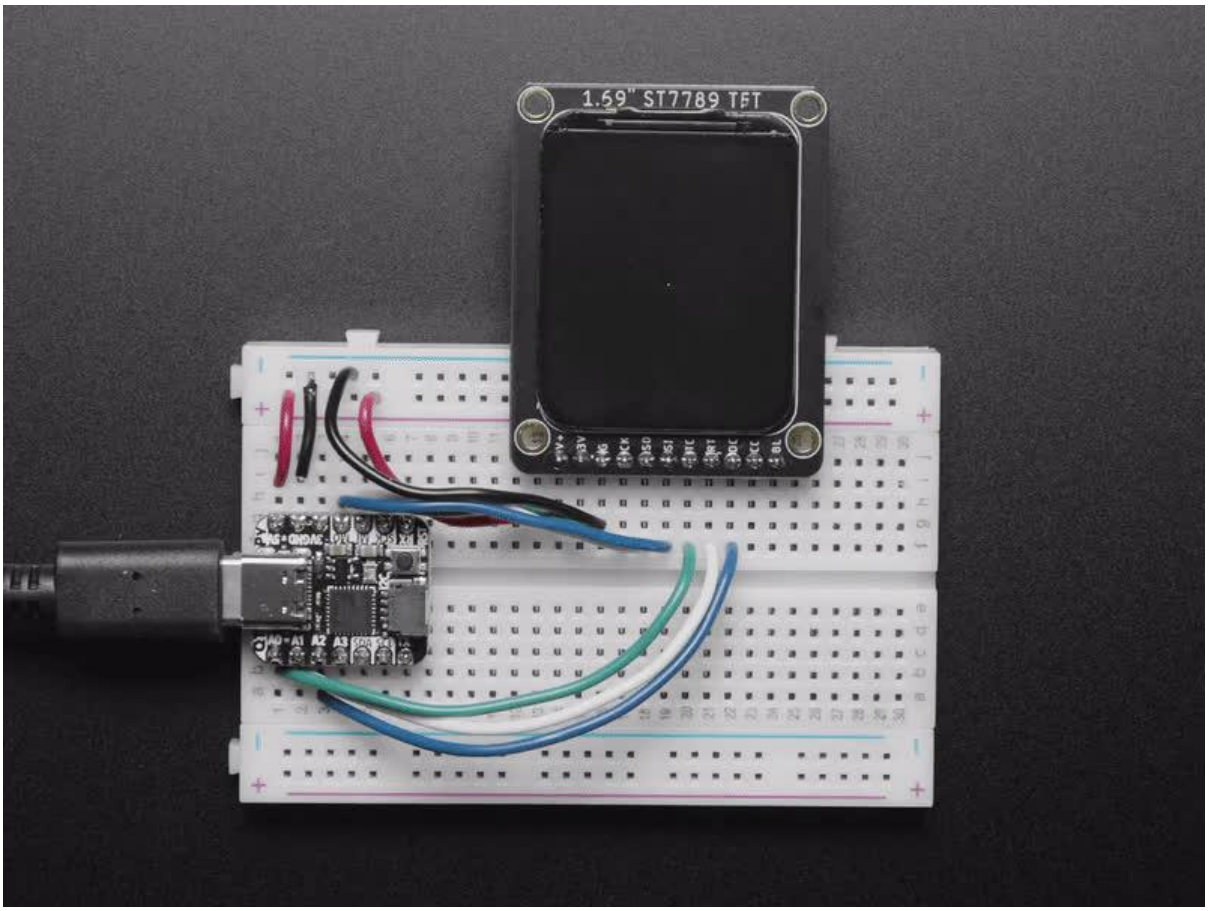
# Adafruit 1.69" 280x240 Round Rectangle Color IPS TFT Display

Created by Melissa LeBlanc-Williams

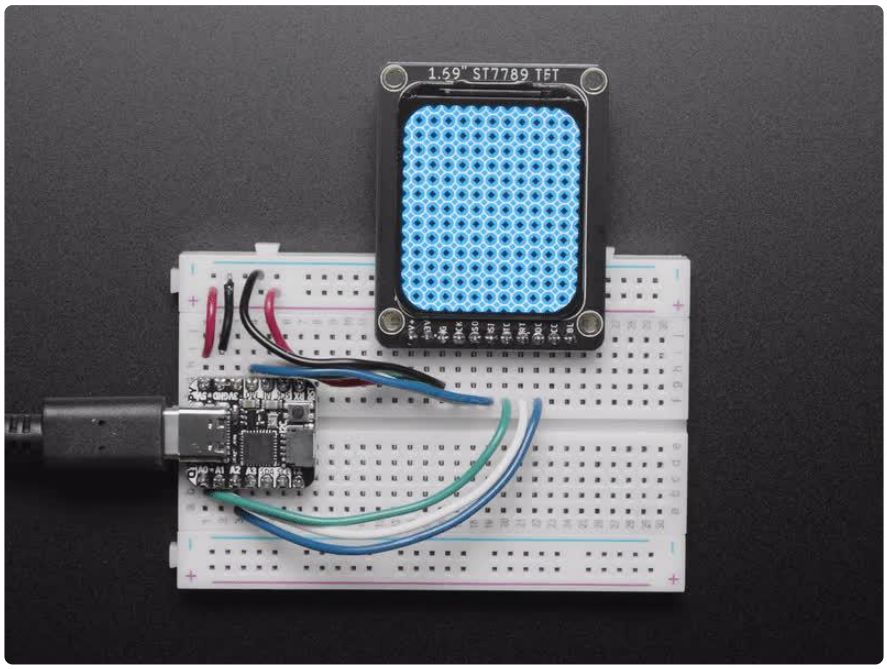



https://learn.adafruit.com/adafruit-1-69-280x240-round-rectangle-color-ips-tft-display

