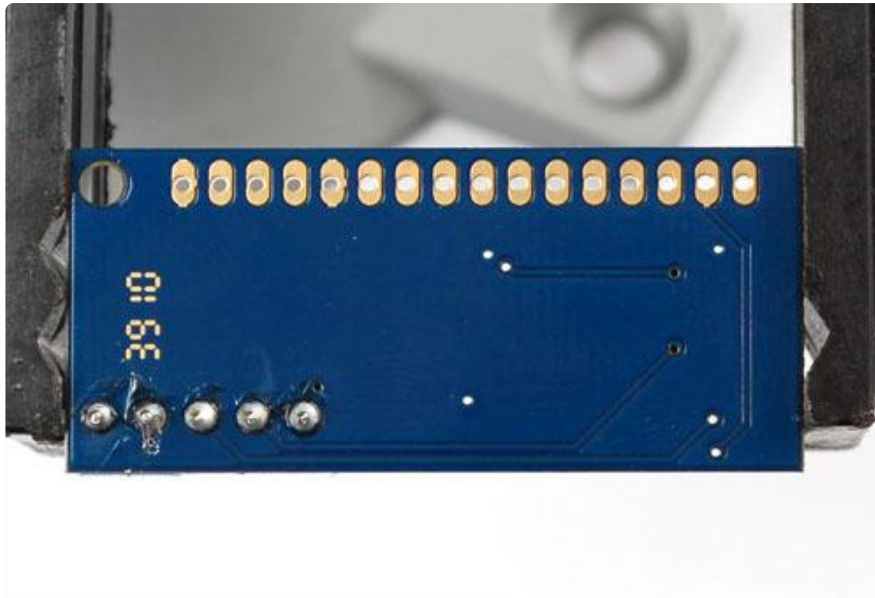
[The terminal blocks come in 2 and 3-pin pieces](#) (). Slide them together.



Place the blocks over the corner area of the backpack so that the holes stick out (unless for some reason you want them to face the other way)



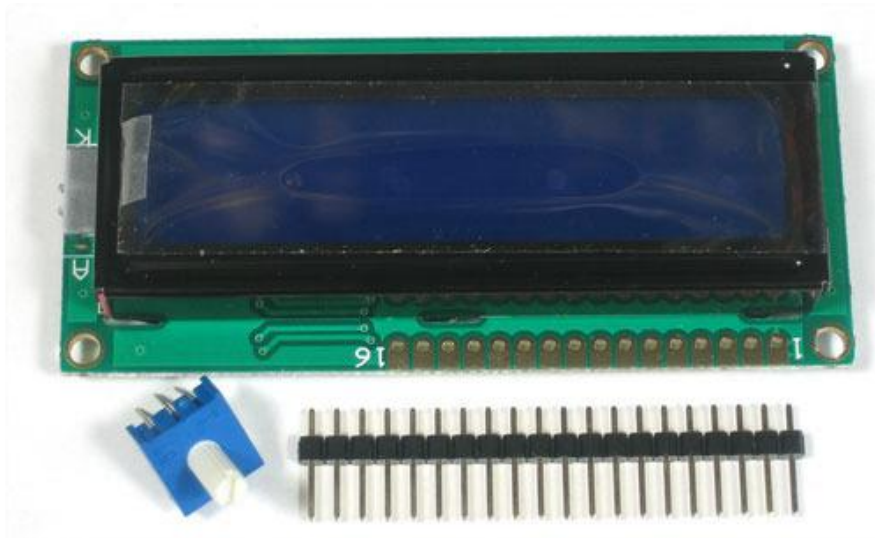
Solder them into place



Prepare LCD

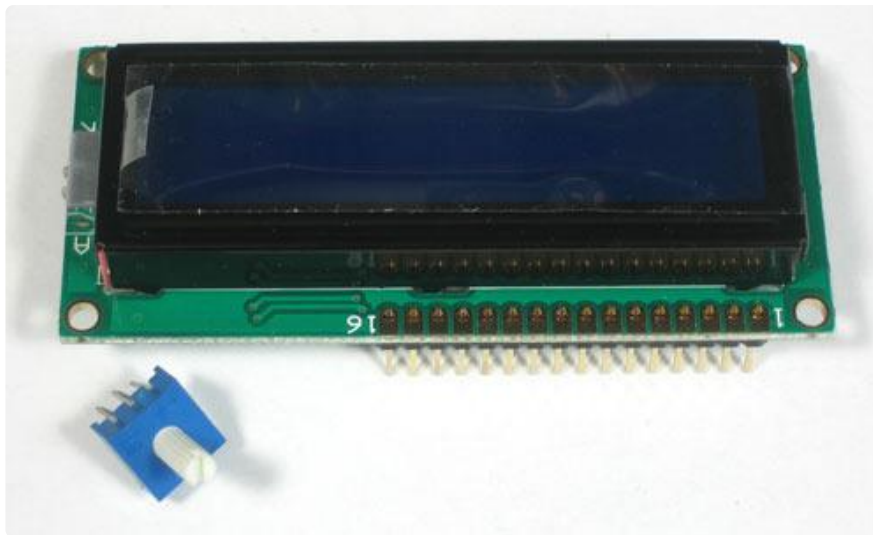
Next we will attach the backpack to the LCD. First we must put header onto the LCD, if you bought the LCD from us, it will come with a stick of header. Otherwise, pick up some [standard 0.1" male header](#) ()

