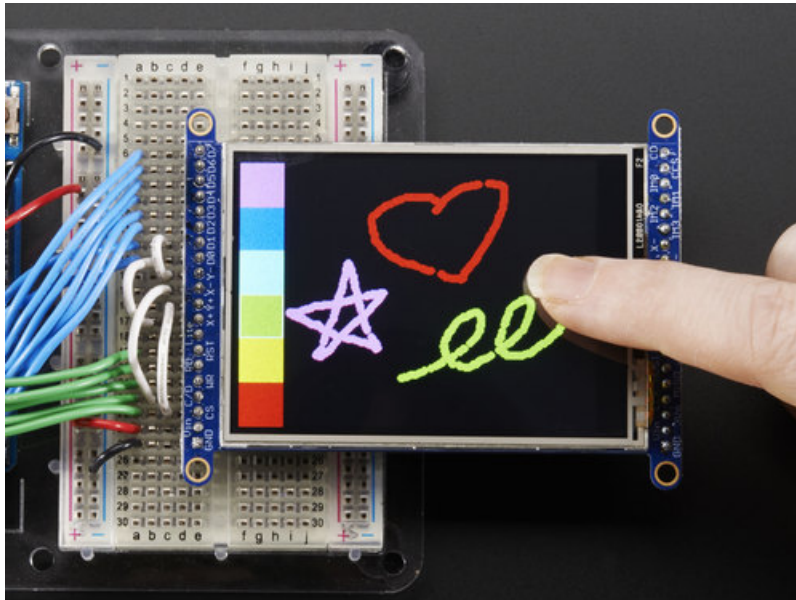


Adafruit 2.8" and 3.2" Color TFT Touchscreen Breakout v2

Created by lady ada

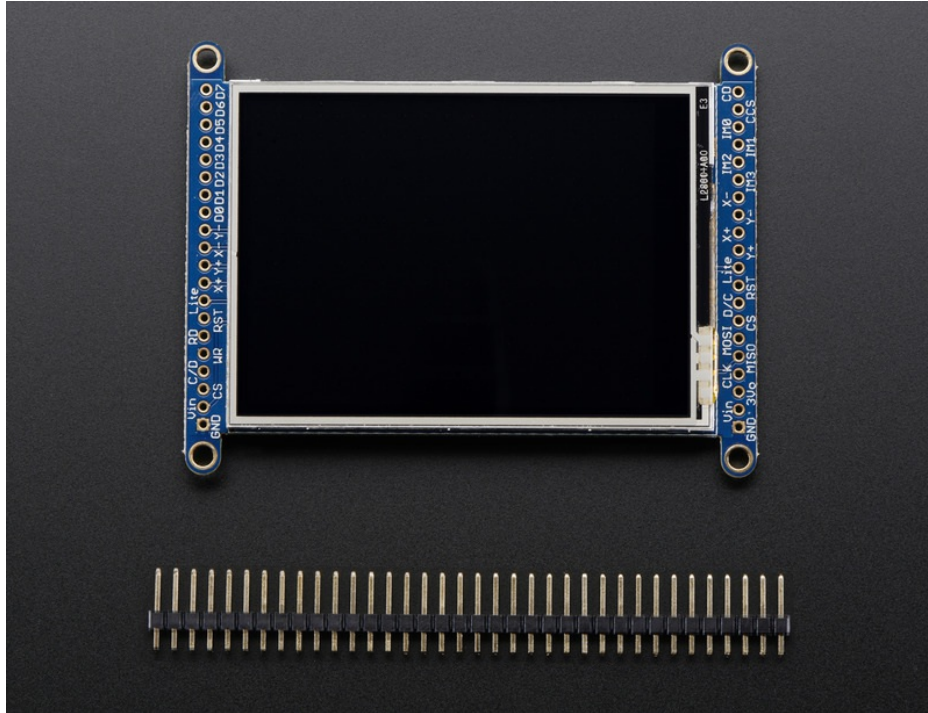


Last updated on 2019-02-10 07:49:29 PM UTC

Guide Contents

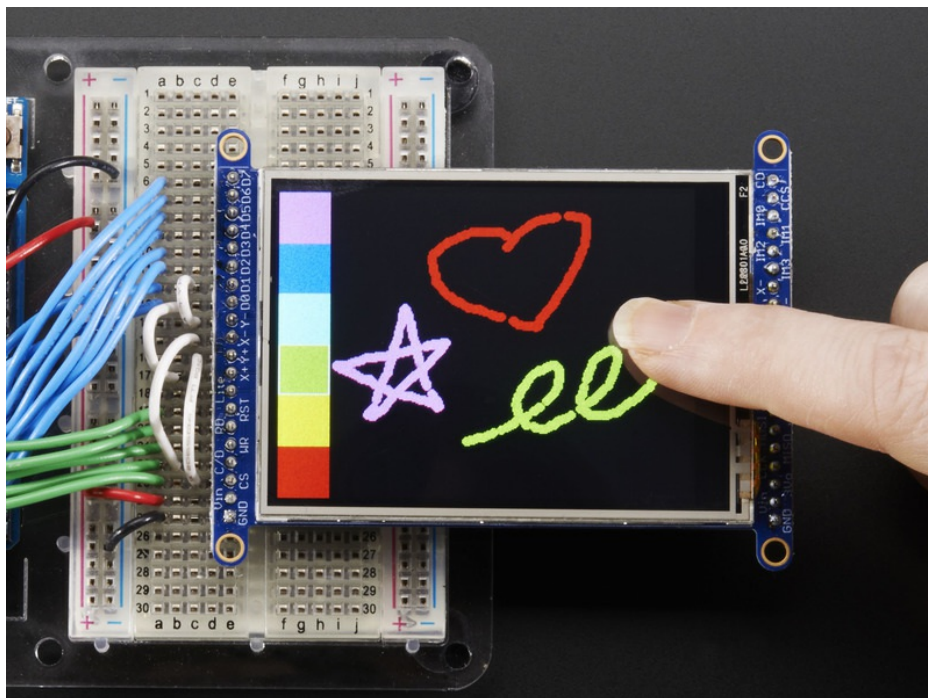
Guide Contents	2
Overview	3
Pinouts	6
SPI Mode	6
Resistive touch pins	7
Capacitive touch pins	7
8-Bit Mode	8
Wiring and Test	9
8-Bit Wiring and Test	10
8-Bit Wiring	10
Part 1 - Power & backlight test	10
Part 2 - Data Bus Lines	11
8-Bit Library Install	14
SPI Wiring and Test	18
SPI Mode Jumpers	18
Wiring	19
Install Libraries	20
Install Adafruit ILI9341 TFT Library	20
Bitmaps (SPI Mode)	24
Adafruit GFX library	26
Resistive Touchscreen	28
Download Library	28
Touchscreen Paint (SPI mode)	29
Touchscreen Paint (8-Bit mode)	29
Capacitive Touchscreen	31
Download the FT6206 Library	31
FT6206 Library Reference	32
Touchscreen Interrupt pin	32
FT6206 Library Reference	34
Downloads	35
Datasheets & Files	35
2.8" and 3.2" Resistive Touch Schematic	35
Capacitive Touch Schematic	35
2.8" TFT Layout Diagram	36
F.A.Q.	38
If I drive this display at very high speeds I get 'video tearing' effects, how can I synchronize the display refreshes?	38
Display does not work on initial power but does work after a reset.	38

Overview



Add some jazz & pizzazz to your project with a color touchscreen LCD. These TFT displays are big (2.8" or 3.2" diagonal) bright (4 or 6 white-LED backlight) and colorful! 240x320 pixels with individual RGB pixel control, this has way more resolution than a black and white 128x64 display.

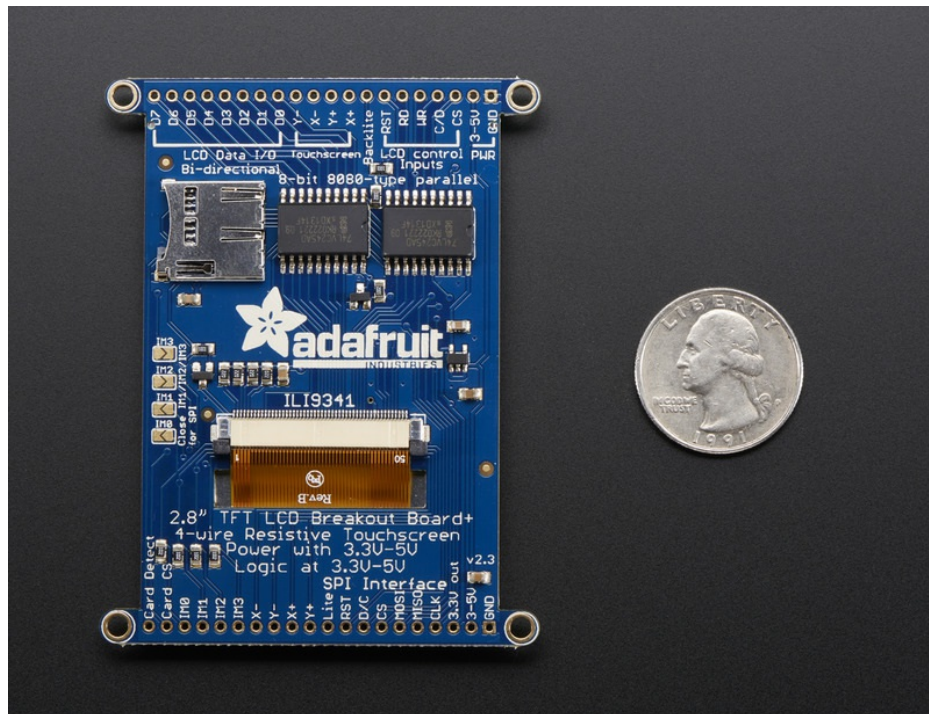
As a bonus, this display has either a resistive or capacitive touchscreen attached to it already, so you can detect finger presses anywhere on the screen.



This display has a controller built into it with RAM buffering, so that almost no work is done by the microcontroller. **The display can be used in two modes: 8-bit or SPI.** For 8-bit mode, you'll need 8 digital data lines and 4 or 5 digital control lines to read and write to the display (12 lines total). SPI mode requires only 5 pins total (SPI data in, data out, clock, select, and d/c) but is slower than 8-bit mode.

If you have the **resistive** touch version, 4 pins are required for the touch screen (2 digital, 2 analog) or [you can purchase and use our resistive touchscreen controller \(not included\) to use I2C or SPI](http://adafruit.it/1571) (<http://adafruit.it/1571>)

If you have the **capacitive** touch version, there is a capacitive touch controller chip already installed that communicates of standard I2C plus an IRQ line.

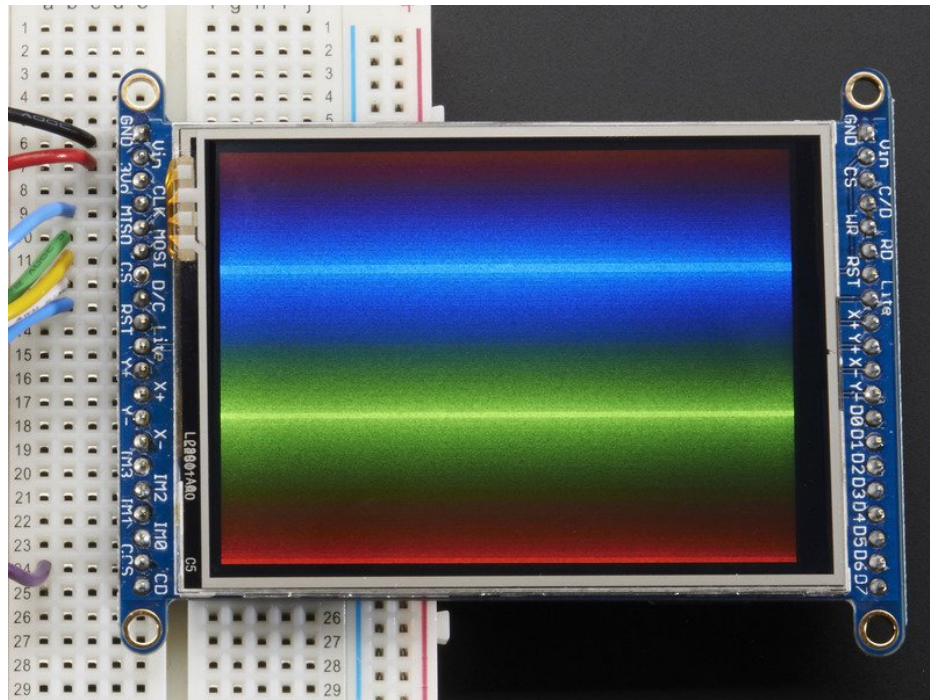


Of course, we wouldn't just leave you with a datasheet and a "good luck!". For 8-bit interface fans [we've written a full open source graphics library that can draw pixels, lines, rectangles, circles, text, and more](https://adafruit.it/aHk) (<https://adafruit.it/aHk>). For SPI users, [we have a library as well](https://adafruit.it/d4d) (<https://adafruit.it/d4d>), its separate from the 8-bit library since both versions are heavily optimized.

For resistive touch, we [also have a touch screen library that detects x, y and z \(pressure\)](https://adafruit.it/aT1) (<https://adafruit.it/aT1>) and example code to demonstrate all of it.