Last updated on 2022-12-01 02:16:25 PM EST

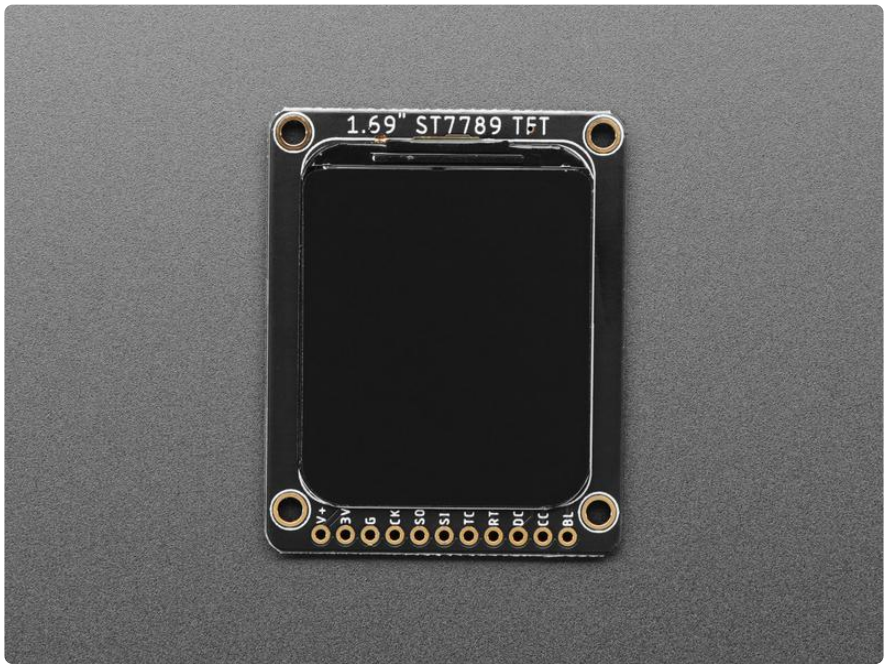
# Table of Contents

# Overview

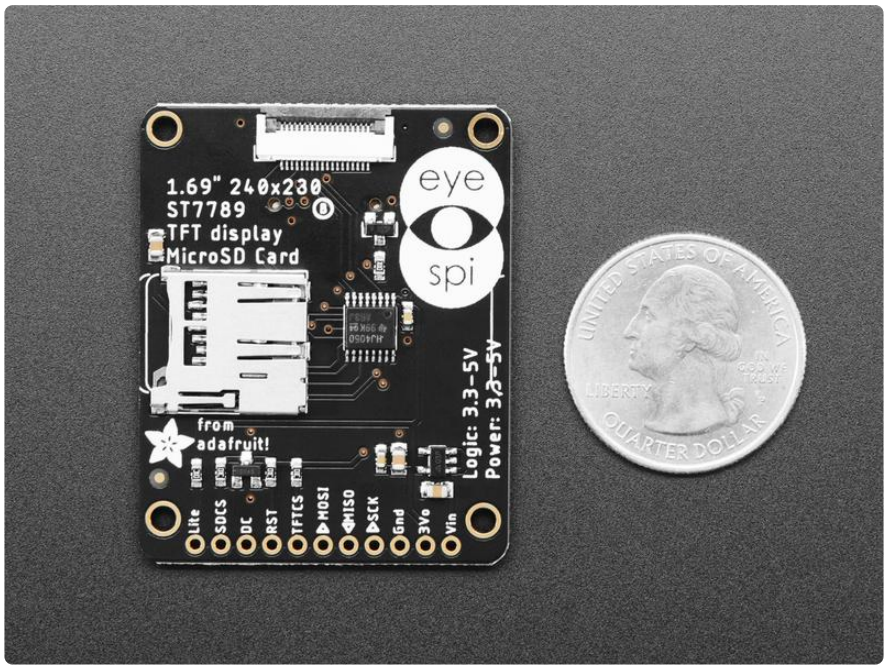



Don't be such a square - throw a curve-ball into your electronics with a curved-edge miniature display. Here's a new "round rect" TFT display - it's 1.69" diagonal and has a high-density 220 ppi, 280x240 full color pixels with IPS any-angle viewing.

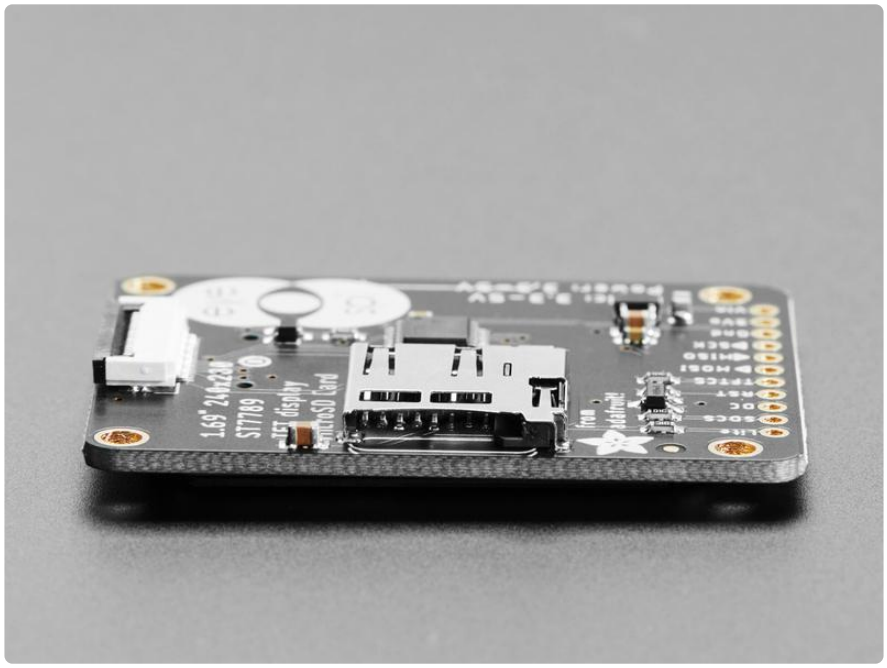
We've seen displays of this caliber used in smartwatches and small electronic devices but they've always been MIPI interface. Finally, we found one that is SPI and has a friendly display driver, so it works with any and all microcontrollers or microcomputers!