These photos shows a 10K potentiometer, you can ignore it

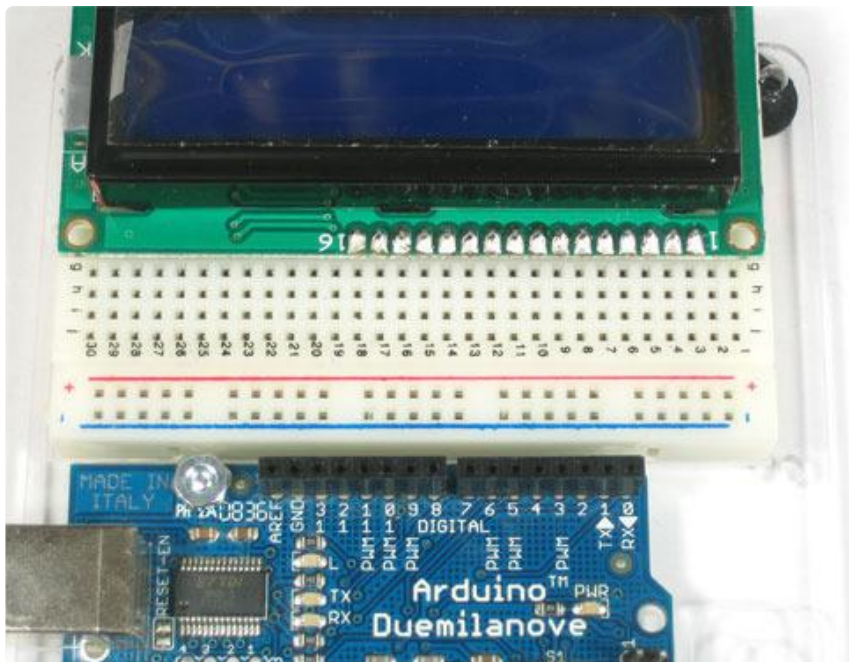


If the header is too long, just cut/snap it short so that it is 16 pins

Next you'll need to solder the header to the LCD. You must do this, it is not OK to just try to 'press fit' the LCD!



The easiest way we know of doing this is sticking the header into a breadboard and then sitting the LCD on top while soldering. this keeps it steady.



Attach LCD

Now we will attach the backpack. We will show how to do this in a permanent fashion. If you think you would like to remove and replace the LCD at some time, you can use [a piece of 16-pin long 0.1" female header](#) () as a socket but be aware it will stick out a lot.

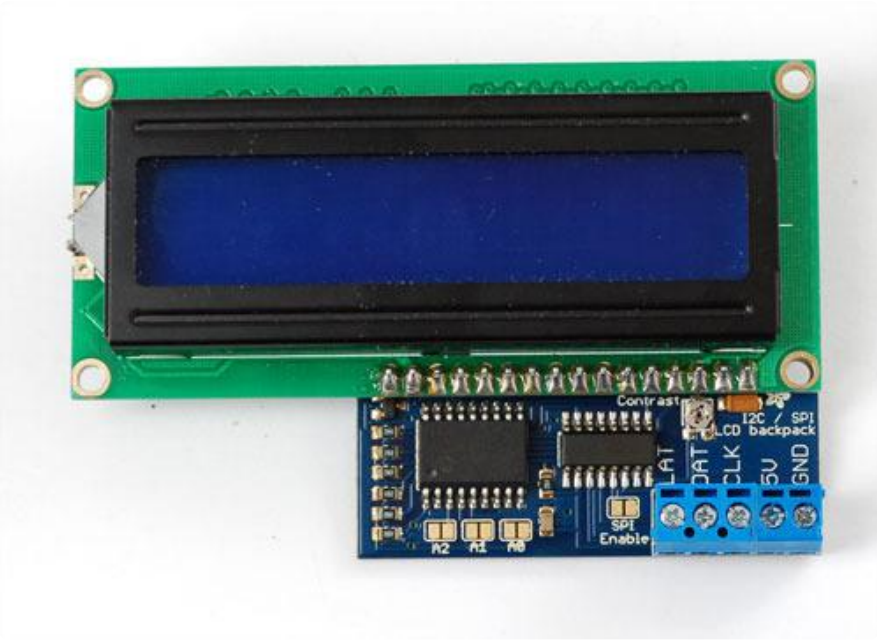
There are two options, you can tuck the backpack behind the LCD

Make sure that you line up pin 1 of the backpack with pin 1 of the display. Pin 1 is on the right side of the backpack closest to the Adafruit logo. Pin one of the

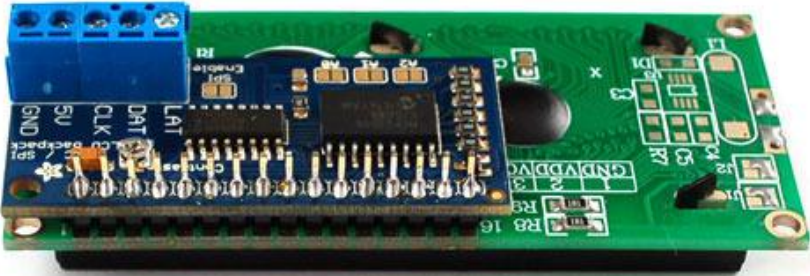
display should be marked. When correctly installed, the blue terminal block will be close to the edge of the display.



Or solder it so it's to the side, this way it's thinner.



Solder the header to the backpack.



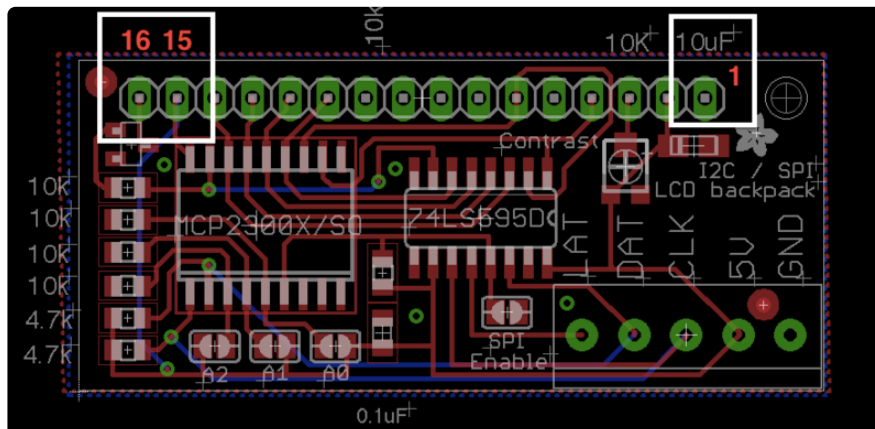
Make sure that as you solder the first pin, the backpack PCB isn't leaning against the LCD, where the terminal blocks could short against some components. You can put some electrical tape or foam tape behind to avoid this if you think it could be an issue.