For capacitive touch, we have an I2C interface library for the captouch chip. (<https://adafruit.it/dGG>)

If you are using an Arduino-shaped microcontroller, [check out our TFT shield version of this same display, with SPI control and a touch screen controller as well](http://adafruit.it/1651) (<http://adafruit.it/1651>)

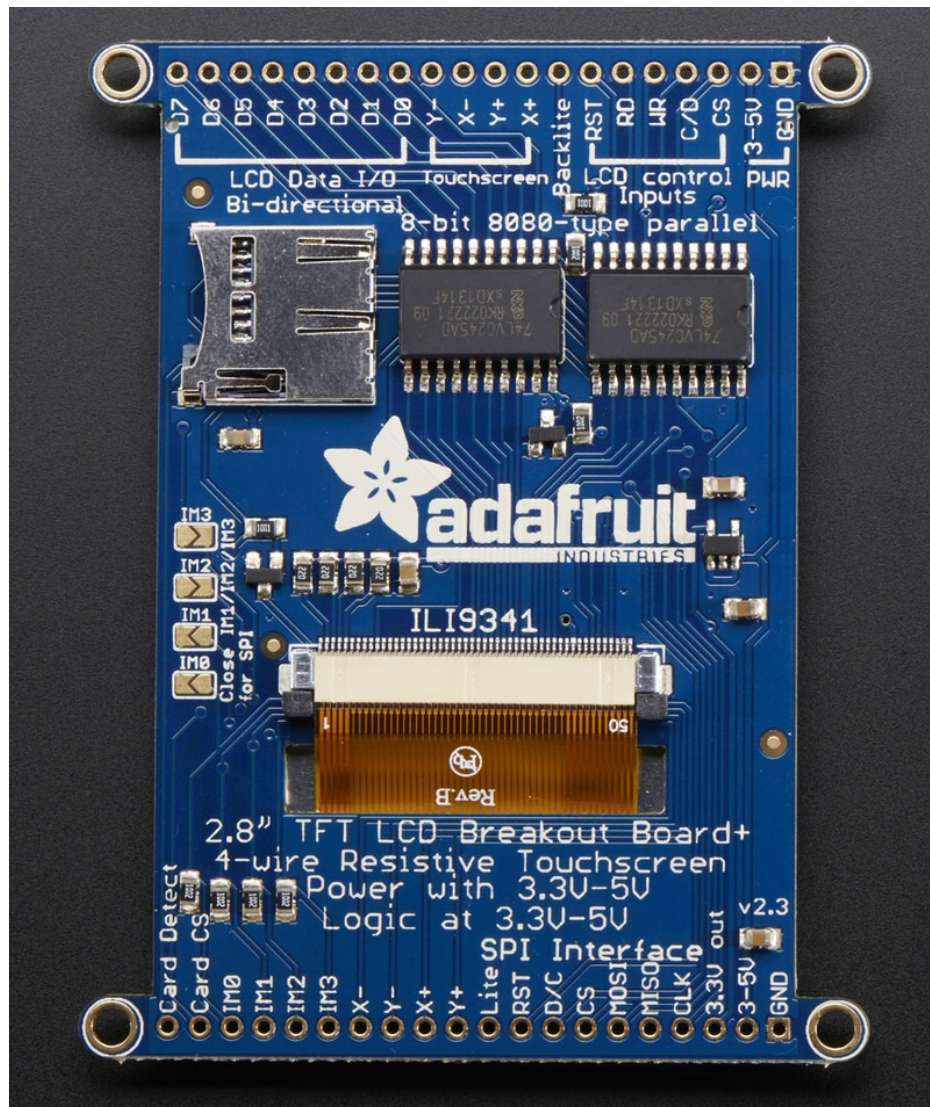


Pinouts

The 2.8" and 3.2" TFT display on this breakout supports many different modes - so many that the display itself has 50 pins. However, we think most people really only use 2 different modes, either "SPI" mode or 8-bit mode (which includes both 6800 and 8080). Each 'side' of the display has all the pins required for that mode. You can switch between modes, by rewiring the display, but it cannot be used in two modes at the same time!

All logic pins, both 8-bit and SPI sides, are 3-5V logic level compatible, the 74LVX245 chips on the back perform fast level shifting so you can use either kind of logic levels. If there's data output, the levels are at 3.3V

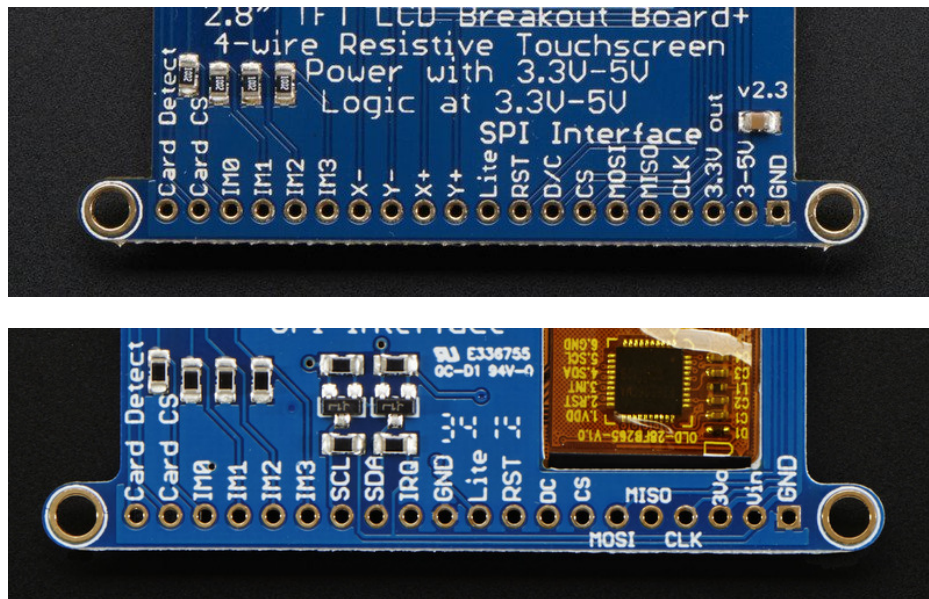
We show the 2.8" version of this breakout in the photos below but the 3.2" TFT is identical, just a lil bit bigger



SPI Mode

This is what we think will be a popular mode when speed is not of the utmost importance. It doesn't use as many pins (only 4 to draw on the TFT if you skip the MISO pin), is fairly flexible, and easy to port to various microcontrollers. It also allows using a microSD card socket on the same SPI bus. However, it's slower than parallel 8-bit mode because you

have to send each bit at a time instead of 8-bits at a time. Tradeoffs!



- **GND** - this is the power and signal ground pin
- **3-5V / Vin** - this is the power pin, connect to 3-5VDC - it has reverse polarity protection but try to wire it right!
- **3.3Vout** - this is the 3.3V output from the onboard regulator
- **CLK** - this is the SPI clock input pin
- **MISO** - this is the SPI Master In Slave Out pin, its used for the SD card mostly, and for debugging the TFT display. It isn't necessary for using the TFT display which is write-only
- **MOSI** - this is the SPI Master Out Slave In pin, it is used to send data from the microcontroller to the SD card and/or TFT
- **CS** - this is the TFT SPI chip select pin
- **D/C** - this is the TFT SPI data or command selector pin
- **RST** - this is the TFT reset pin. There's auto-reset circuitry on the breakout so this pin is not required but it can be helpful sometimes to reset the TFT if your setup is not always resetting cleanly. Connect to ground to reset the TFT
- **Lite** - this is the PWM input for the backlight control. It is by default pulled high (backlight on) you can PWM at any frequency or pull down to turn the backlight off
- **IM3 IM2 IM1 IM0** - these are interface control set pins. In general these breakouts aren't used, and instead the onboard jumpers are used to fix the interface to SPI or 8-bit. However, we break these out for advanced use and also for our test procedures
- **Card CS / CCS** - this is the SD card chip select, used if you want to read from the SD card.
- **Card Detect / CD** - this is the SD card detect pin, it floats when a card is inserted, and tied to ground when the card is not inserted. We don't use this in our code but you can use this as a switch to detect if an SD card is in place without trying to electrically query it. Don't forget to use a pullup on this pin if so!

Resistive touch pins

- **Y+ X+ Y- X-** these are the 4 resistive touch screen pads, which can be read with analog pins to determine touch points. They are completely separated from the TFT electrically (the overlay is glued on top)

Capacitive touch pins

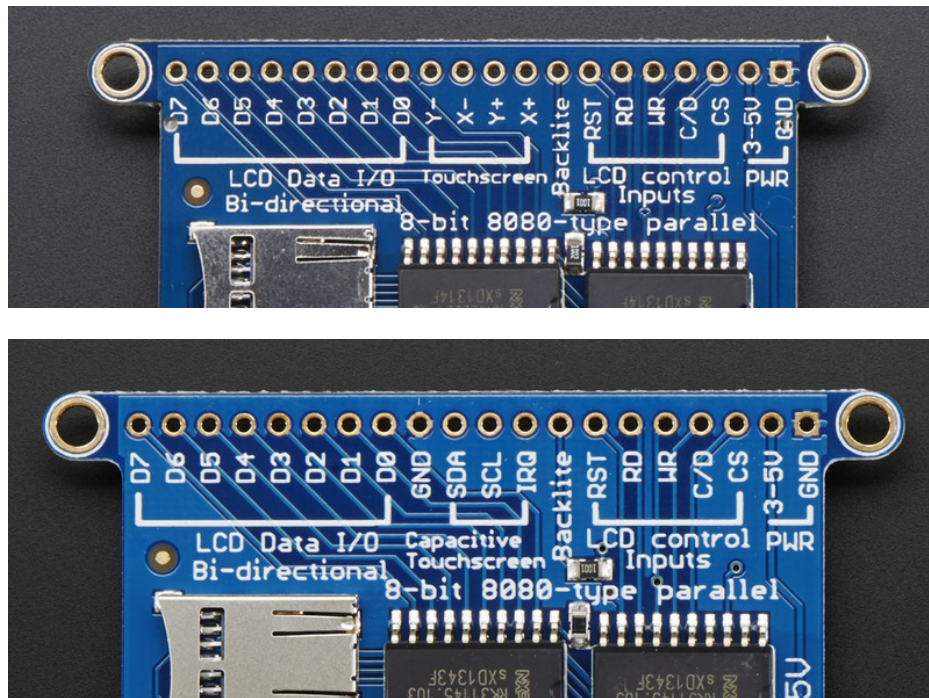
- **SDA** - this is the I2C data pin for the captouch chip, there's level shifting on this pin so you can use 3-5V logic.

There's also a 10K pullup

- **SDA** - this is the I2C clock pin for the captouch chip, there's level shifting on this pin so you can use 3-5V logic. There's also a 10K pullup
- **IRQ** - this is the captouch interrupt pin. When a touch is detected, this pin goes low.

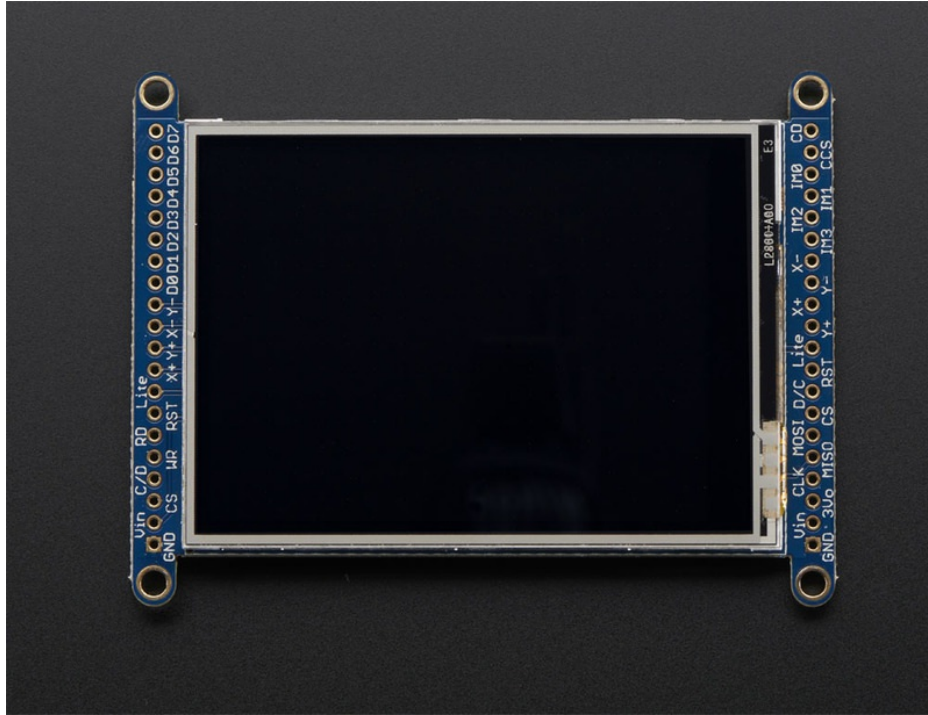
8-Bit Mode

This mode is for when you have lots of pins and want more speed. In this mode we send 8 bits at a time, so it needs way more pins, 12 or so (8 bits plus 4 control)! This isn't recommended because most microcontrollers don't have a ton of pins and also we optimize our libraries for SPI!



- **GND** - this is the power and signal ground pin
- **3-5V (Vin)** - this is the power pin, connect to 3-5VDC - it has reverse polarity protection but try to wire it right!
- **CS** - this is the TFT 8-bit chip select pin (it is also tied to the SPI mode **CS** pin)
- **C/D** - this is the TFT 8-bit data or command selector pin. It is **not the same as the SPI D/C pin!** Instead, it's the same as the SPI CLK pin.
- **WR** - this is the TFT 8-bit write strobe pin. It is also connected to the SPI **D/C** pin
- **RD** - this is the TFT 8-bit read strobe pin. You may not need this pin if you don't want to read data from the display
- **RST** - this is the TFT reset pin. There's auto-reset circuitry on the breakout so this pin is not required but it can be helpful sometimes to reset the TFT if your setup is not always resetting cleanly. Connect to ground to reset the TFT
- **Backlite** - this is the PWM input for the backlight control. It is by default pulled high (backlight on) you can PWM at any frequency or pull down to turn the backlight off
- **D0** thru **D8** - these are the 8 bits of parallel data sent to the TFT in 8-bit mode. **D0** is the least-significant-bit and **D8** is the MSB

Wiring and Test



We tried to make this TFT breakout useful for both high-pin microcontrollers that can handle 8-bit data transfer modes as well as low-pincount micros like the Arduino UNO and Leonardo that are OK with SPI.

Essentially, the tradeoff is pins for speed. SPI is about 2-4 times slower than 8-bit mode, but that may not matter for basic graphics!