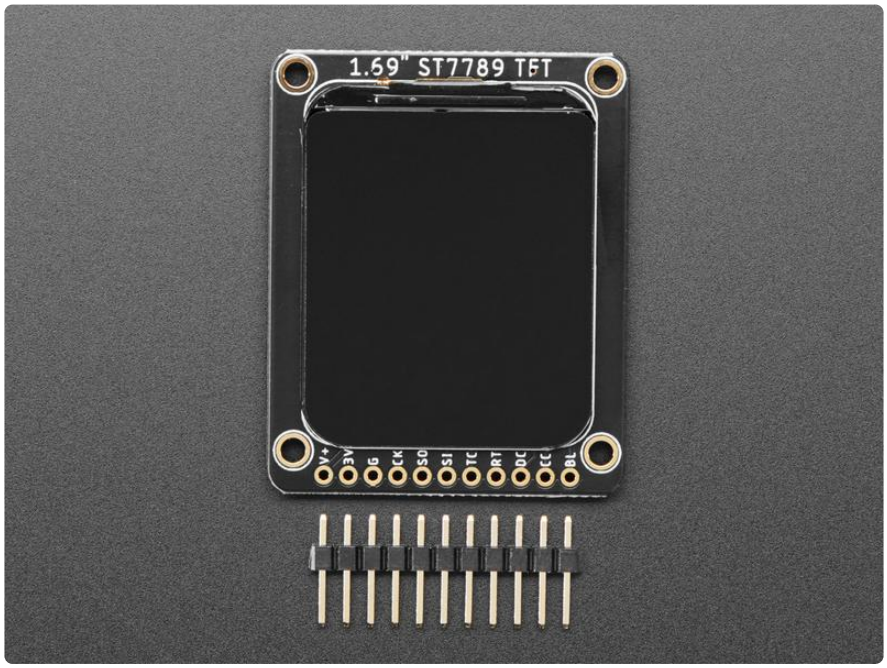
This lovely little display breakout is the best way to add a small, colorful, and very bright display to any project. Since the display uses 4-wire SPI to communicate and has its own pixel-addressable frame buffer, it can be used with every kind of microcontroller. Even a very small one with low memory and few pins available! The 1.69" display has 280x240 16-bit full color pixels and is an IPS display, so the color looks great up to 80 degrees off-axis in any direction. The TFT driver (ST7789) is very similar to the popular ST7735, and our Arduino library supports it well.



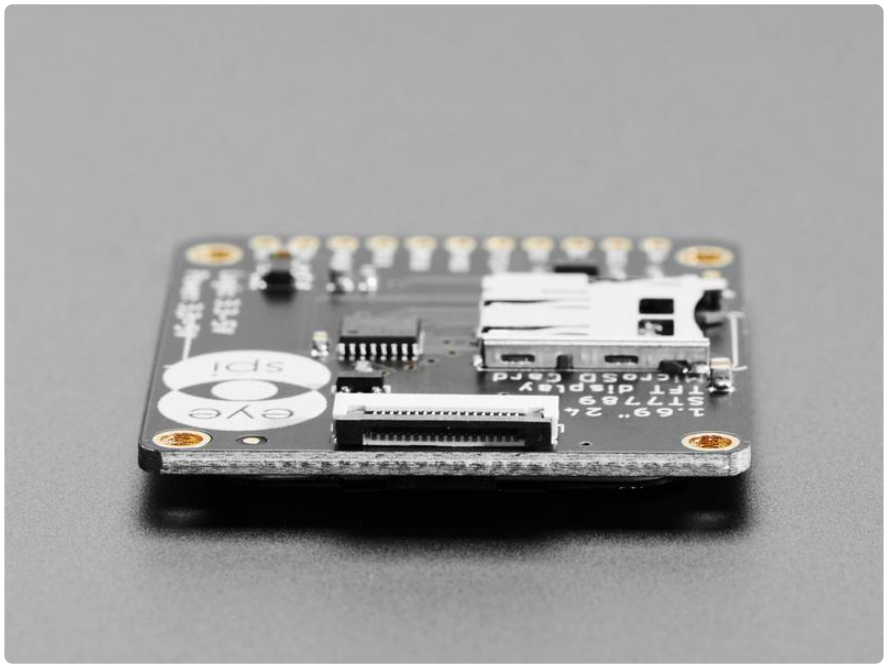

Note that the way we get the rounded corners is by deleting pixels. The corner pixels are still addressed in RAM, they just don't appear, so it isn't like you have to do some special radial-pixel mapping. Treat it like a rectangular display.