That's it!

My LCD only has 14 pins. Which holes do I use?

The extra two pins are for the backlight. Your LCD doesn't have a backlight. Here's a picture of the backpack board layout, with pins 15 and 16 located. Just leave those empty. Cut the pin header down to 14 pins instead of 16, and attach your LCD to holes 1-14 in the backpack.



If your LCD has 18 pins, you won't be able to use the extra backlight pins, just connect 1-16 and leave 17 and 18 disconnected

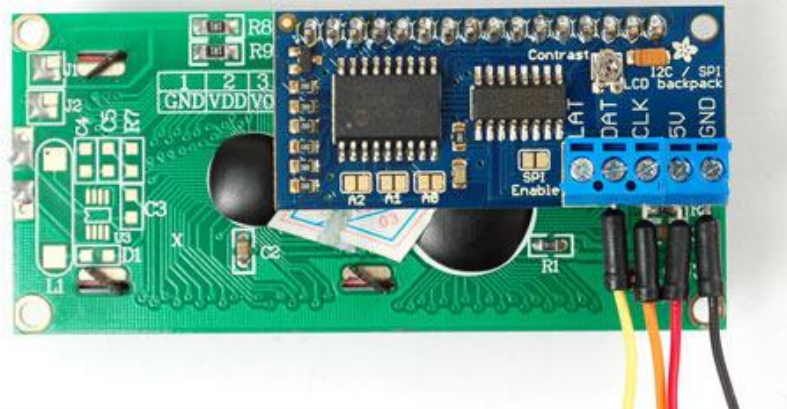
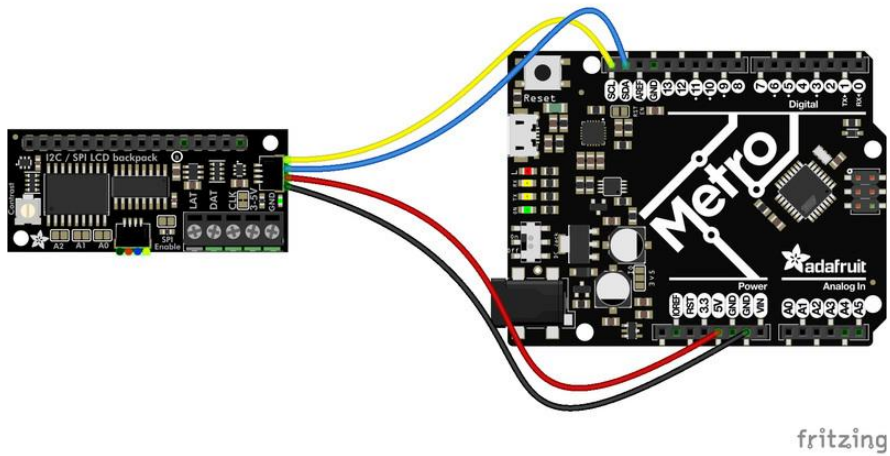
Arduino I2C Use

The first option we'll show is how to use the I2C interface on the backpack. We'll be showing how to connect with an Arduino, for other microcontrollers please see our

MCP23008 library code for the commands to send to the I2C I/O expander. I2C is nice because it only uses two pins, and you can put multiple I2C devices on the same two pins.

So for example, you could have up to 8 LCD backpacks+LCDs all on two pins! The bad news is that you have to use the 'hardware' I2C pins. You can't change those pins and you can't use them for reading analog data. If you absolutely need those two pins, use SPI (see the next section).

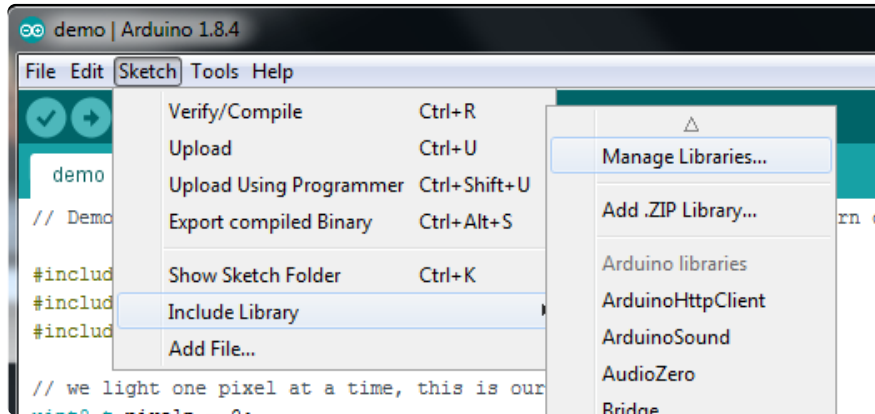
For this, we'll need to connect four wires: GND, 5V, CLK (clock) and DAT (data) via the STEMMA QT connection or the terminal block pins.



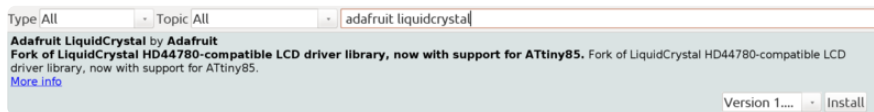
Install Adafruit_LiquidCrystal

To begin reading sensor data, you will need to download the Adafruit_LiquidCrystal library from the Arduino library manager.