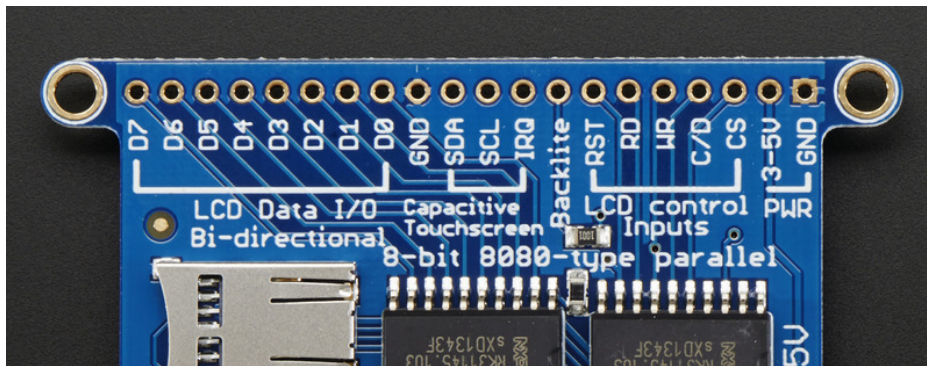
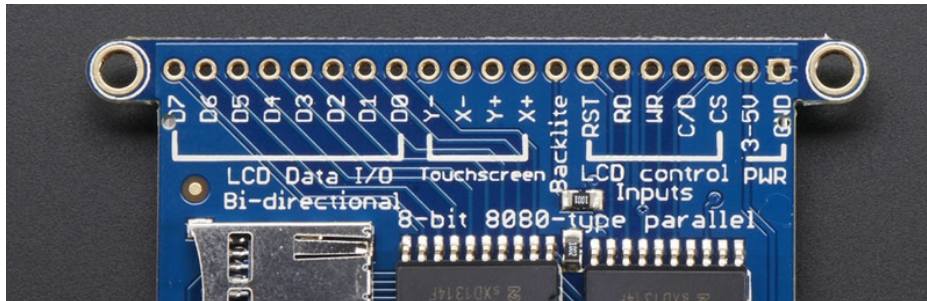
In addition, SPI mode has the benefit of being able to use the onboard microSD card socket for reading images. We don't have support for this in 8-bit mode so if you want to have an all-in-one image viewer type application, use SPI!

8-Bit Wiring and Test

8-Bit Wiring

Wiring up the 8-bit mode is kind of a pain, so we really only recommend doing it for UNO (which we show) and Mega (which we describe, and is pretty easy since its 8 pins in a row). Anything else, like a Leonardo or Micro, we strongly recommend going with SPI mode since we don't have an example for that. Any other kind of 'Arduino compatible' that isn't an Uno, try SPI first. The 8-bit mode is hand-tweaked in the **Adafruit_TFTLCD pin_magic.h** file. Its really only for advanced users who are totally cool with figuring out bitmasks for various ports & pins.

Really, we'll show how to do the UNO but anything else? go with SPI!



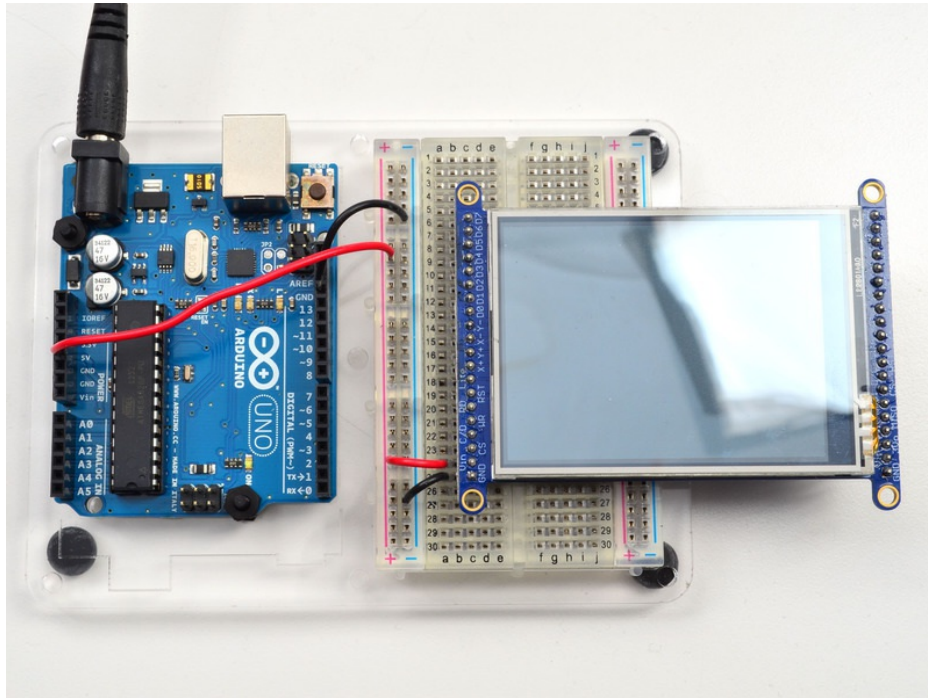
We show the 2.8" version of this breakout in the photos below but the 3.2" TFT is identical, just a lil bit bigger

Make sure you're soldering and connecting to the 8-bit side!

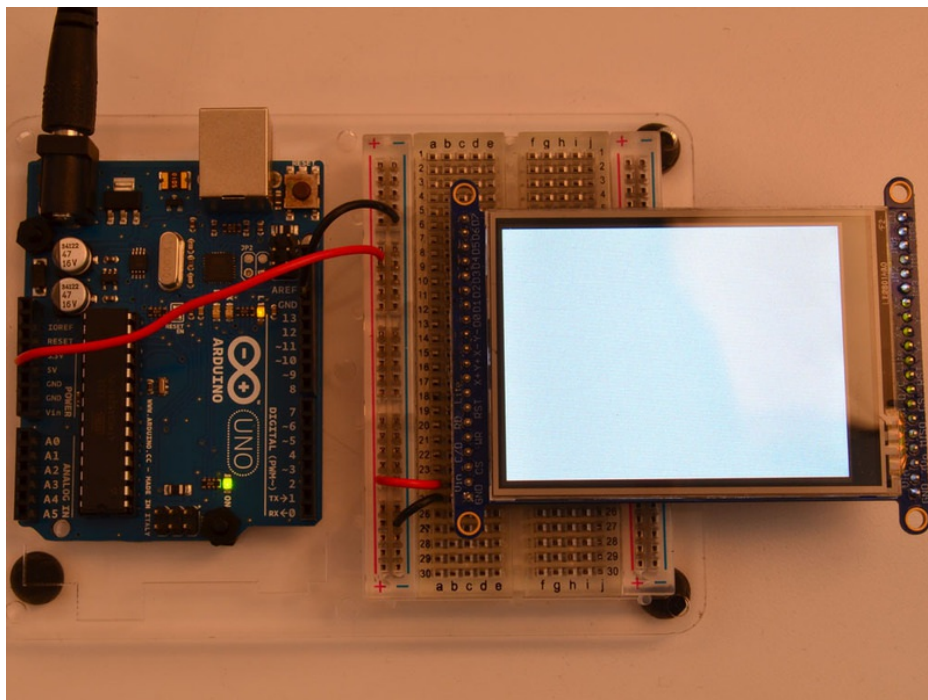
Part 1 - Power & backlight test

Begin by wiring up the **3-5VDC** and **GND** pins.

Connect the **3-5V** pin to **5V** and **GND** to **GND** on your Arduino. I'm using the breadboard rails but you can also just wire directly.



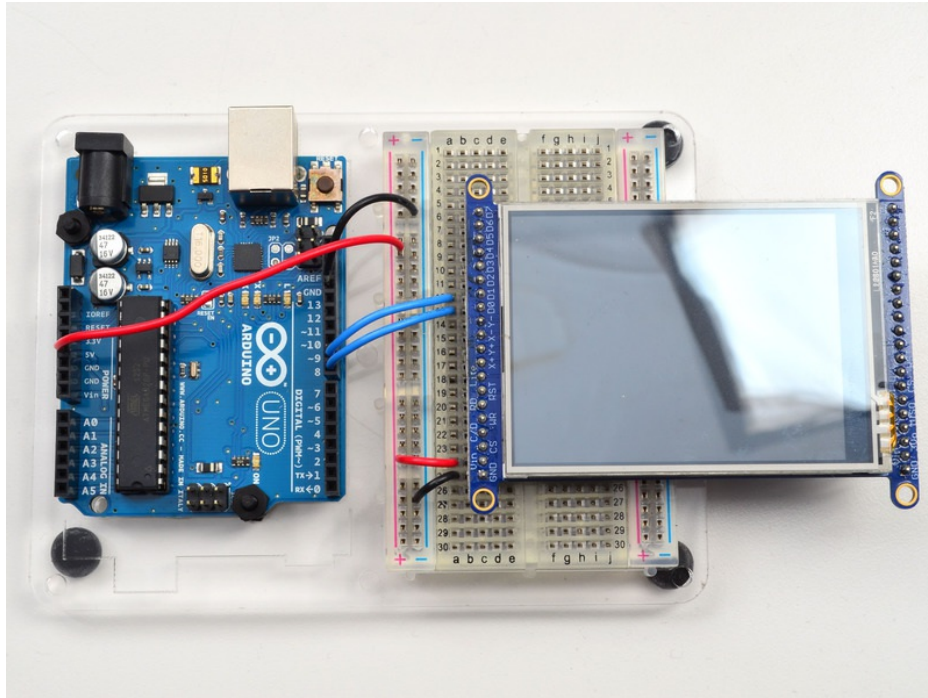
Power it up and you should see the white backlight come on.



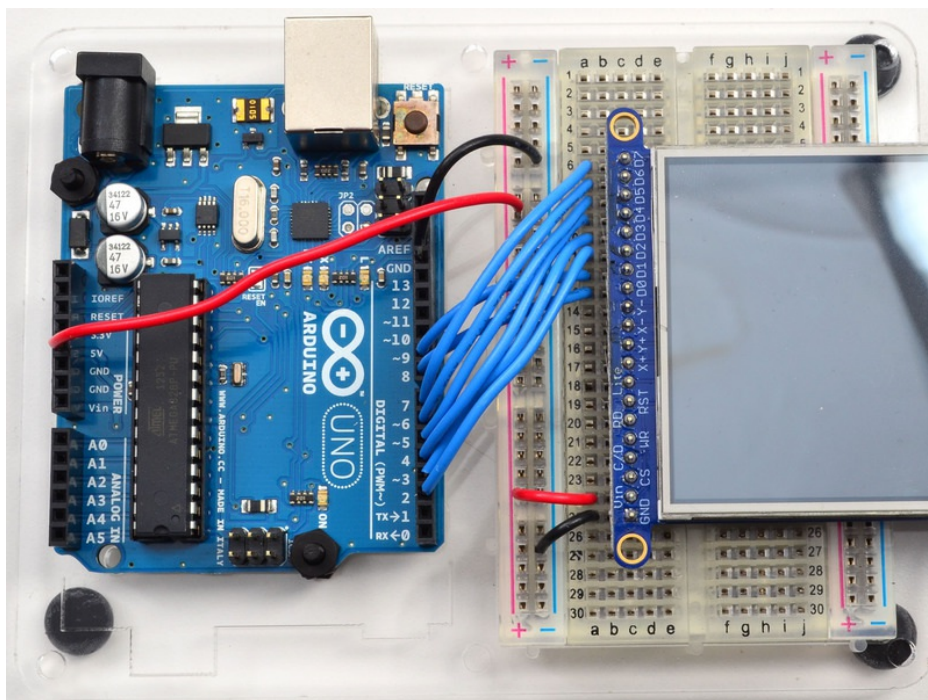
Part 2 - Data Bus Lines

Now that the backlight is working, we can get the TFT LCD working. There are many pins required, and to keep the code running fairly fast, we have 'hardcoded' Arduino digital pins **#2-#9** for the 8 data lines.

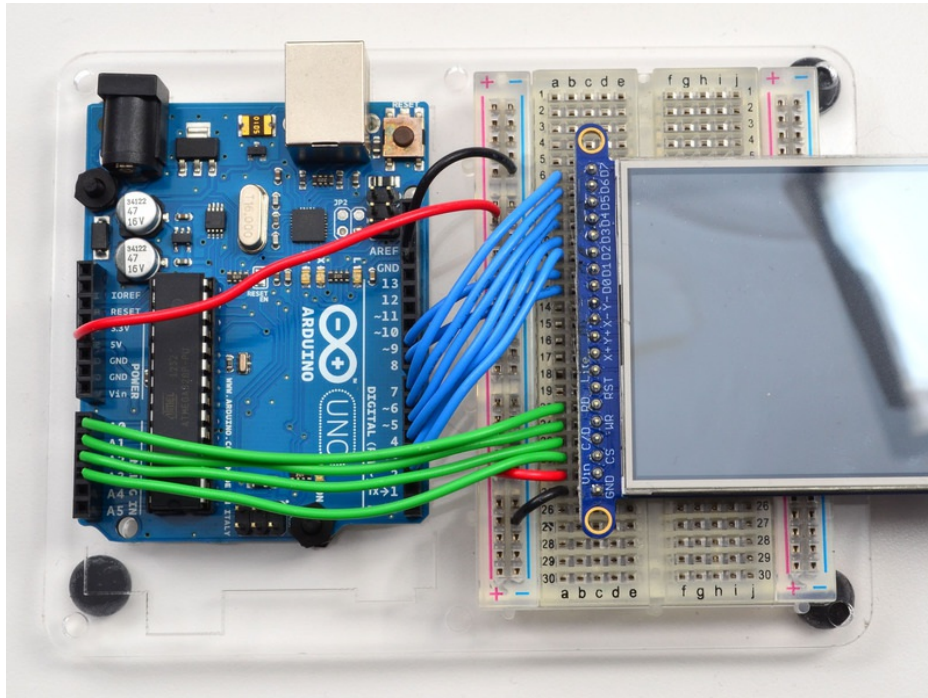
However, they are not in that order! D0 and D1 go to **digital #8 and #9**, then D2-D7 connect to **#2 thru #7**. This is because Arduino pins **#0 and #1** are used for serial data so we can't use them



Begin by connecting **D0** and **D1** to digital **#8** and **9** respectively as seen above. If you're using a Mega, connect the TFT Data Pins **D0-D1** to Mega pins **#22-23**, in that order. Those Mega pins are on the 'double' header.