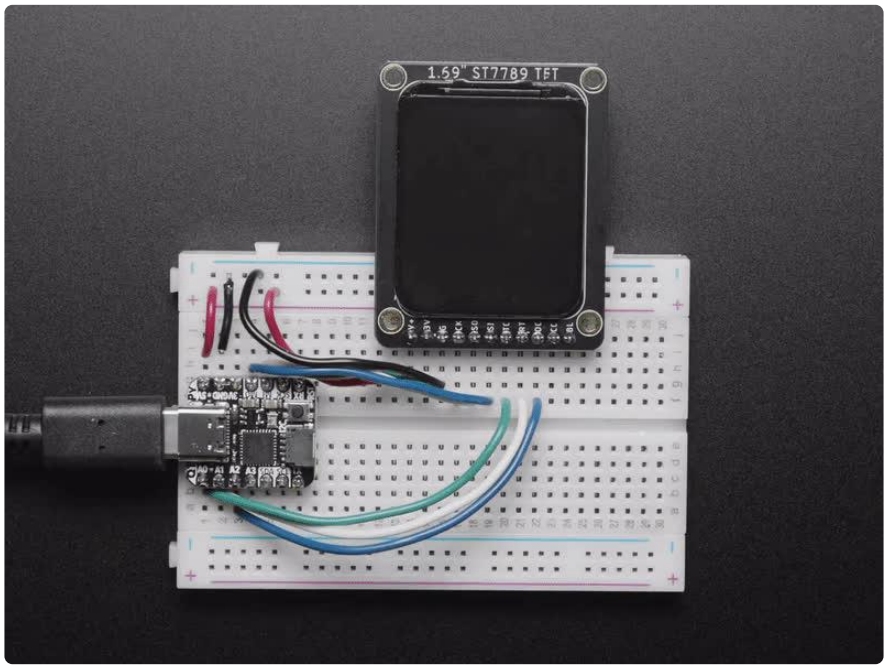
Our breakout has the TFT display soldered on (it uses a delicate flex-circuit connector) as well as an ultra-low-dropout 3.3V regulator and a 3/5V level shifter so you can use it with 3.3V or 5V power and logic. We also had a little space so we placed a microSD card holder so you can easily load full-color bitmaps from a FAT16/ FAT32 formatted microSD card. The microSD card is not included, but you can pick one up here (http://adafru.it/102).



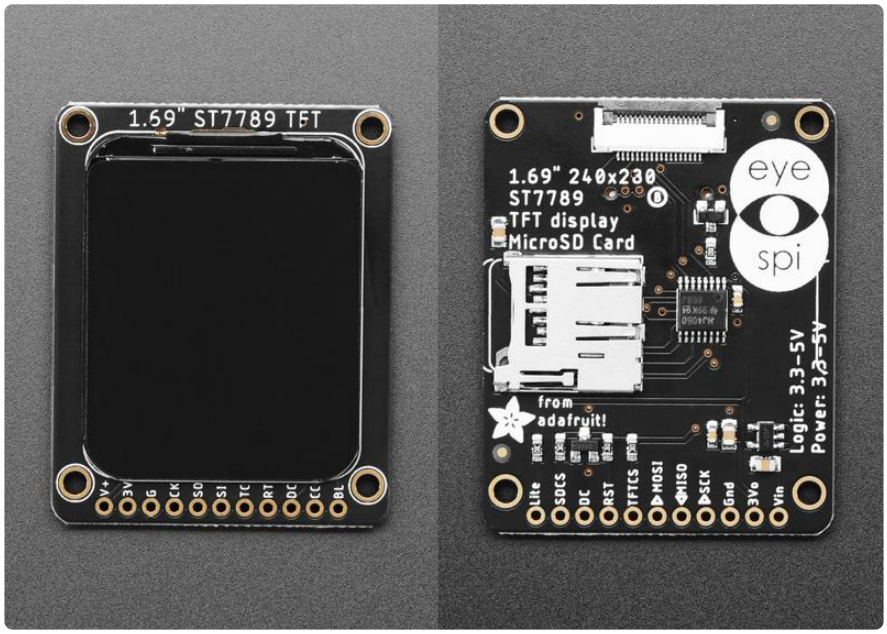

Of course, we wouldn't just leave you with a datasheet and a "good luck!" - we've written a full open-source graphics library that can draw pixels, lines, rectangles, circles, text, and bitmaps as well as example code and a wiring tutorial (). The code is written for Arduino but can be easily ported to your favorite microcontroller!



This display breakout also features a 18-pin "EYE SPI" standard FPC connector with flip-top connector. You can use a 18-pin 0.5mm pitch FPC cable to connect to all the GPIO pins, for when you want to skip the soldering.




# Pinouts




## Power Pins

- V+ / VIN - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 3V microcontroller like a Feather M4, use 3V, or for a 5V microcontroller like Arduino, use 5V.

- 3V / 3Vo - This is the output from the onboard 3.3V regulator. If you have a need for a clean 3.3V output, you can use this! It can provide at least 100mA output.
- G / Gnd - This is common ground for power and logic.

## SPI Pins

- CK / SCK - this is the SPI Clock pin. Use 3-5V logic level.
- SI / MISO - this is the Serial Data Out / Microcontroller In Sensor Out pin. It is used for the SD card. It isn't used for the TFT display which is write-only. It is 3.3V logic out (but can be read by 5V logic)
- SO / MOSI - this is the Serial Data In / Microcontroller Out Sensor In pin. It is used to send data from the microcontroller to the SD card and/or TFT. Use 3-5V logic level
- TC / TFTCS - this is the TFT Chip Select pin. Use 3-5V logic level

## Other Pins

- RT / RST - This is the TFT reset pin. Connect to ground to reset the TFT! It's best to have this pin controlled by the library so the display is reset cleanly, but you can also connect it to the Arduino Reset pin, which works for most cases. There is an automatic-reset chip connected so it will reset on power-up. Use 3-5V logic level
- DC - This is the TFT SPI data or command selector pin. Use 3-5V logic level
- CC / SDCS - This is the SD card chip select pin, used if you want to read from the SD card. Use 3-5V logic level
- BL / 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. Use 3-5V logic level