Open up the Arduino library manager:



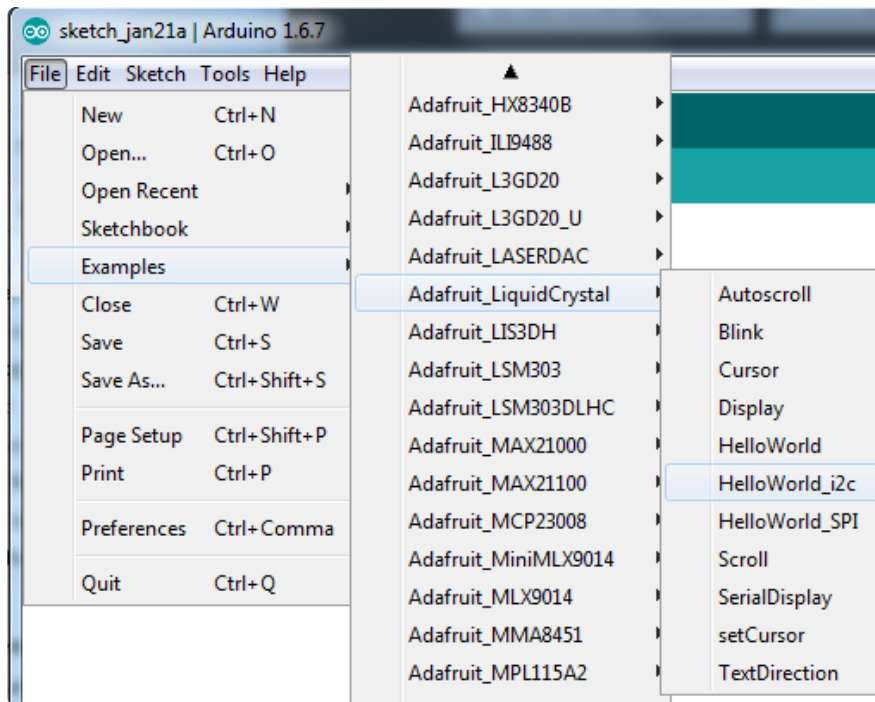
Search for the Adafruit LiquidCrystal library and install it



We also have a great tutorial on Arduino library installation at: <http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use> ()

Load Demo

Restart the IDE and load up the Adafruit_LiquidCrystal->>HelloWorld_i2c demo



Upload the sketch. You should see the backlight turn on when the Arduino resets. If you don't see any characters, adjust the Contrast trim potentiometer with a mini-screwdriver until you see the text clearly

The default HelloWorld sketch blinks the backlight as well as updating the text.

If you see the backlight blinking that means your connection to the I2C port is OK but the contrast is too low or too high, or the LCD data pins are not solidly connected.

Check the contrast first by gently twisting the mini trim potentiometer, if that doesn't help, recheck your soldering and resolder all 16 of the LCD pins!

Once you're done, you can remove the blinking LED backlight code:

```
lcd.setBacklight(HIGH);  
delay(500);  
lcd.setBacklight(LOW);  
delay(500);
```



Changing the I2C Address

If you want to have more than one LCD backpack device each one needs to have a unique 'address'. You can set the address by jumpering the A0 A1 and A2 solder jumpers. By default, no jumpers are soldered, giving an address of 0x20 (offset 0). If you want to have an address of 0x23 (0x20 + offset 3) you would solder A0 (bit 0) and A1 (bit 1) for an address offset of "011" = 3 in binary.

ADDR	A0	A1	A2	ADDR	A0	A1	A2
0x20	L	L	L	0x24	L	L	H
0x21	H	L	L	0x25	H	L	H
0x22	L	H	L	0x26	L	H	H
0x23	H	H	L	0x27	H	H	H

Then, in the code change:

```
// Connect via i2c, default address #0 (A0-A2 not jumpered)
LiquidCrystal lcd(0);
```

to

```
// Connect via i2c, address #3 (A0&A1 jumpered)
LiquidCrystal lcd(3);
```

Arduino SPI Use

Another option for connecting is to use SPI, which is a simpler protocol. The good news about SPI is that its very simple and you can use any 3 pins to connect. You can share the data and clock pins with another device as long as they remain outputs, the latch pin should only be used for the backpack. So if you wanted 3 LCDs, for example, they would all have the same data and clock pins, but the latch pin would be different, for 5 pins total

This library does not use the Hardware SPI library, which means you can use any 3 pins! However, if you are using the hardware SPI port (such as for Ethernet, WiFi, an LCD, etc. etc) you cannot share those pins with this LCD!

The first thing you will need to do is to enable SPI. To do this, solder the SPI Enable solder jumper by heating up the pads with a soldering iron and soldering a blob onto both pins:



This will switch the backpack over to SPI mode instead of I2C. If you want to go back to I2C, use wick or a solder sucker to remove the jumper

Next we will connect 5 wires, 5V, GND, DAT, CLK, and LAT.