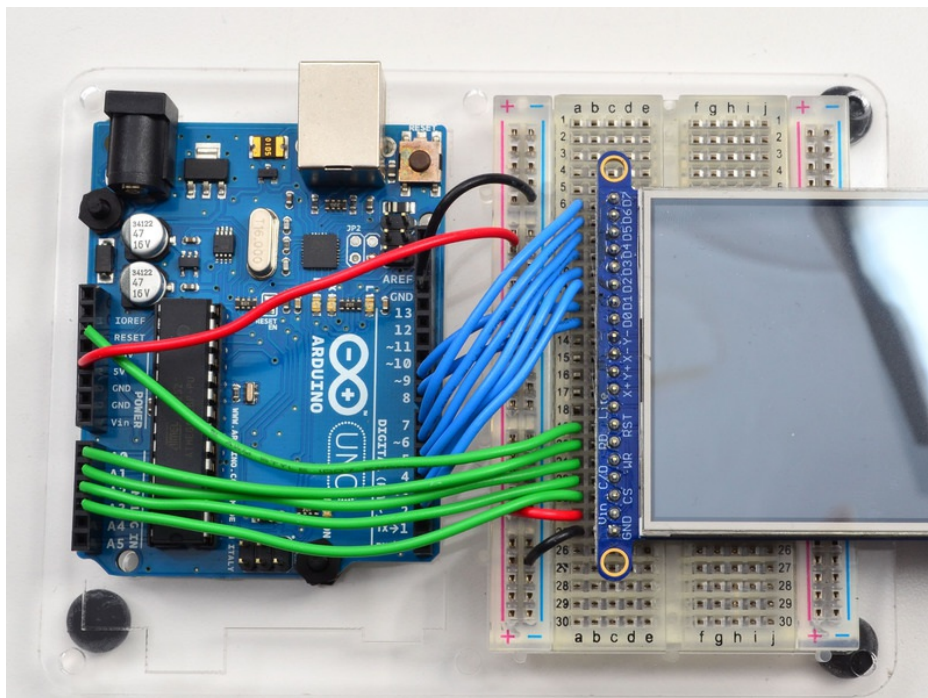


Now you can connect the remaining 6 pins over. Connect **D2-D7** on the TFT pins to digital **2** thru **7** in that order. If you're using a Mega, connect the TFT Data Pins **D2-D7** to Mega pins **#24-29**, in that order. Those Mega pins are on the 'double' header.



In addition to the 8 data lines, you'll also need 4 or 5 control lines. These can later be reassigned to any digital pins, they're just what we have in the tutorial by default.

- Connect the third pin **CS** (Chip Select) to Analog 3
- Connect the fourth pin **C/D** (Command/Data) to Analog 2
- Connect the fifth pin **WR** (Write) to Analog 1
- Connect the sixth pin **RD** (Read) to Analog 0



You can connect the seventh pin **RST** (Reset) to the Arduino Reset line if you'd like. This will reset the panel when the Arduino is Reset. You can also use a digital pin for the LCD reset if you want to manually reset. There's auto-reset

circuitry on the board so you probably don't need to use this pin at all and leave it disconnected

The **RD** pin is used to read the chip ID off the TFT. Later, once you get it all working, you can remove this pin and the ID test, although we suggest keeping it since its useful for debugging your wiring.

OK! Now we can run some code

8-Bit Library Install

We have example code ready to go for use with these TFTs. It's written for Arduino, which should be portable to any microcontroller by adapting the C++ source.

Two libraries need to be downloaded and installed: first is the [Adafruit_TFTLCD library \(https://adafru.it/aHk\)](https://adafru.it/aHk) (this contains the low-level code specific to this device), and second is the [Adafruit GFX Library \(https://adafru.it/aJa\)](https://adafru.it/aJa) (which handles graphics operations common to many displays we carry). If you have **Adafruit_GFX** already, make sure its the most recent version since we've made updates for better performance

<https://adafru.it/dcW>

<https://adafru.it/dcW>

<https://adafru.it/cBB>

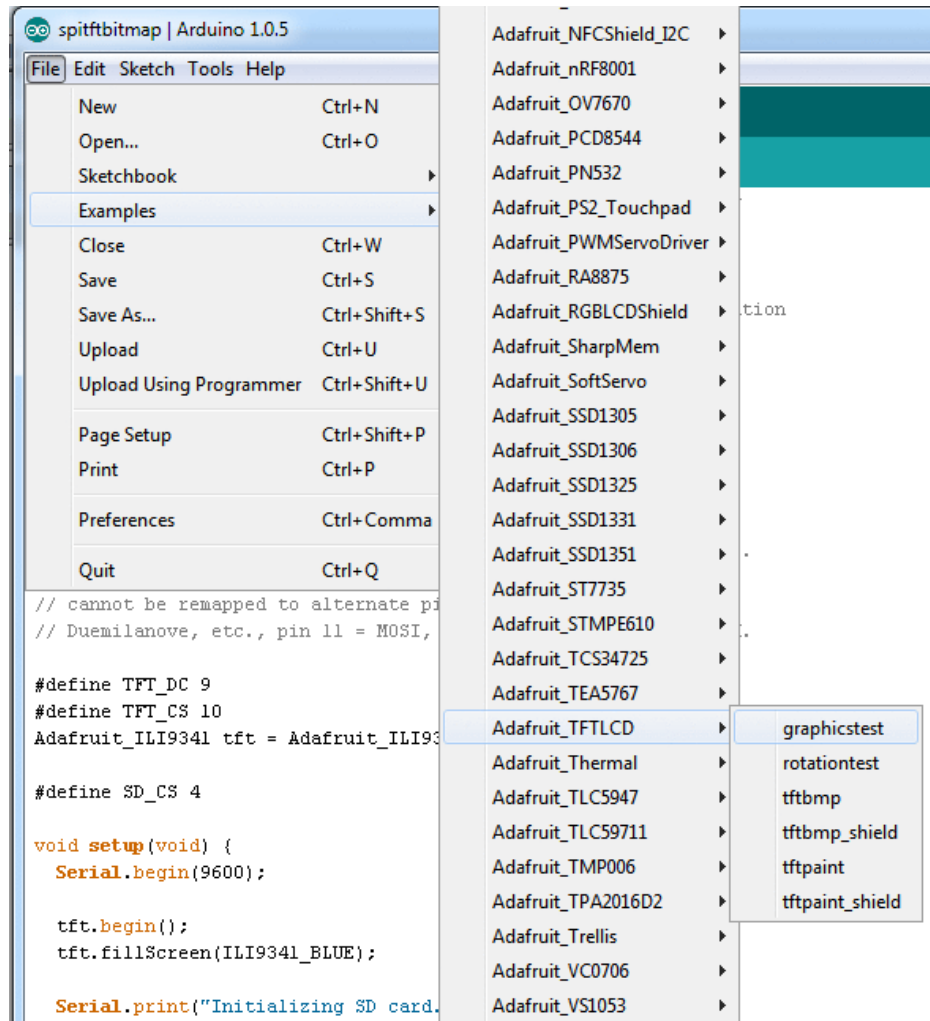
<https://adafru.it/cBB>

Download both ZIP files, uncompress and rename the folders to **Adafruit_TFTLCD** (contains **Adafruit_TFTLCD.cpp** and **.h**) and **Adafruit_GFX** (contains **Adafruit_GFX.cpp** and **.h**) respectively. Then place them inside your Arduino **libraries** folder and restart the Arduino IDE. If this is all unfamiliar, we have a [tutorial introducing Arduino library concepts and installation \(https://adafru.it/aYM\)](https://adafru.it/aYM).

In the **Adafruit_TFTLCD** Library folder, you may need to edit **Adafruit_TFTLCD.h**. On about line 12, you will see

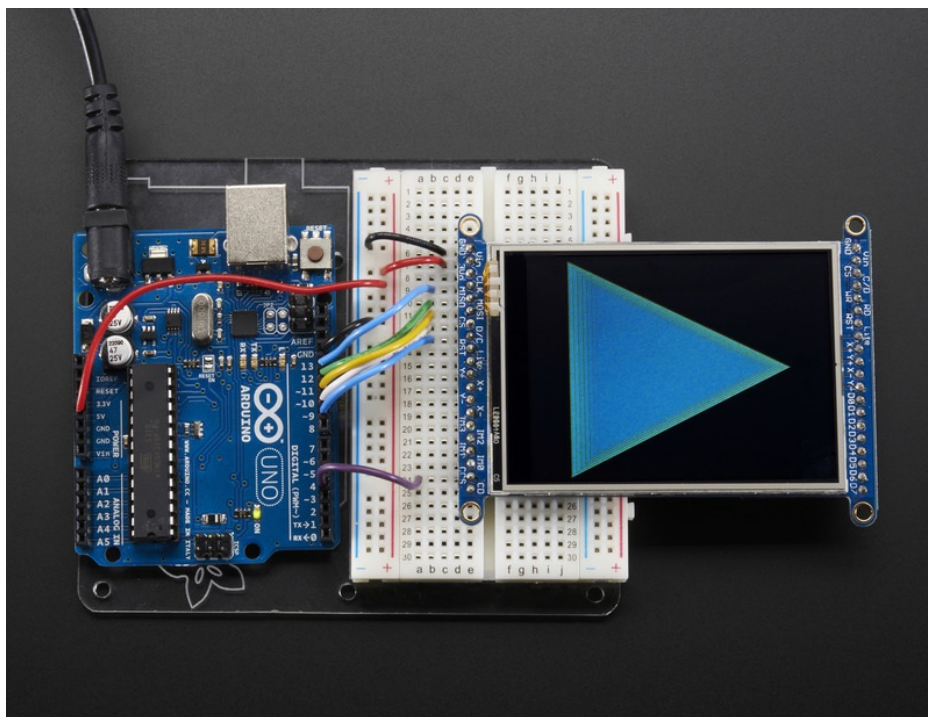
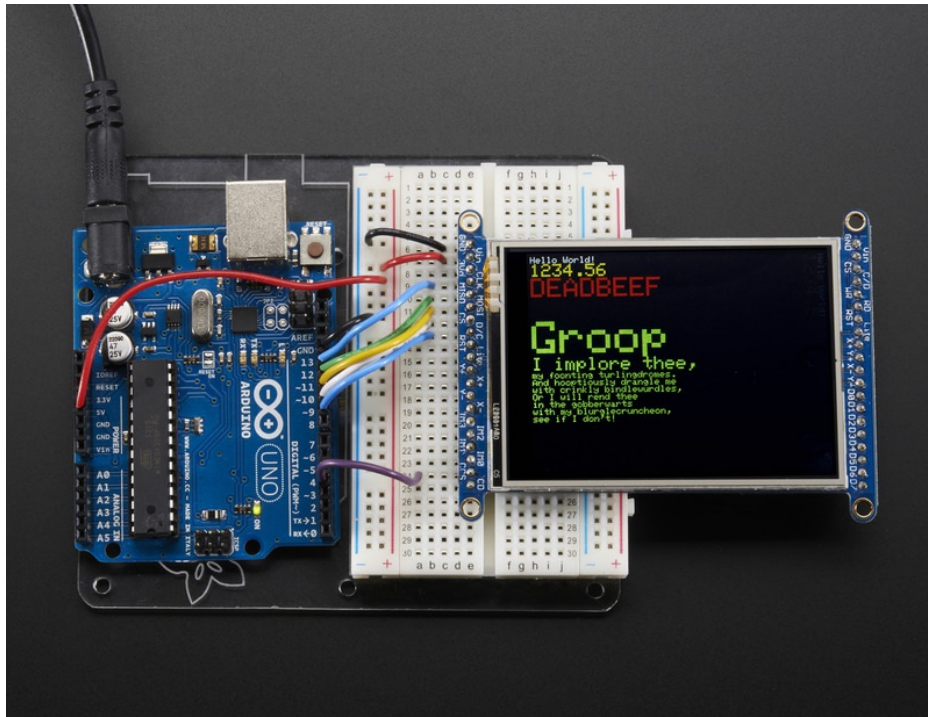
```
#define USE_ADAFRUIT_SHIELD_PINOUT
```

Make sure this line is commented out with a **//** in front (it should but if you're having issues, its worth checking).

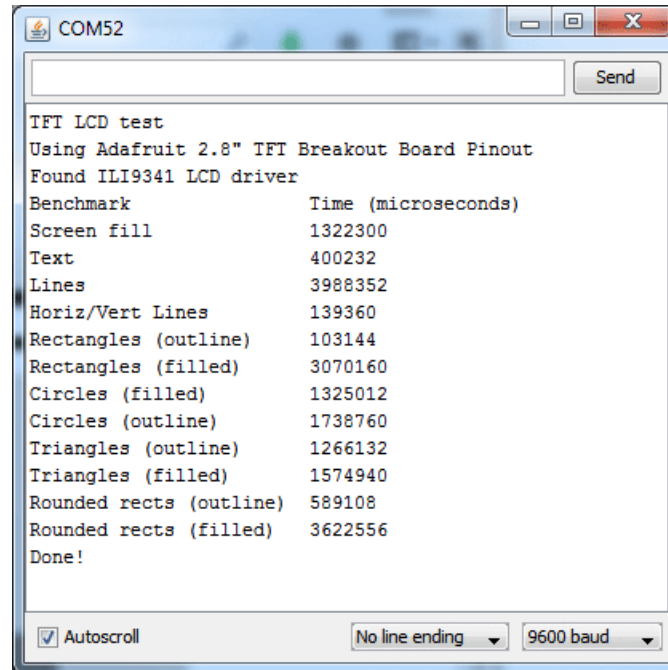


After restarting the Arduino software, you should see a new **example** folder called **Adafruit_TFTLCD** and inside, an example called **graphicstest**. Upload that sketch to your Arduino. You may need to press the Reset button to reset the arduino and TFT. You should see a collection of graphical tests draw out on the TFT.

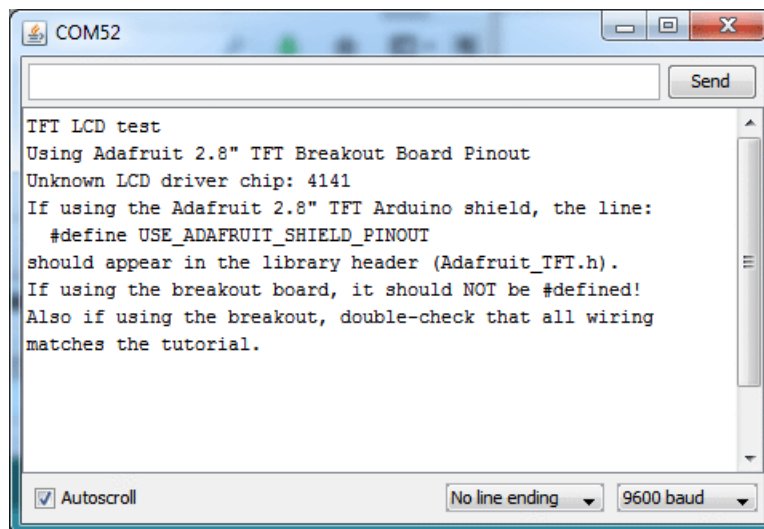
(The images below shows SPI wiring but the graphical output should be similar!)