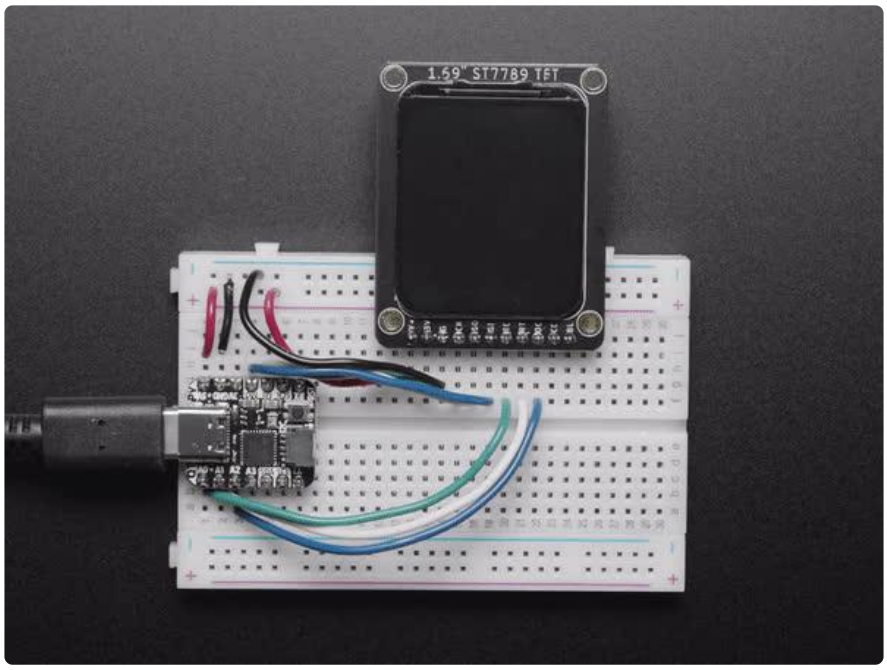
## EYE SPI 18-pin FPC Connector

1. VIN (3 to 5V DC power)
2. Backlight (3~5V logic PWM optional input)
3. Ground
4. SPI Clock (3~5V logic in)
5. SPI MOSI (3~5V logic Microcontroller Out, Screen/SD In)
6. SPI MISO (3~5V logic Microcontroller In, Screen/SD Out)
7. TFT Data/Command (3~5V logic in)
8. TFT Reset (optional 3~5V logic in)
9. TFT SPI Chip Select (3~5V logic in)

10. SD Card SPI Chip Select (3~5V logic in)
11. Unused
12. Unused
13. Unused
14. Unused
15. Unused
16. Unused
17. Unused
18. Unused

---

# Arduino Wiring & Test



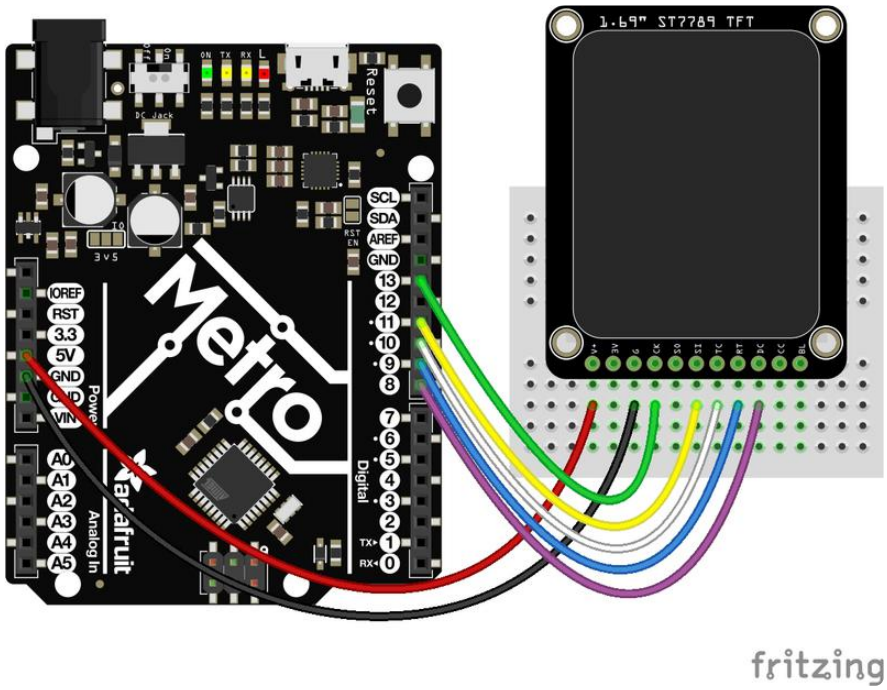

# Basic Graphics Test Wiring

Wiring up the display in SPI mode is pretty easy as there's not that many pins! We'll be using hardware SPI, but you can also use software SPI (any pins) later. Start by connecting the power pins

- 3-5V Vin or V+ connects to the microcontroller 5V pin
- Gnd or G connects to Arduino ground
- SCK or CK connects to SPI clock. On Arduino Uno/Duemilanove/328-based, that's Digital 13. On Mega, it's Digital 52 and on other boards its ICSP-3 (See SPI Connections for more details ())
- MISO or SO is not connected

- MOSI or SI connects to SPI MOSI. On Arduino Uno/Duemilanove/328-based, that's Digital 11. On Mega, it's Digital 51 and on other boards its ICSP-4 (See SPI Connections for more details ())
- TCS or TC connects to our SPI Chip Select pin. We'll be using Digital 10 but you can later change this to any pin.
- RST or RT connects to our Display Reset pin. We'll be using Digital 9 but you can later change this pin too.
- DC connects to our SPI data/command select pin. We'll be using Digital 8 but you can later change this pin too.