- To match the example, CLK goes to to Digital 2
- DAT to Digital 3,
- LAT to Digital 4

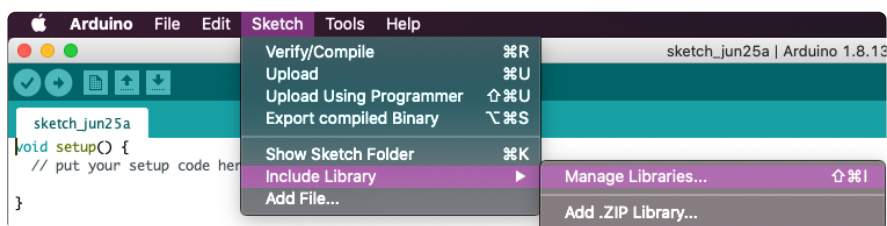
Once we have the example sketch running you can of course change these to anything you'd like

- Connect 5V and GND to the 5v and Ground Arduino power pins. If you are using a 3.3V Arduino, you still need to power the LCD with 5V power! You can use 3.3V logic just fine

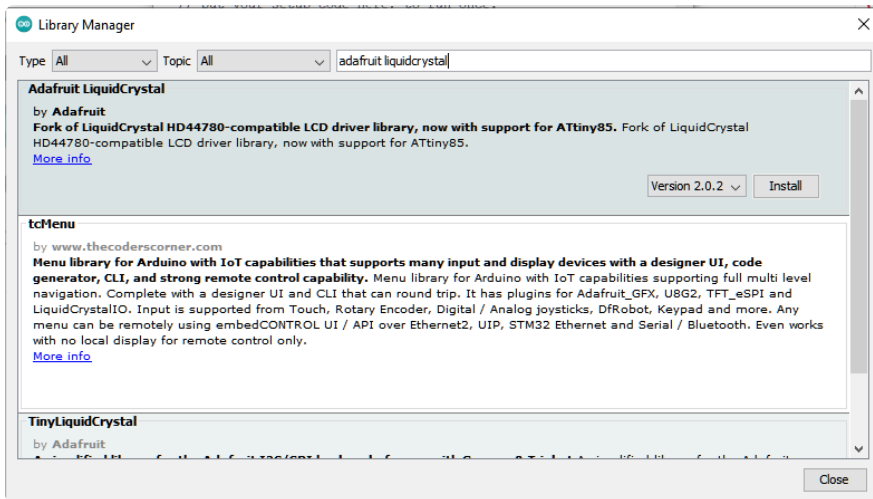


Install Adafruit_LiquidCrystal

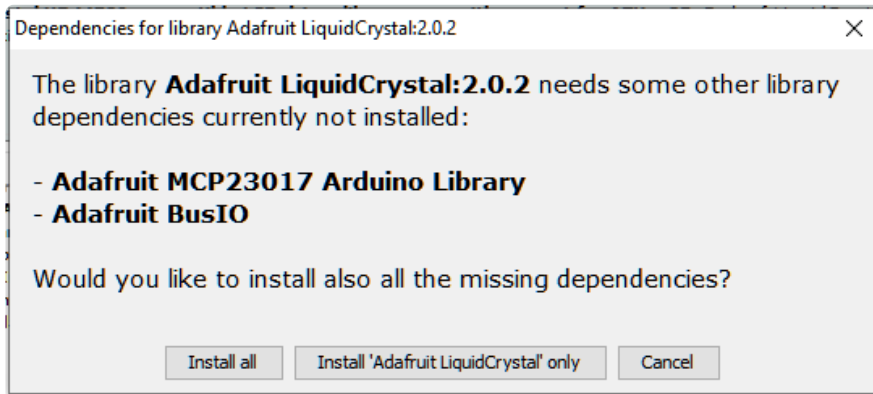
To begin reading sensor data, you will need to [use the Adafruit_LiquidCrystal library.](#) (). You can install the Adafruit_LiquidCrystal library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for Adafruit LiquidCrystal and select the Adafruit LiquidCrystal library:

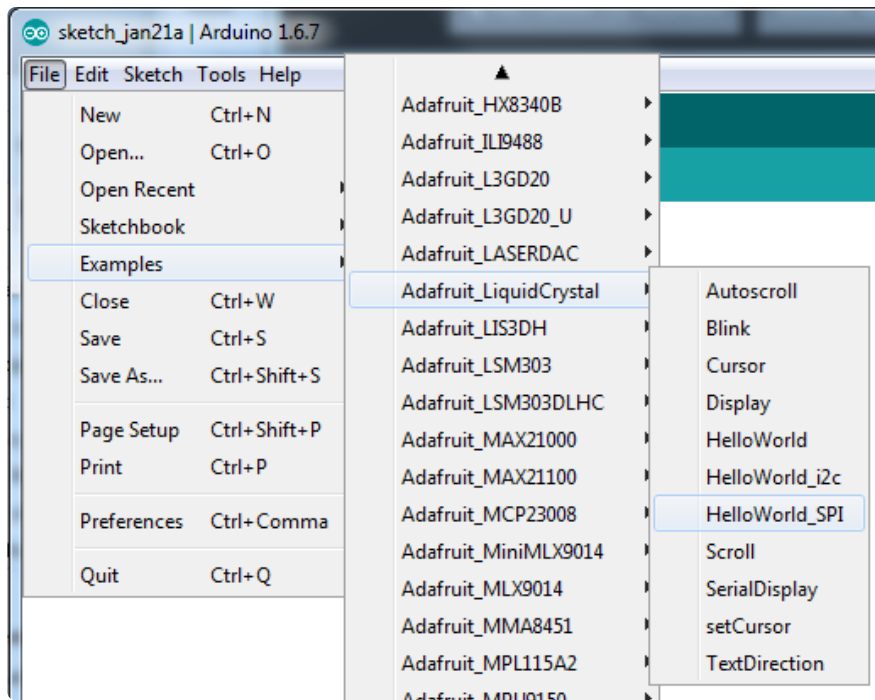


If asked about any dependencies, click "Install All".



Load Demo

Restart the IDE and load up the Adafruit_LiquidCrystal->HelloWorld_SPI demo



Upload the sketch. You should see the backlight turn on when the Arduino resets. If you don't see any characters, adjust the Contrast trim potentiometer with a mini-screwdriver until you see the text clearly

The default HelloWorld sketch blinks the backlight as well as updating the text.

If you see the backlight blinking that means your connection to the SPI port is OK but the contrast is too low or too high, or the LCD data pins are not solidly connected.