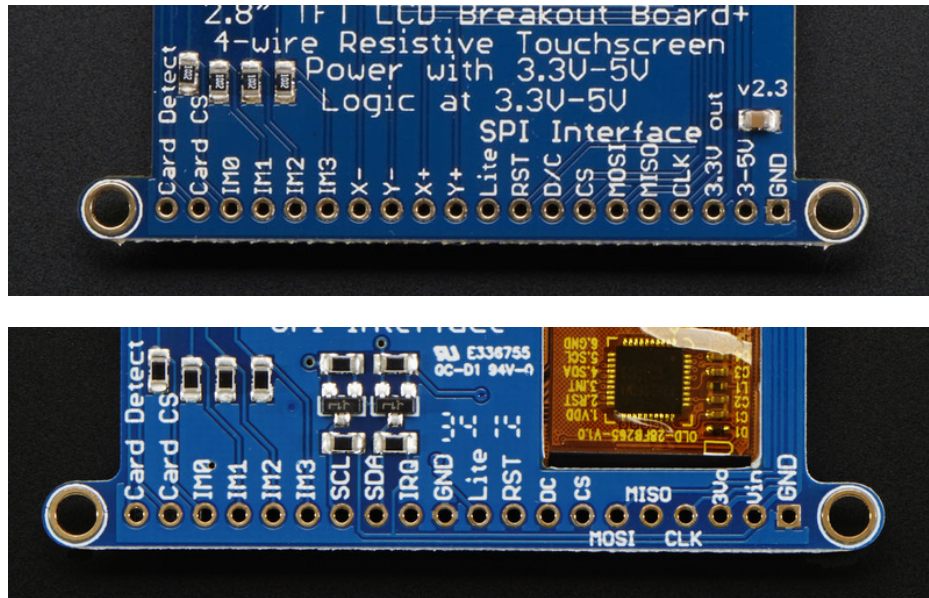
If you're having difficulties, check the serial console. The first thing the sketch does is read the driver code from the TFT. It should be **0x9341** (for the **ILI9341** controller inside)



If you **Unknown Driver Chip** then it's probably something with your wiring, double check and try again!



SPI Wiring and Test

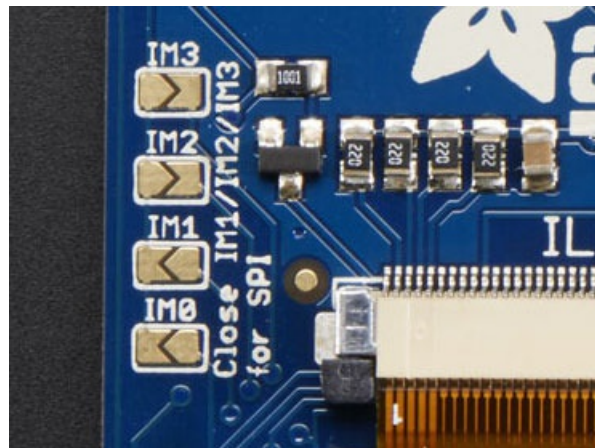


We show the 2.8" version of this breakout in the photos below but the 3.2" TFT is identical, just a lil bit bigger

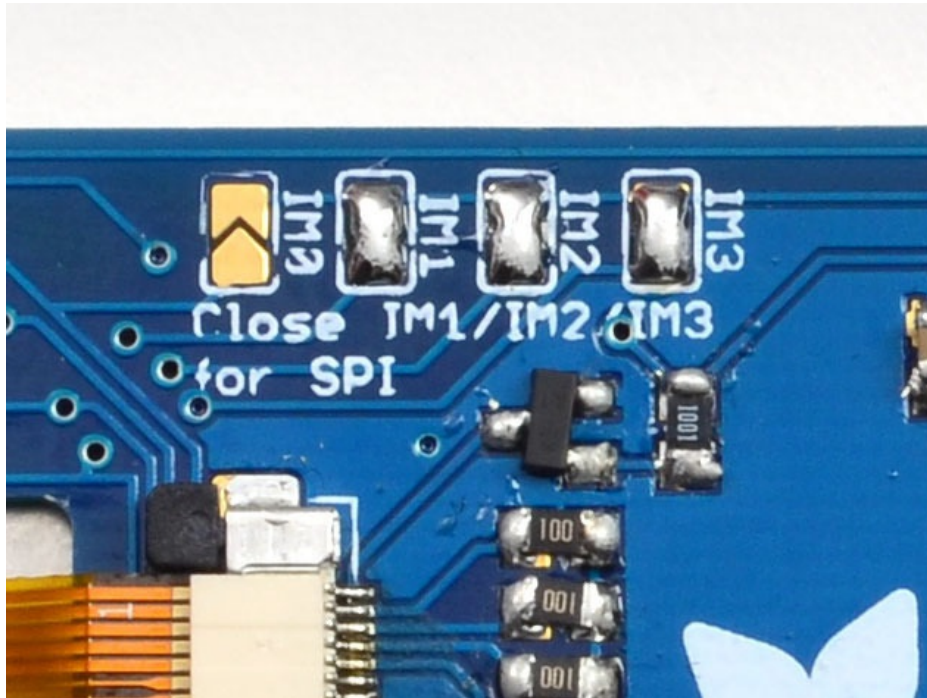
Don't forget, we're using the SPI interface side of the PCB!

SPI Mode Jumpers

Before you start, we'll need to tell the display to put us in SPI mode so it will know which pins to listen to. To do that, we have to connect the **IM1**, **IM2** and **IM3** pins to 3.3V. The easiest way to do that is to solder closed the **IMx** jumpers on the back of the PCB. Turn over the PCB and find the solder jumpers



With your soldering iron, melt solder to close the three jumpers indicated **IM1** **IM2** and **IM3** (do not solder closed **IM0**!)



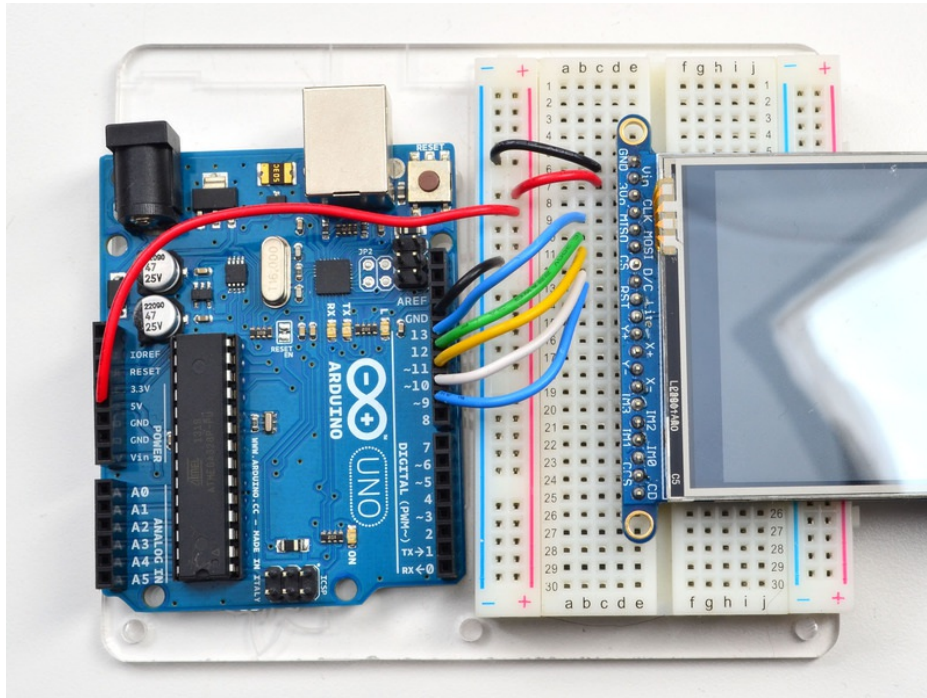
If you really don't want to solder them, you can also wire the breakout pins to the **3vo** pin, just make sure you don't tie them to 5V by accident! For that reason, we suggest going with the solder-jumper route.

Wiring

Wiring up the display in SPI mode is much easier than 8-bit mode since there's way fewer wires. Start by connecting the power pins

- **3-5V Vin** connects to the Arduino **5V** pin
- **GND** connects to Arduino ground
- **CLK** connects to SPI clock. On Arduino Uno/Duemilanove/328-based, that's **Digital 13**. On Mega's, it's **Digital 52** and on Leonardo/Due it's **ICSP-3** ([See SPI Connections for more details \(https://adafru.it/d5h\)](https://adafru.it/d5h))
- **MISO** connects to SPI MISO. On Arduino Uno/Duemilanove/328-based, that's **Digital 12**. On Mega's, it's **Digital 50** and on Leonardo/Due it's **ICSP-1** ([See SPI Connections for more details \(https://adafru.it/d5h\)](https://adafru.it/d5h))
- **MOSI** connects to SPI MOSI. On Arduino Uno/Duemilanove/328-based, that's **Digital 11**. On Mega's, it's **Digital 51** and on Leonardo/Due it's **ICSP-4** ([See SPI Connections for more details \(https://adafru.it/d5h\)](https://adafru.it/d5h))
- **CS** connects to our SPI Chip Select pin. We'll be using **Digital 10** but you can later change this to any pin
- **D/C** connects to our SPI data/command select pin. We'll be using **Digital 9** but you can later change this pin too.

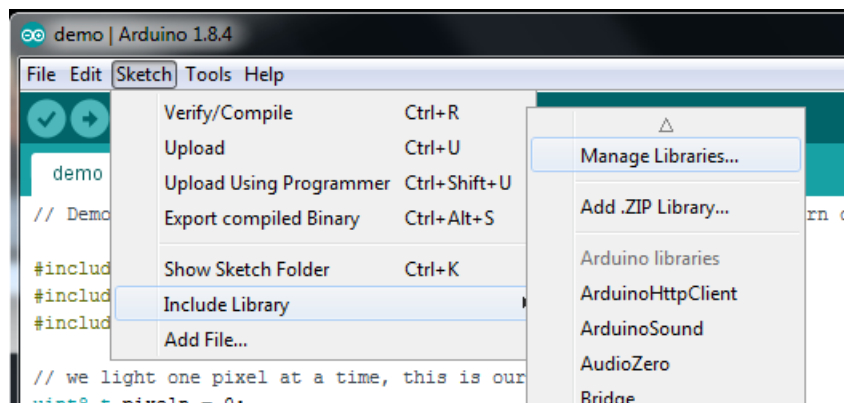
That's it! You do not need to connect the **RST** or other pins for now.



Install Libraries

You'll need a few libraries to use this display

From within the Arduino IDE, open up the Library Manager...

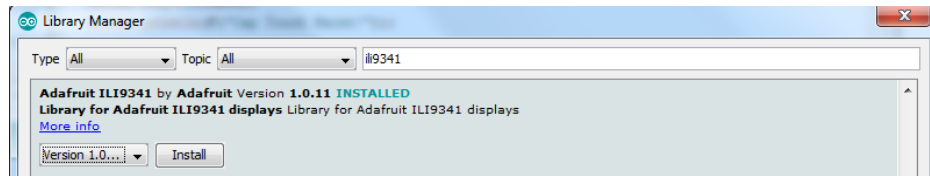


Install Adafruit ILI9341 TFT Library

We have example code ready to go for use with these TFTs.

Two libraries need to be downloaded and installed: first is the [Adafruit ILI9341 library \(https://adafru.it/d4d\)](https://adafru.it/d4d) (this contains the low-level code specific to this device), and second is the [Adafruit GFX Library \(https://adafru.it/aJa\)](https://adafru.it/aJa) (which handles graphics operations common to many displays we carry). If you have **Adafruit_GFX** already, make sure its the most recent version since we've made updates for better performance

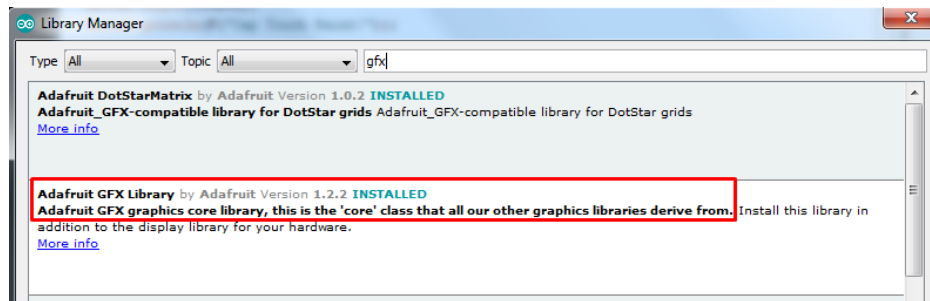
Search for **ILI9341** and install the **Adafruit ILI9341** library that pops up!



For more details, especially for first-time library installers, [check out our great tutorial at http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use](http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use) (<https://adafru.it/aYM>)

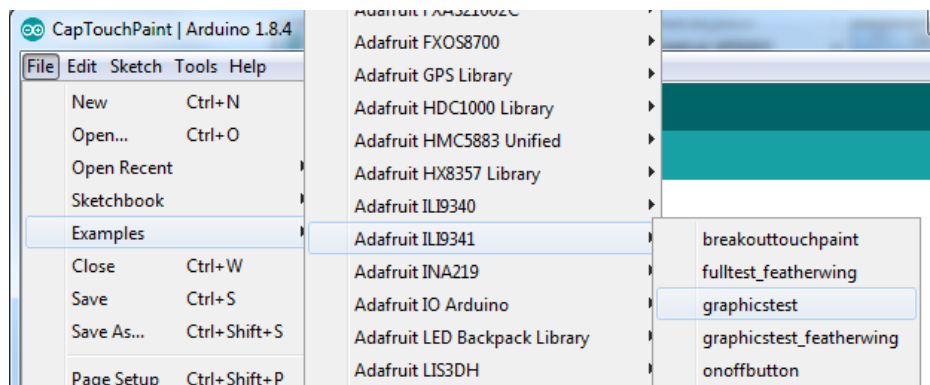
Next up, search for **Adafruit GFX** and locate the core library. A lot of libraries may pop up because we reference it in the description so just make sure you see **Adafruit GFX Library** in bold at the top.