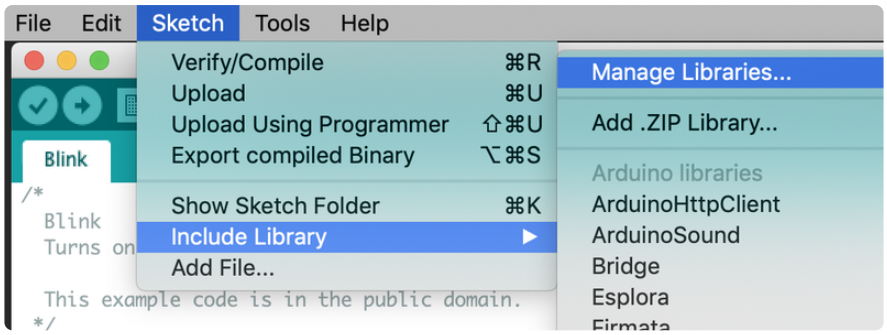
For the level shifter, we use the CD74HC4050 () which has a typical propagation delay of ~10ns



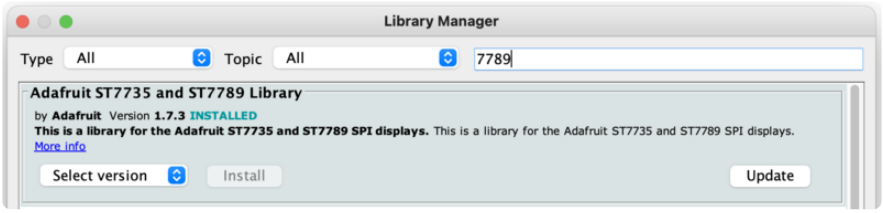

# Install Arduino Libraries

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.

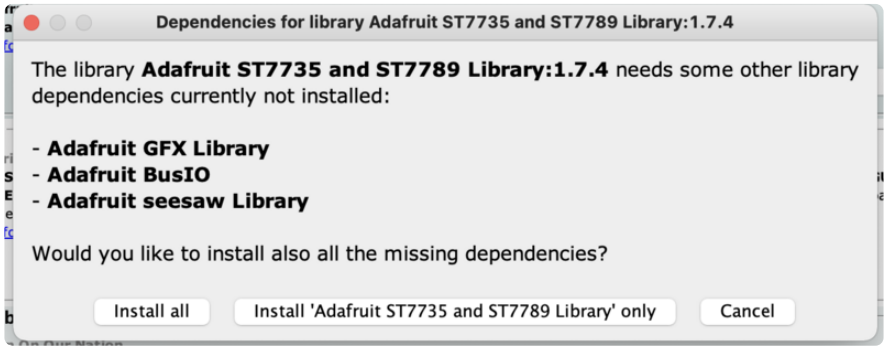
Five libraries need to be installed using the Arduino Library Manager...this is the preferred and modern way. From the Arduino “Sketch” menu, select “Include Library” then “Manage Libraries...”




Type “7789” in the search field to quickly find the first library — Adafruit ST7735 and ST7789 Library:

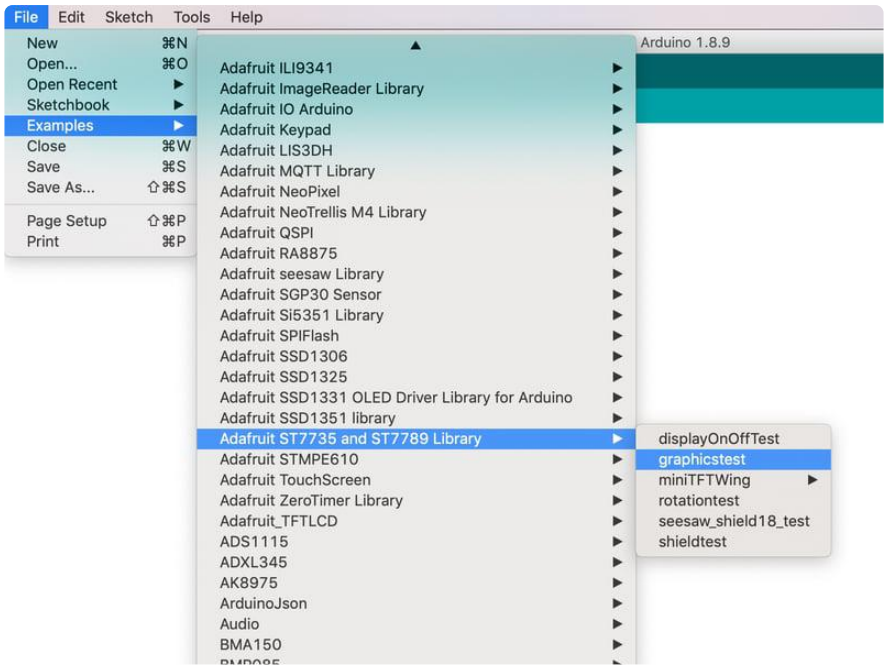



Arduino should ask you about installing dependencies. Be sure to chose "Install all".




Repeat the search and install steps, looking for the Adafruit Zero DMA, Adafruit SPIFlash, and SdFat - Adafruit Fork libraries.

After restarting the Arduino software, you should see a new example folder called Ad afruit ST7735 and ST7789 Library, and inside, an example called graphicstest.