Check the contrast first by gently twisting the mini trim potentiometer, if that doesn't help, recheck your soldering and resolder all 16 of the LCD pins!



Once you're done, you can remove the blinking LED backlight code:

```
lcd.setBacklight(HIGH);  
delay(500);  
lcd.setBacklight(LOW);  
delay(500);
```

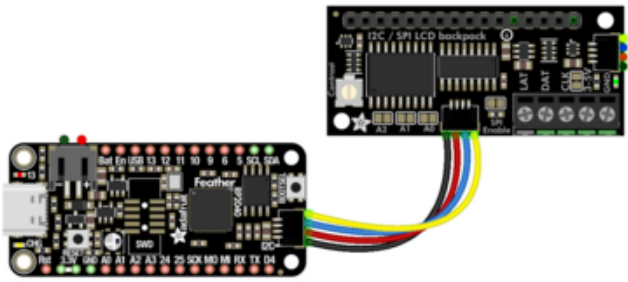
Python & CircuitPython

It's easy to use the character LCD backpack with CircuitPython or Python and the [Adafruit CircuitPython CharLCD \(\)](#) module. Just like [wiring up a LCD in parallel and controlling it from CircuitPython \(\)](#), you can use the I2C/SPI backpack to also control a LCD from CircuitPython or Python. The same [Adafruit CircuitPython CharLCD \(\)](#) module allows you to easily write Python code that controls a character LCD (remember only the single color backlight displays are supported by the backpack!).

CircuitPython Microcontroller Wiring

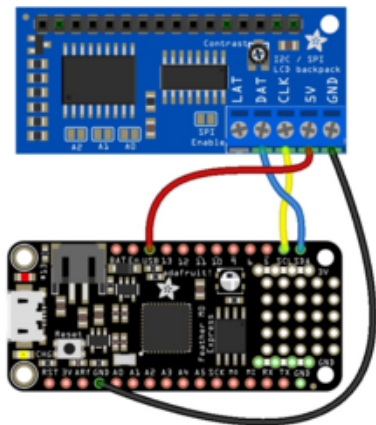
Here's an example of wiring a Feather RP2040 to the backpack with a STEMMA QT cable or via the terminal blocks:

A SAMD21 (M0) board does not have enough memory for the CharLCD library. Please use an M4, RP2040 or ESP32-S2/S3 board.



fritzing

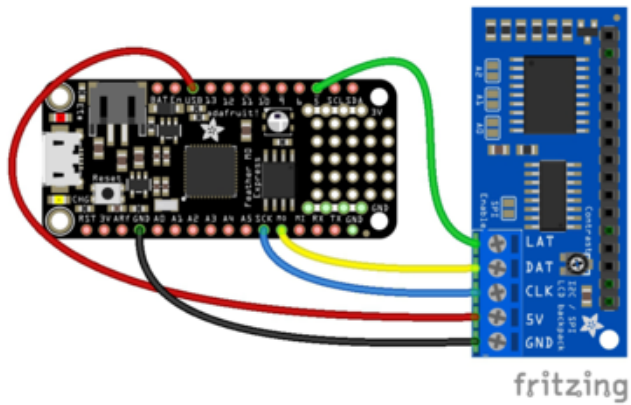
Board 3V to backpack VIN
 Board GND to backpack GND
 Board SCL to backpack CLK
 Board SDA to backpack DAT
 If using the original version of the backpack:



fritzing

Board VUSB to backpack 5V (remember using the VUSB output on a Feather means it must be connected to a 5V USB power source like your computer or a battery pack! You can alternatively use the 3.3V output of the Feather but it might not be enough to power the LCD.)

Or you can wire the LCD backpack to your board using the SPI interface. Remember you must also solder closed the SPI Enable bridge on the backpack! Here's an example of wiring a Feather M4 to the backpack with a hardware SPI connection:



Board 3V to backpack VIN
 Board GND to backpack GND.
 Board SCK to backpack CLK.
 Board MOSI to backpack DAT.
 Board D5 to backpack LAT.
 If using the original version of the backpack:



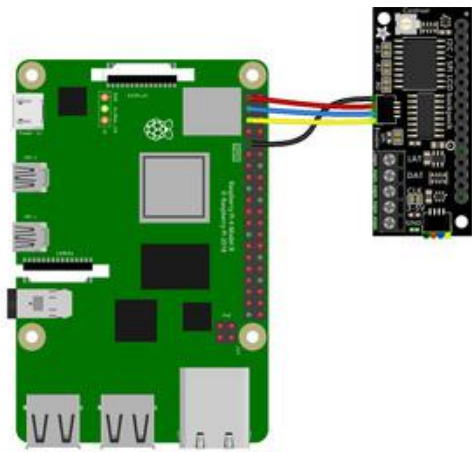
Board VUSB to backpack VIN (remember using the VUSB output on a Feather means it must be connected to a 5V USB power source like your computer or a battery pack! You can alternatively use the 3.3V output of the Feather but it might not be enough to power the LCD.)

Don't forget to solder the SPI jumper closed on the backpack if using the SPI wiring!

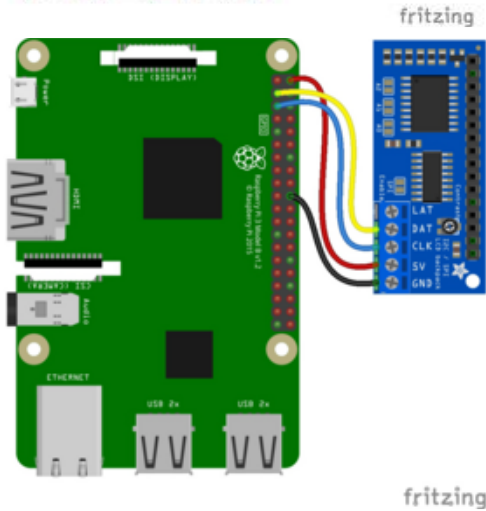
Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported \(\)](#).