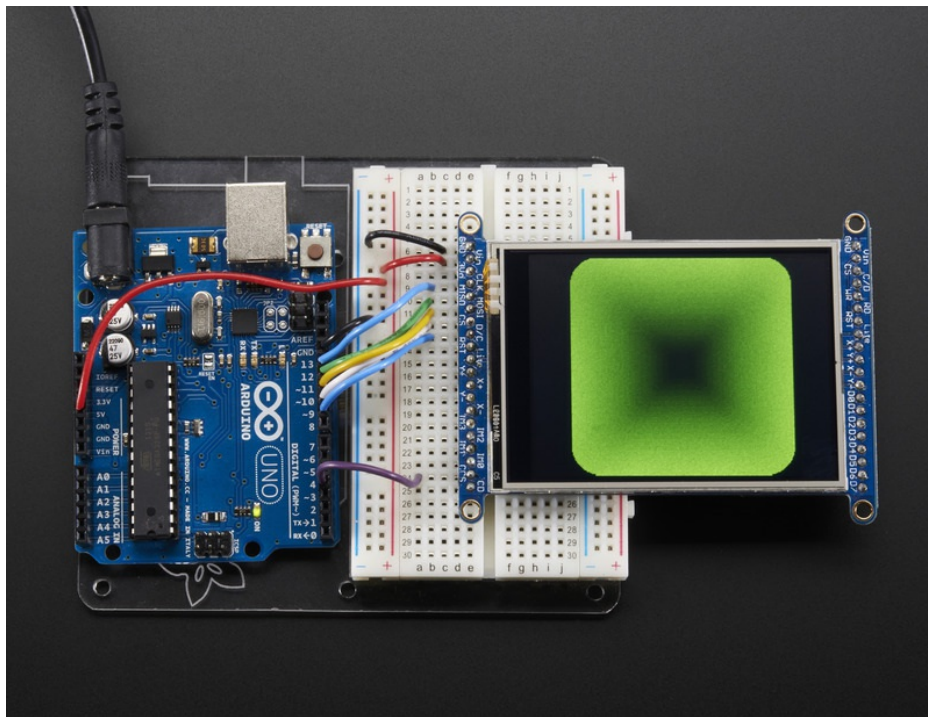
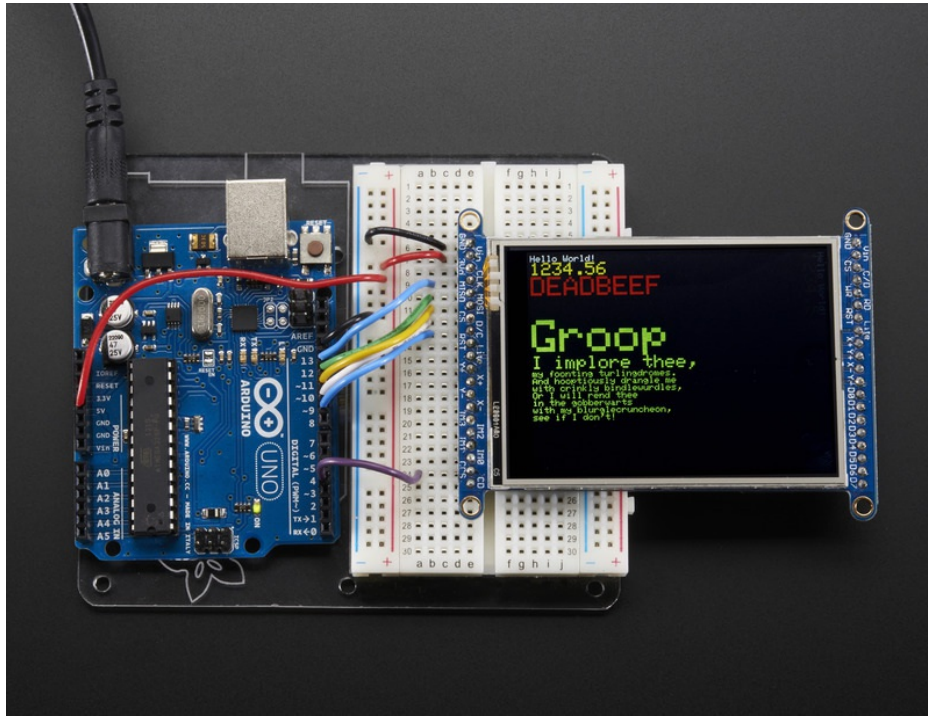
Install it!



Repeat the steps once more, this time looking for the **Adafruit_ZeroDMA** library. Install that one too!

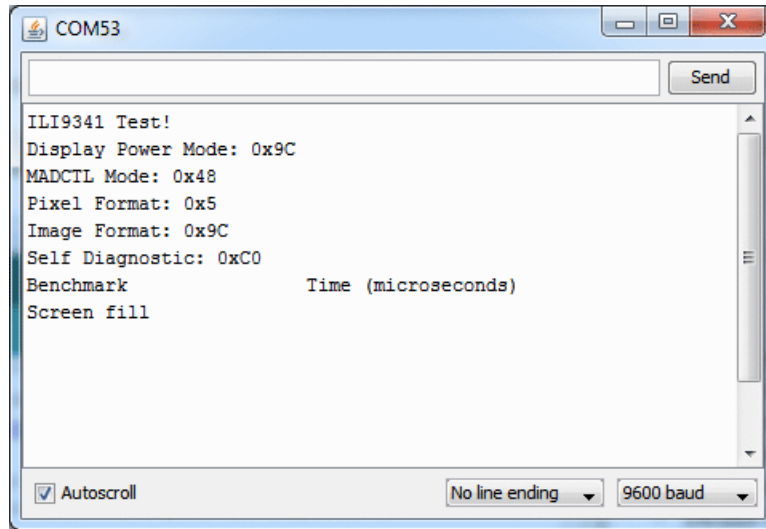
After restarting the Arduino software, you should see a new **example** folder called **Adafruit_ILI9341** and inside, an example called **graphicstest**. Upload that sketch to your Arduino. You may need to press the Reset button to reset the arduino and TFT. You should see a collection of graphical tests draw out on the TFT.





If you're having difficulties, check the serial console. The first thing the sketch does is read the driver configuration from the TFT, you should see the same numbers as below

If you did not connect up the MISO line to the TFT, you won't see the read configuration bytes so please make sure you connect up the MISO line for easy debugging! Once it's all working, you can remove the MISO line



Bitmaps (SPI Mode)

There is a built in microSD card slot into the breakout, and we can use that to load bitmap images! You will need a microSD card formatted **FAT16** or **FAT32** (they almost always are by default).

Its really easy to draw bitmaps. **However, this is only supported when talking to the display in SPI mode, not 8-bit mode!**

Lets start by downloading this image of pretty flowers (pix by johngineer)



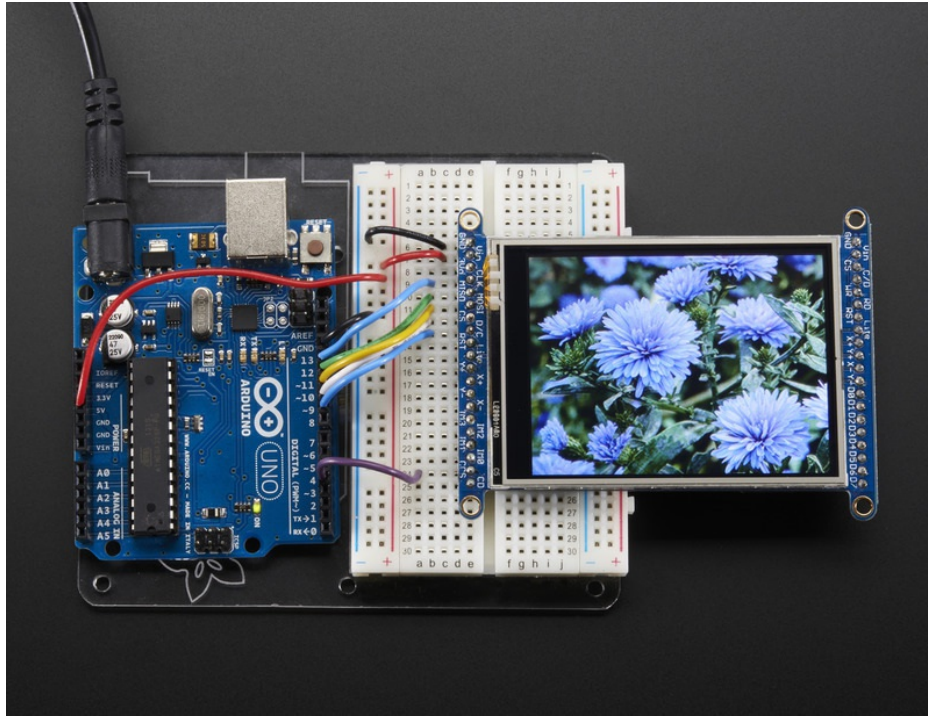
Copy **purple.bmp** into the base directory of a microSD card and insert it into the microSD socket in the breakout.

You'll need to connect up the **CCS** pin to **Digital 4** on your Arduino as well. In the below image, its the extra purple wire

You may want to try the **SD library** examples before continuing, especially one that lists all the files on the SD card

Now upload the **file->examples->Adafruit_ILI9341->spitftbitmap** example to your Arduino + breakout. You will see the flowers appear!

We show the 2.8" version of this breakout in the photos below but the 3.2" TFT is identical, just a lil bit bigger



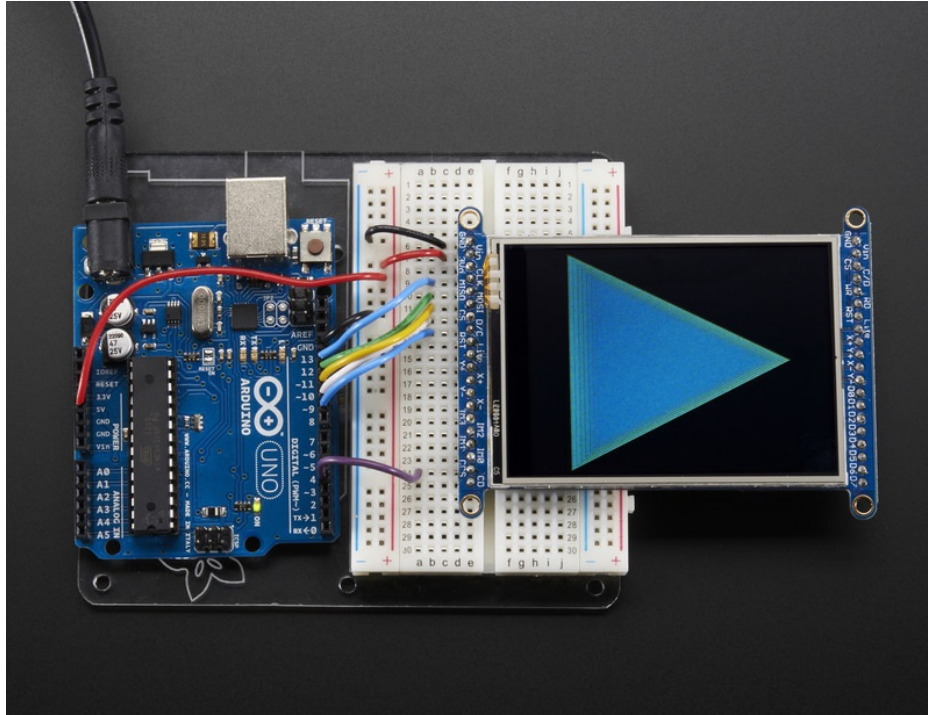
To make new bitmaps, make sure they are less than 240 by 320 pixels and save them in **24-bit BMP format**! They must be in 24-bit format, even if they are not 24-bit color as that is the easiest format for the Arduino. You can rotate images using the `setRotation()` procedure

You can draw as many images as you want - dont forget the names must be less than 8 characters long. Just copy the BMP drawing routines below `loop()` and call

```
bmpDraw(bmpfilename, x, y);
```

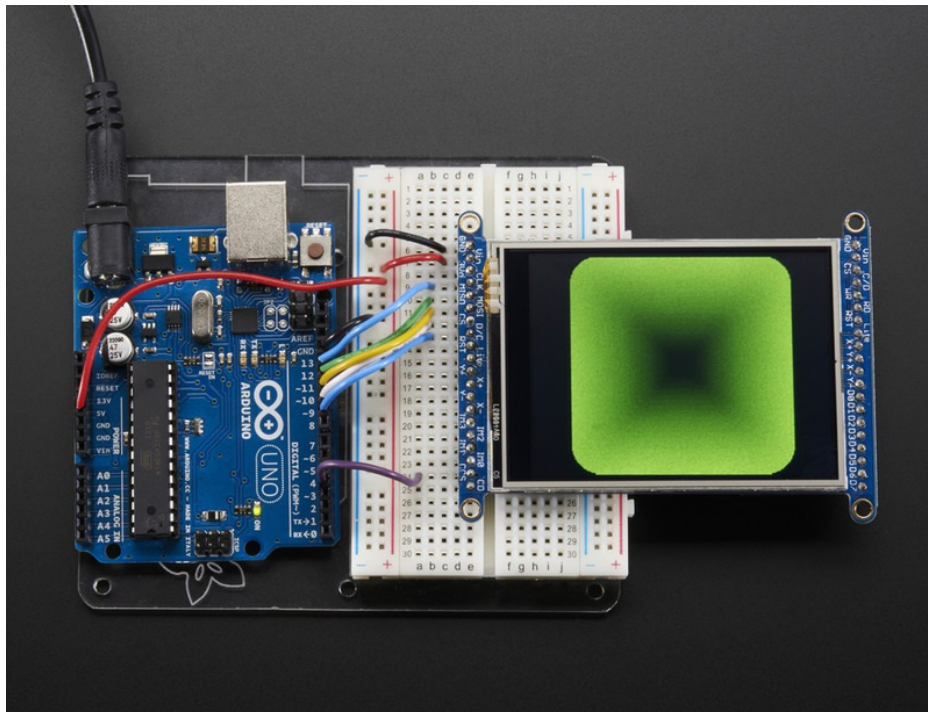
For each bitmap. They can be smaller than 320x240 and placed in any location on the screen.