Since this example is written for several displays, there are two changes we need to make in order to use it with this display.

First, in the graphicstest source code, look for the lines as follows:

```
// For 1.44" and 1.8" TFT with ST7735 (including HalloWing) use:
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// For 1.14", 1.3", 1.54", 1.69", and 2.0" TFT with ST7789:
//Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);
```

Comment out the first line, and uncomment the second, so it looks like:

```
// For 1.44" and 1.8" TFT with ST7735 (including HalloWing) use:
//Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// For 1.14", 1.3", 1.54", 1.69", and 2.0" TFT with ST7789:
Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);
```

Second, we need to set the correct initialization sequence. In the graphicstest source code, look for the lines as follows:

```
// Use this initializer if using a 1.8" TFT screen:
tft.initR(INITR_BLACKTAB);      // Init ST7735S chip, black tab

// OR use this initializer (uncomment) if using a 1.44" TFT:
//tft.initR(INITR_144GREENTAB); // Init ST7735R chip, green tab

// OR use this initializer (uncomment) if using a 0.96" 180x60 TFT:
//tft.initR(INITR_MINI160x80);  // Init ST7735S mini display

// OR use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
//tft.init(240, 240);           // Init ST7789 240x240
```

```
// OR use this initializer (uncomment) if using a 1.69" 280x240 TFT:
//tft.init(240, 280);           // Init ST7789 280x240

// OR use this initializer (uncomment) if using a 2.0" 320x240 TFT:
//tft.init(240, 320);           // Init ST7789 320x240
```

Comment out the first line, and uncomment the fifth, so it looks like:

```
// Use this initializer if using a 1.8" TFT screen:
//tft.initR(INITR_BLACKTAB);      // Init ST7735S chip, black tab

// OR use this initializer (uncomment) if using a 1.44" TFT:
//tft.initR(INITR_144GREENTAB); // Init ST7735R chip, green tab

// OR use this initializer (uncomment) if using a 0.96" 180x60 TFT:
//tft.initR(INITR_MINI160x80);  // Init ST7735S mini display

// OR use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
//tft.init(240, 240);           // Init ST7789 240x240

// OR use this initializer (uncomment) if using a 1.69" 280x240 TFT:
tft.init(240, 280);           // Init ST7789 280x240

// OR use this initializer (uncomment) if using a 2.0" 320x240 TFT:
//tft.init(240, 320);           // Init ST7789 320x240
```

Now upload the 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.

# Changing Pins

Now that you have it working, there's a few things you can do to change around the pins.

If you're using Hardware SPI, the CLOCK and MOSI pins are 'fixed' and can't be changed. But you can change to software SPI, which is a bit slower, and that lets you pick any pins you like. Find these lines:

```
// OPTION 1 (recommended) is to use the HARDWARE SPI pins, which are unique
// to each board and not reassignable. For Arduino Uno: MOSI = pin 11 and
// SCLK = pin 13. This is the fastest mode of operation and is required if
// using the breakout board's microSD card.

// For 1.44" and 1.8" TFT with ST7735 use:
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// For 1.14", 1.3", 1.54", 1.69", and 2.0" TFT with ST7789:
//Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);


// OPTION 2 lets you interface the display using ANY TWO or THREE PINS,
// tradeoff being that performance is not as fast as hardware SPI above.
//#define TFT_MOSI 11  // Data out
//#define TFT_SCLK 13  // Clock out

// For ST7735-based displays, we will use this call
//Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
TFT_RST);

// OR for the ST7789-based displays, we will use this call
//Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
TFT_RST);
```

Comment out option 1, and uncomment option 2 for the ST7789. Then you can change the TFT_ pins to whatever pins you'd like!

The 1.69" IPS TFT display has an auto-reset circuit on it, so you probably don't need to use the RST pin. You can change