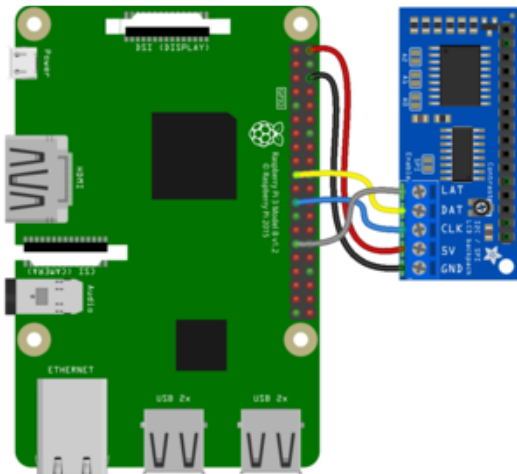
Here's the Raspberry Pi wired to the backpack using I2C:



Pi 3V to backpack VIN
 Pi GND to backpack GND.
 Pi SCL to backpack CLK.
 Pi SDA to backpack DAT.



Here's the Raspberry Pi wired to the backpack using SPI:



Pi 3V to backpack VIN
 Pi GND to backpack GND.
 Pi SCK to backpack CLK.
 Pi MOSI to backpack DAT.
 Pi GPIO5 to backpack LAT.

CircuitPython Installation of CharLCD Library

You'll need to install the [Adafruit CircuitPython CharLCD \(\)](#) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython \(\)](#) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle \(\)](#). Our CircuitPython starter guide has [a great page on how to use the library bundle \(\)](#).

You'll need to install the necessary libraries from the bundle:

- `adafruit_character_lcd`
- `adafruit_mcp230xx` (used only for I2C)
- `adafruit_74hc595` (used only for SPI)
- `adafruit_bus_device` (only needed on non-Express boards with `adafruit_bus_device` not built in)

Before continuing make sure your board's lib folder or root filesystem has the library files and folders copied over.

Next [connect to the board's serial REPL \(\)](#) so you are at the CircuitPython `>>>` prompt.

Python Installation of CharLCD Library

You'll need to install the `Adafruit_Blinka` library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready \(\)!](#)

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-charlcd`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

Python & CircuitPython Usage

To demonstrate the usage of the character LCD we'll initialize it and write text using Python code.

I2C Initialization

For an I2C backpack run the following code to import necessary modules, initialize the I2C bus, and create an instance of the character LCD class:

```
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd
i2c = busio.I2C(board.SCL, board.SDA)
cols = 16
rows = 2
lcd = character_lcd.Character_LCD_I2C(i2c, cols, rows)
```

Also notice you need to specify the number of columns and rows for your character LCD. The cols and rows variables above are setting these values for a 16x2 character LCD. Make sure to set the right value for your LCD!

SPI Initialization

For a SPI backpack as wired above run the following code instead (again be sure to set the right cols and rows for your display!):

```
import board
import busio
import digitalio
import adafruit_character_lcd.character_lcd_spi as character_lcd
spi = busio.SPI(board.SCK, MOSI=board.MOSI)
latch = digitalio.DigitalInOut(board.D5)
cols = 16
rows = 2
lcd = character_lcd.Character_LCD_SPI(spi, latch, cols, rows)
```

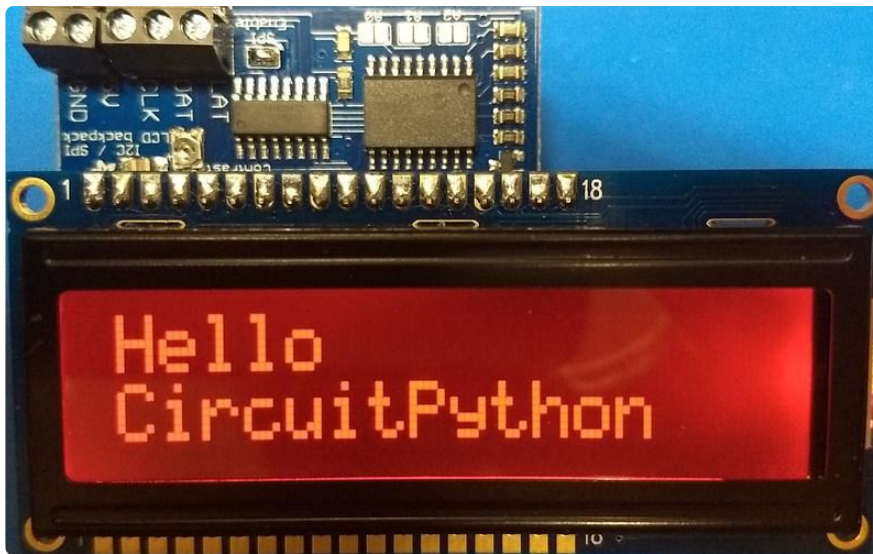
The only big difference with a SPI connection is that you must also import the digitalio module and pass in a DigitalInOut pin for the latch line. Notice board pin D5 is used for this wiring--be sure to change the pin if your wiring is different.

Backpack Usage

Now you're ready to start writing text and characters on the display! The usage of the LCD class is exactly the same as [shown in the parallel LCD wiring guide \(\)](#). Be sure to [check out that guide \(\)](#) for a complete discussion of LCD usage.

As a quick test though you can run the following code to use the `message` property to write text to the display:

```
lcd.message = "Hello\nCircuitPython!"
```



[See the parallel LCD guide for more functions you can call to control the LCD! \(\)](#)

That's all there is to using the I2C/SPI backpack with CircuitPython!