Adafruit GFX library



The Adafruit_GFX library for Arduino provides a common syntax and set of graphics functions for all of our TFT, LCD and OLED displays. This allows Arduino sketches to easily be adapted between display types with minimal fuss...and any new features, performance improvements and bug fixes will immediately apply across our complete offering of color displays.

The GFX library is what lets you draw points, lines, rectangles, round-rects, triangles, text, etc.



Check out our detailed tutorial here <http://learn.adafruit.com/adafruit-gfx-graphics-library> (<https://adafru.it/aPx>)

It covers the latest and greatest of the GFX library. The GFX library is used in both 8-bit and SPI modes so the underlying commands (`drawLine()` for example) are identical!
(<https://adafru.it/aPx>)

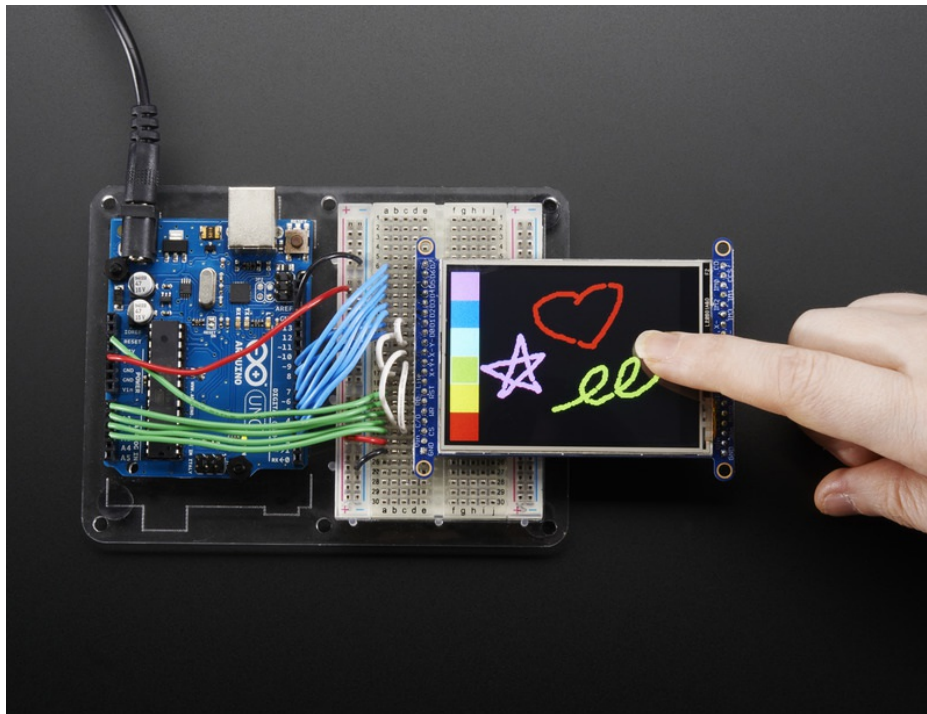
Resistive Touchscreen

The LCD has a 2.8" or 3.2" 4-wire resistive touch screen glued onto it. You can use this for detecting finger-presses, stylus', etc. You'll need 4 pins to talk to the touch panel, and at least 2 must be analog inputs. The touch screen is a completely separate part from the TFT, so be aware if you rotate the display or have the TFT off or reset, the touch screen doesn't "know" about it - its just a couple resistors!

We have a demo for the touchscreen + TFT that lets you 'paint' simple graphics. There's versions for both SPI and 8-bit mode and are included in the libraries. Just make sure you have gone thru the TFT test procedure already since this builds on that.

Remember, if you rotate the screen drawing with `setRotation()` you'll have to use `map()` or similar to flip around the X/Y coordinates for the touchscreen as well! It doesn't know about drawing rotation

We show the 2.8" version of this breakout in the photos below but the 3.2" TFT is identical, just a lil bit bigger



Download Library

Begin by grabbing our [analog/resistive touchscreen library from github \(https://adafru.it/aT1\)](https://adafru.it/aT1) (or just click the download button)

<https://adafru.it/dd0>

<https://adafru.it/dd0>

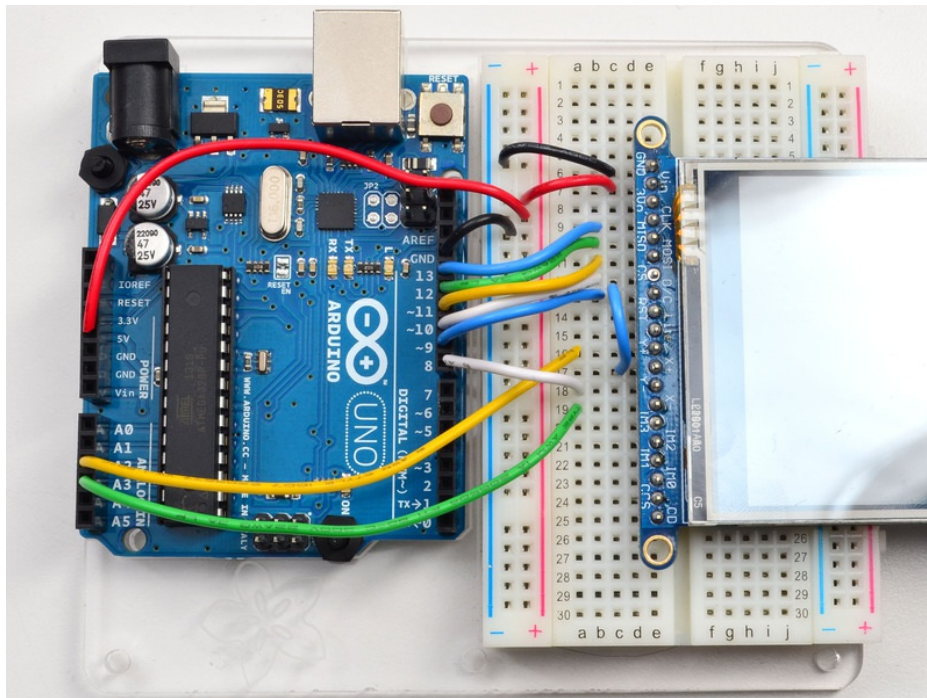
Uncompress the zip file and rename the folder **TouchScreen** (make sure it contains **Touchscreen.cpp** and **Touchscreen.h**) then install in your **libraries** folder just like you did for the TFT library

Touchscreen Paint (SPI mode)

An additional 4 pins are required for the touchscreen. For the two analog pins, we'll use **A2** and **A3**. For the other two connections, you can pin any two digital pins but we'll be using **D9** (shared with **D/C**) and **D8** since they are available. We can save the one pin by sharing with **D/C** but you can't share any other SPI pins. So basically you can get away with using only three additional pins.

Wire the additional 4 pins as follows:

- Y+ to Arduino **A2**
- X+ to Arduino **D9** (Same as **D/C**)
- Y- to Arduino **D8**
- X- to Arduino **A3**



Load up the **breakoutTouchPaint** example from the **Adafruit_ILI9341** library and try drawing with your fingernail! You can select colors by touching the 'palette' of colors on the right

Touchscreen Paint (8-Bit mode)

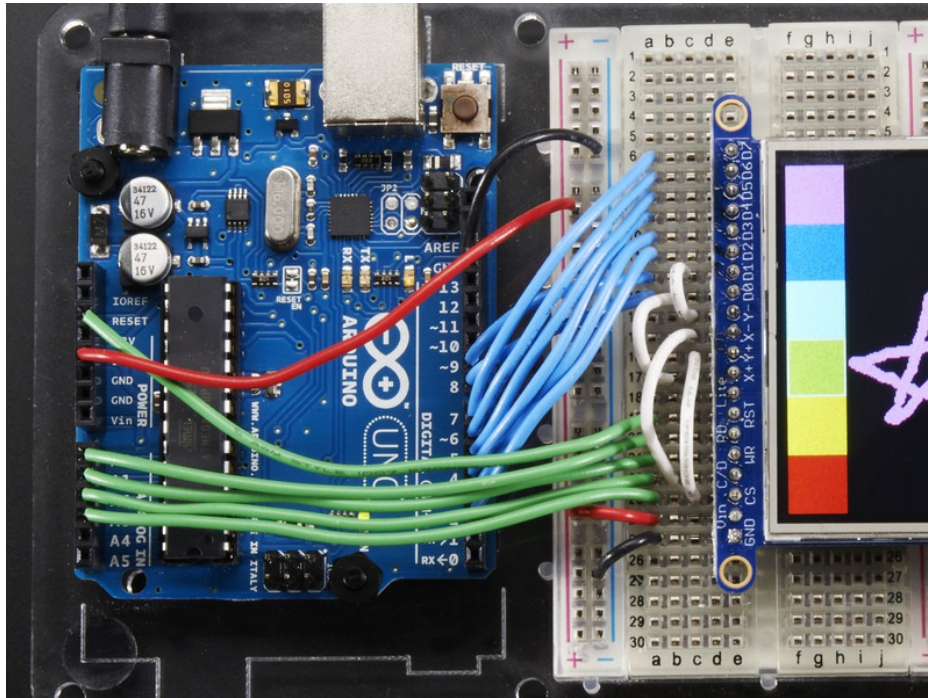
Another 4 pins seems like a lot since already 12 are taken up with the TFT **but** you can **reuse** some of the pins for the TFT LCD! This is because the resistance of the panel is high enough that it doesn't interfere with the digital input/output and we can query the panel in between TFT accesses, when the pins are not being used.

We'll be building on the wiring used in the previous drawing test for UNO

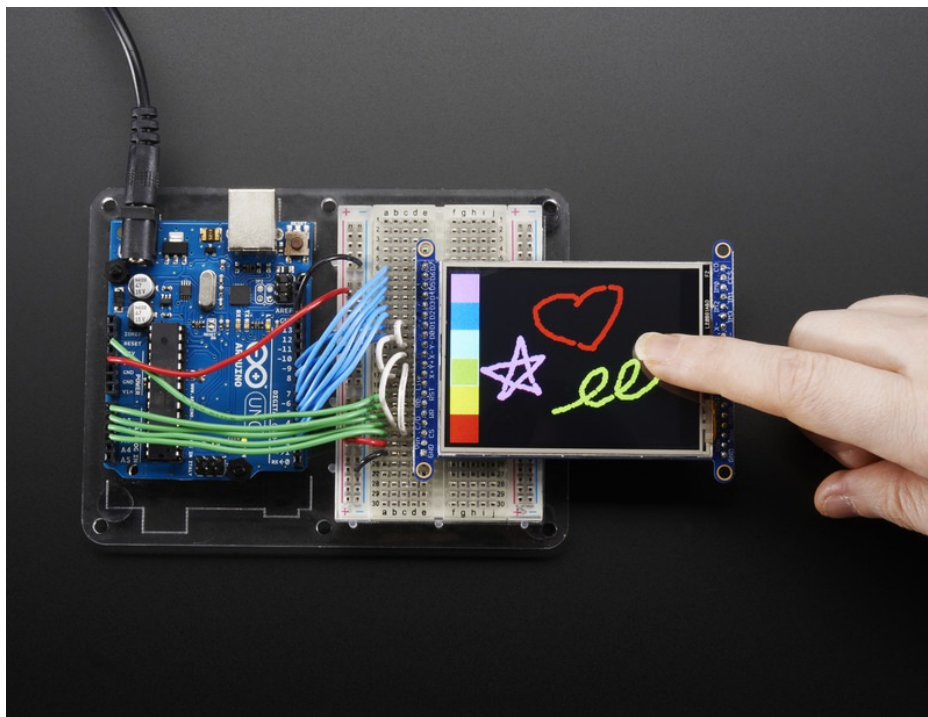
You can wire up the 4 touchscreen pins as follows. Starting from the top

- Y- connects to digital **#9** (also **D1**)
- The next one down (**X-**) connects to **Analog 2** (also **C/D**)
- The next one over (**Y+**) connects to **Analog 3** (also **CS**)
- The last one (**X+**) connects to digital **8**. (also **D0**)

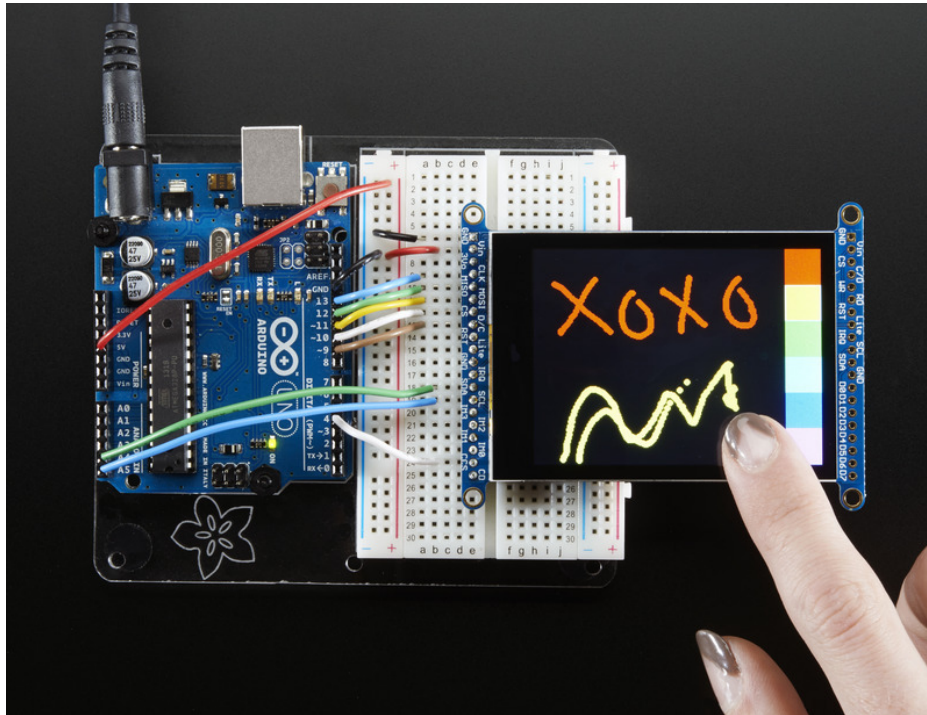
The X- and Y+ pins pretty much have to connect to those analog pins (or to analog 4/5) but Y-/X+ can connect to any digital or analog pins.



Load up the `tftpaint` example from the `Adafruit_TFTLCD` library and try drawing with your fingernail! You can select colors by touching the 'palette' of colors on the right



Capacitive Touchscreen



We now have a super-fancy capacitive touch screen version of this shield. Instead of a resistive controller that needs calibration and pressing down, the capacitive has a hard glass cover and can be used with a gentle fingertip. It is a single-touch capacitive screen only!

The capacitive touch screen controller communicates over I2C, which uses two hardware pins. However, you can share these pins with other sensors and displays as long as they don't conflict with I2C address 0x38.

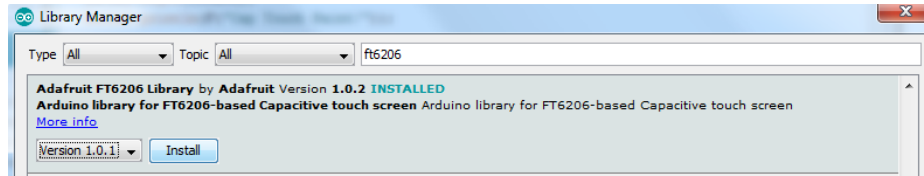
The capacitive touch chip shares the same power and ground as the display, the only new pins you must connect are **SDA** and **SCL** - these must connect to the Arduino I2C pins.

- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

This demo uses the SPI 'side' of the display so get the SPI drawing demos working before you continue! You can adapt the code for use with the 8-bit side, just instantiate the FT6206 library and see the reference below!

Download the FT6206 Library

To control the touchscreen you'll need one more library (<https://adafruit.it/dGG>) - the FT6206 controller library which does all the low level chatting with the FT6206 driver chip. Use the library manager and search for **FT6206** and select the Adafruit FT6206 library:



Once you have the library installed, restart the IDE. Now from the **examples->Adafruit_FT6206** menu select **CapTouchPaint** and upload it to your Arduino.

The touch screen is made of a thin glass sheet, and its very fragile - a small crack or break will make the entire touch screen unusable. Don't drop or roughly handle the TFT and be especially careful of the corners and edges. When pressing on the touchscreen, remember you cannot use a fingernail, it must be a fingerpad. Do not press harder and harder until the screen cracks!

FT6206 Library Reference

Getting data from the touchscreen is fairly straight forward. Start by creating the touchscreen object with

```
Adafruit_FT6206 ts = Adafruit_FT6206();
```

We're using hardware I2C which is fixed in hardware so no pins are defined. Then you can start the touchscreen with

```
ts.begin()
```

Check to make sure this returns a True value, which means the driver was found. You can also call **begin(threshvalue)** with a number from 0-255 to set the touch threshold. The default works pretty well but if you're having too much sensitivity (or not enough) you can try tweaking it

Now you can call

```
if (ts.touched())
```

to check if the display is being touched, if so call:

```
TS_Point p = ts.getPoint();
```

To get the touch point from the controller. `TS_Point` has `.x` and `.y` data points. The `x` and `y` points range from 0 to 240 and 0 to 320 respectively. This corresponds to each pixel on the display. The FT6206 does not need to be 'calibrated' but it also doesn't know about rotation. **So if you want to rotate the screen you'll need to manually rotate the x/y points!**

Touchscreen Interrupt pin

Advanced users may want to get an interrupt on a pin (or even, just test a pin rather than do a full SPI query) when the touchscreen is pressed. That's the IRQ pin, which is a 3V logic output from the breakout, you can connect it to any interrupt pin and use it like a 'button press' interrupt. We find that querying/polling the chip is fast enough for most

beginner Arduino projects!

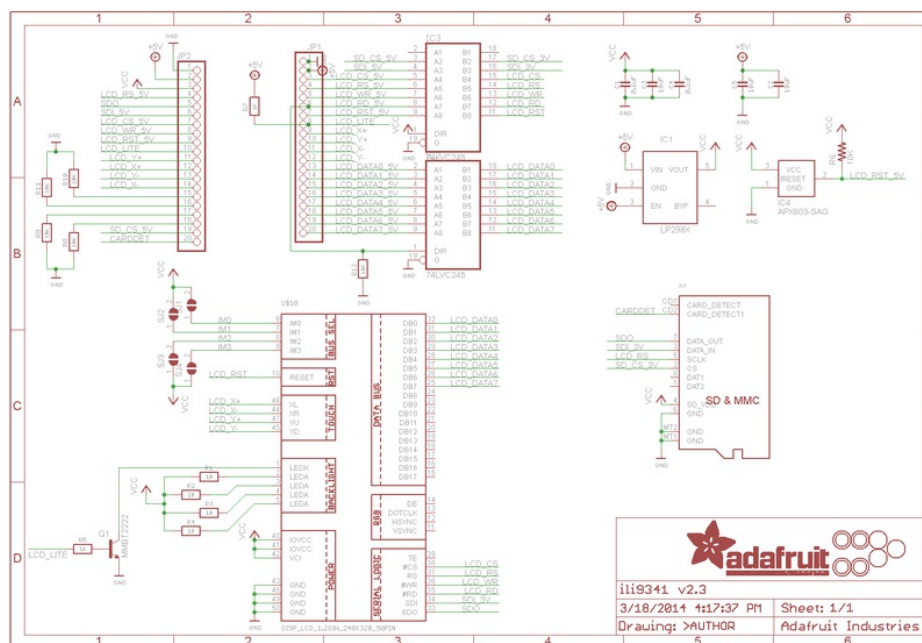
FT6206 Library Reference

[FT6206 Library Reference \(https://adafru.it/Atz\)](https://adafru.it/Atz)

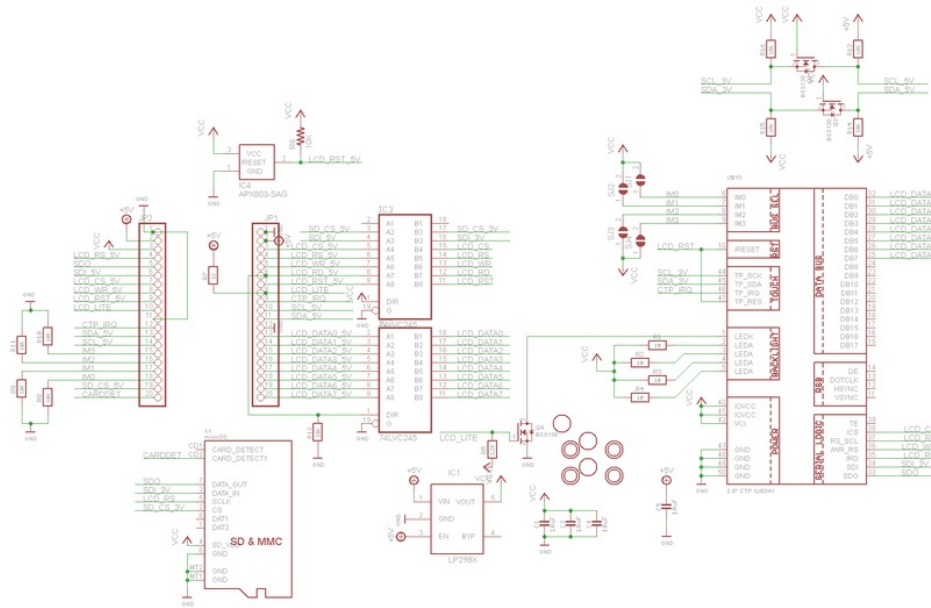
Datasheets & Files

- [ILI9341 TFT controller chip datasheet \(https://adafru.it/d4I\)](https://adafru.it/d4I) (this is what you want to refer to if porting or if you want to look at the TFT command set)
- [Raw 2.8" Resistive TFT datasheet \(https://adafru.it/sEt\)](https://adafru.it/sEt)
- [Raw 2.8" Capacitive TFT datasheet \(https://adafru.it/rwA\)](https://adafru.it/rwA)
- [FT6206 Datasheet \(https://adafru.it/sEu\)](https://adafru.it/sEu) & [App note \(https://adafru.it/dRn\)](https://adafru.it/dRn) (capacitive chip)
- [Fritzing objects in Adafruit Fritzing Library \(https://adafru.it/c7M\)](https://adafru.it/c7M)
- [2.8" TFT with Capacitive Touch EagleCAD files \(https://adafru.it/pAr\)](https://adafru.it/pAr)
- [2.8" TFT with Resistive Touch EagleCAD files \(https://adafru.it/pAs\)](https://adafru.it/pAs)

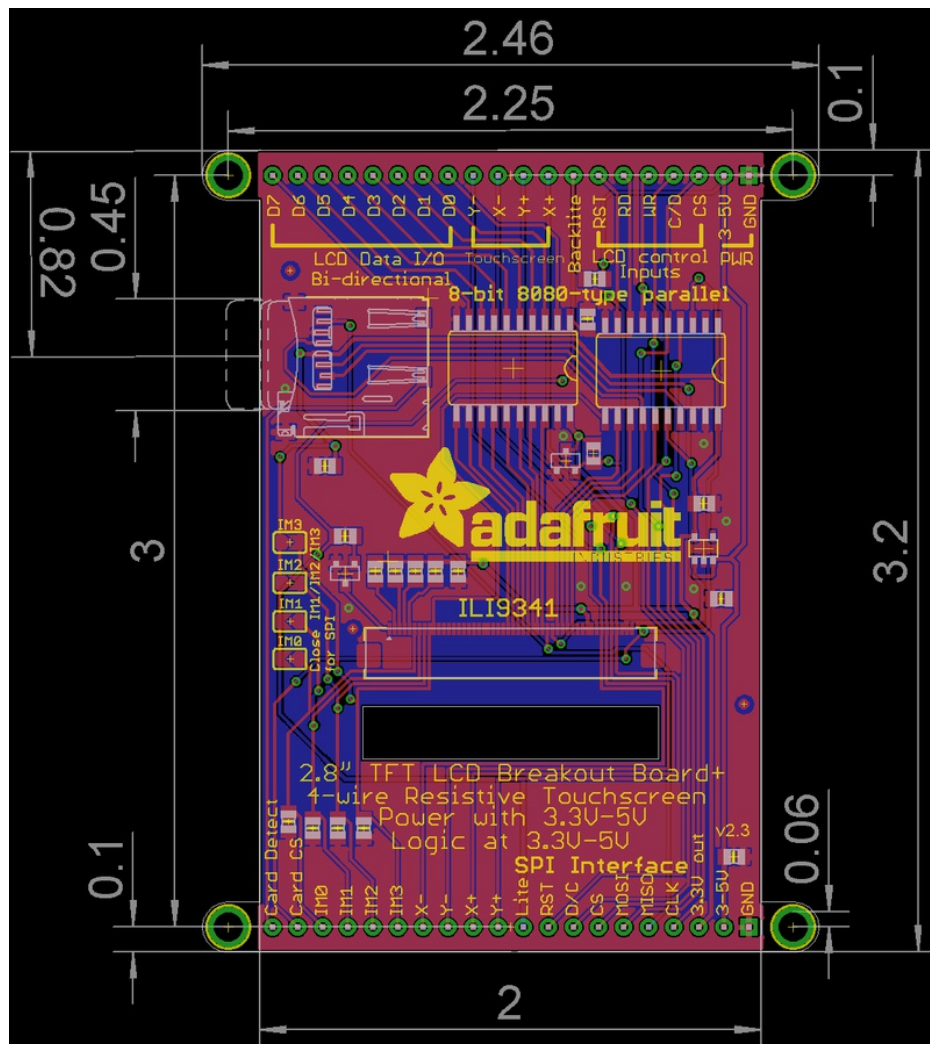
2.8" and 3.2" Resistive Touch Schematic



Capacitive Touch Schematic



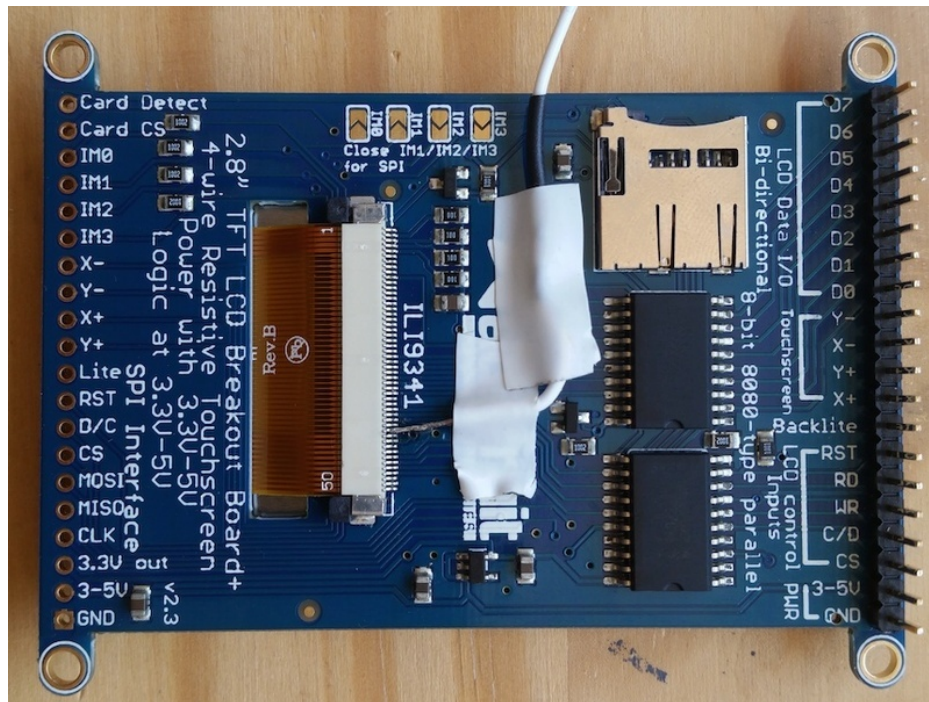
2.8" TFT Layout Diagram



F.A.Q.

If I drive this display at very high speeds I get 'video tearing' effects, how can I synchronize the display refreshes?

We don't break out the TE (tearing effect line) because we use these with small microcontrollers, but if you *do* need to synchronize you can solder to the TE pad on the TFT using fine silicone wire. ([See this forum thread](#))



Display does not work on initial power but does work after a reset.

The display driver circuit needs a small amount of time to be ready after initial power. If your code tries to write to the display too soon, it may not be ready. It will work on reset since that typically does not cycle power. If you are having this issue, try adding a small amount of delay before trying to write to the display.

In Arduino, use `delay()` to add a few milliseconds before calling `tft.begin()`. Adjust the amount of delay as needed to see how little you can get away with for your specific setup.