Full Example Code

As a complete example of I2C usage, see the [charlcd_I2C_simpletest.py \(\)](#) example in the character LCD library. Save this as code.py on your board and it will run a small demo of LCD functions.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Simple test for 16x2 character lcd connected to an MCP23008 I2C LCD backpack."""
import time
import board
import adafruit_character_lcd.character_lcd_i2c as character_lcd

# Modify this if you have a different sized Character LCD
lcd_columns = 16
lcd_rows = 2

# Initialise I2C bus.
i2c = board.I2C() # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C() # For using the built-in STEMMMA QT connector on a
microcontroller

# Initialise the lcd class
lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)

# Turn backlight on
lcd.backlight = True
# Print a two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
```

```

lcd.clear()
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll message to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
time.sleep(5)
# Turn backlight off
lcd.backlight = False
time.sleep(2)

```

And a complete example of SPI usage is in the [charlcd_SPI_simpletest.py \(\)](#) example in the library. Save this as code.py on your board and it will run a small demo of LCD functions.

```

# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Simple test for 16x2 character LCD connected to 74HC595 SPI LCD backpack."""
import time
import board
import busio
import digitalio
import adafruit_character_lcd.character_lcd_spi as character_lcd

# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# Backpack connection configuration:
clk = board.SCK # Pin connected to backpack CLK.
data = board.MOSI # Pin connected to backpack DAT/data.
latch = board.D5 # Pin connected to backpack LAT/latch.

# Initialise SPI bus.
spi = busio.SPI(clk, MOSI=data)

# Initialise the LCD class
latch = digitalio.DigitalInOut(latch)
lcd = character_lcd.Character_LCD_SPI(spi, latch, lcd_columns, lcd_rows)

```



```
# Turn backlight on
lcd.backlight = True
# Print a two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
lcd.clear()
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll message to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
# Turn backlight off
lcd.backlight = False
time.sleep(2)
```

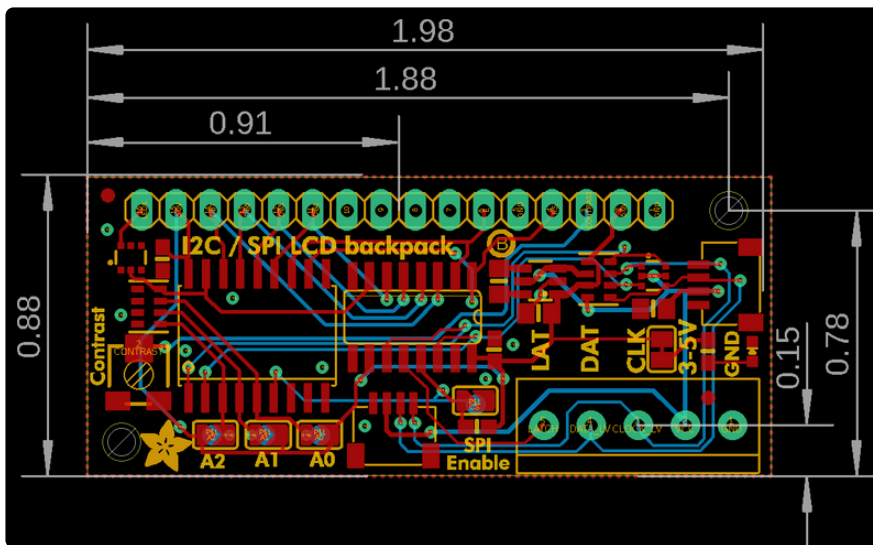
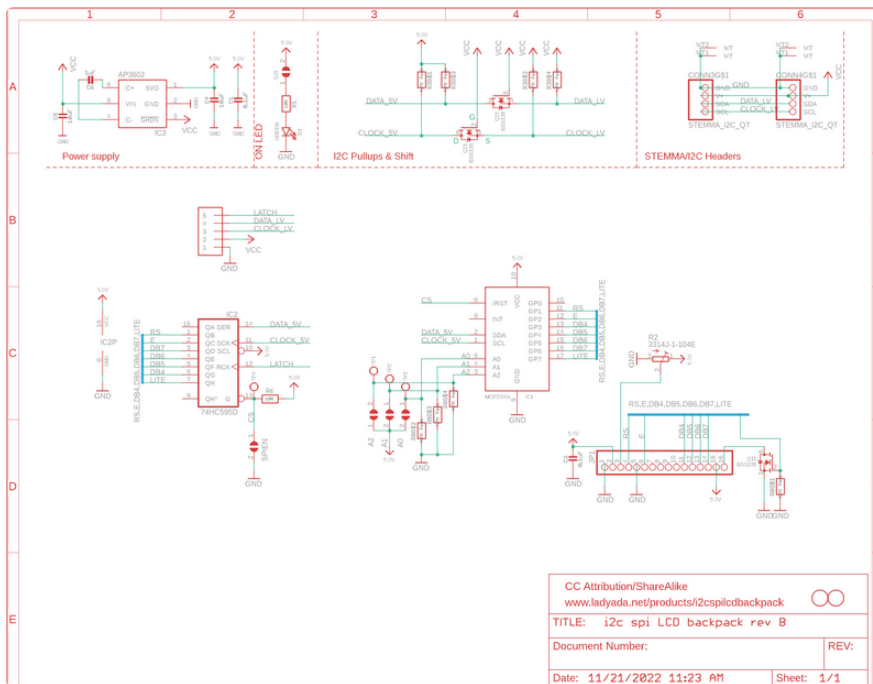
Python Docs

[Python Docs \(\)](#)

Downloads

- [MCP23008 Datasheet \(\)](#)
- [EagleCAD PCB Files on GitHub \(\)](#)
- [Original Fritzing object available in the Adafruit Fritzing Library \(\)](#)
- [STEMMA QT Fritzing object in the Adafruit Fritzing Library \(\)](#)

STEMMA QT Schematic and Fab Print



Original Schematic and Fab Print