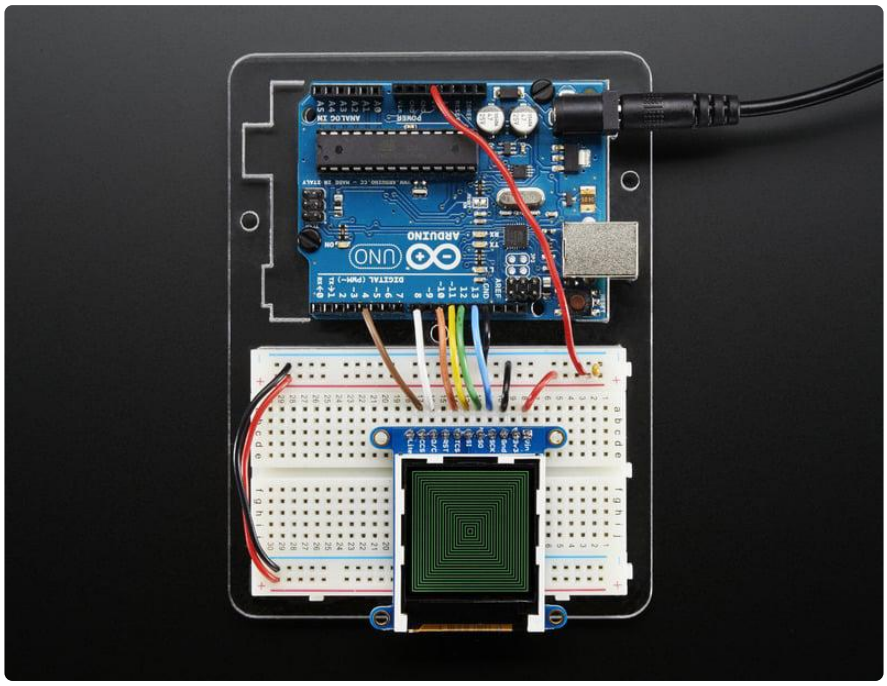
```
#define TFT_RST    9
```

to

```
#define TFT_RST   -1
```

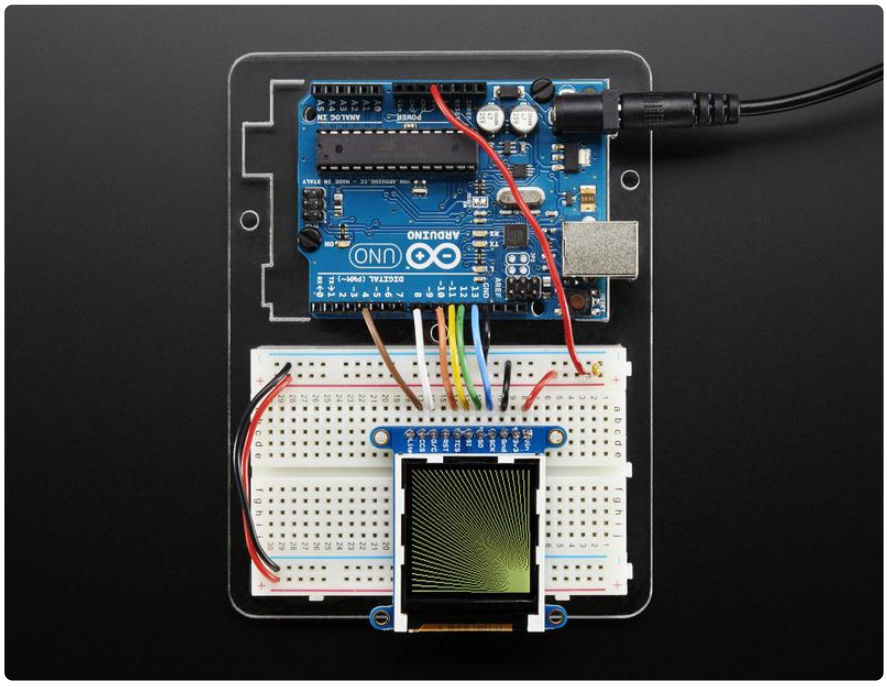
so that pin isn't used either. Or connect it up for manual TFT resetting!

# 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.



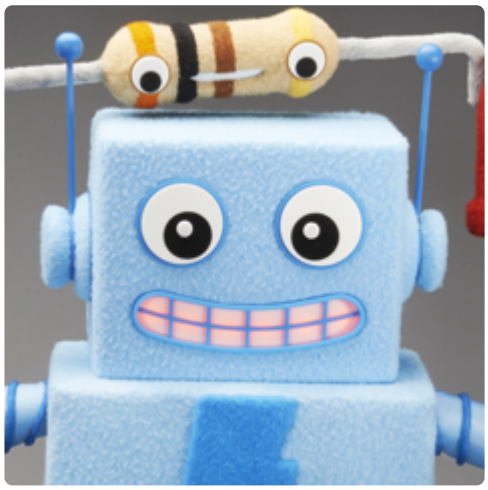
Check out our detailed tutorial here http://learn.adafruit.com/adafruit-gfx-graphics-library () It covers the latest and greatest of the GFX library!

---

# Drawing Bitmaps

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).

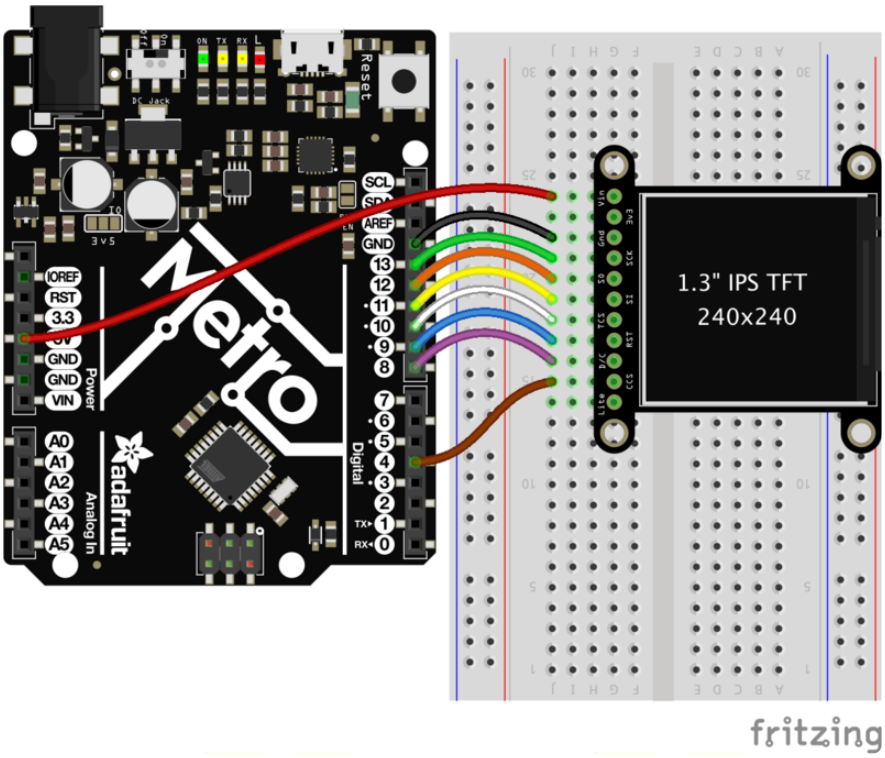
Its really easy to draw bitmaps! Lets start by downloading this image of ADABOT



Copy adabot240.bmp into the base directory of a microSD card and insert it into the microSD socket in the breakout.

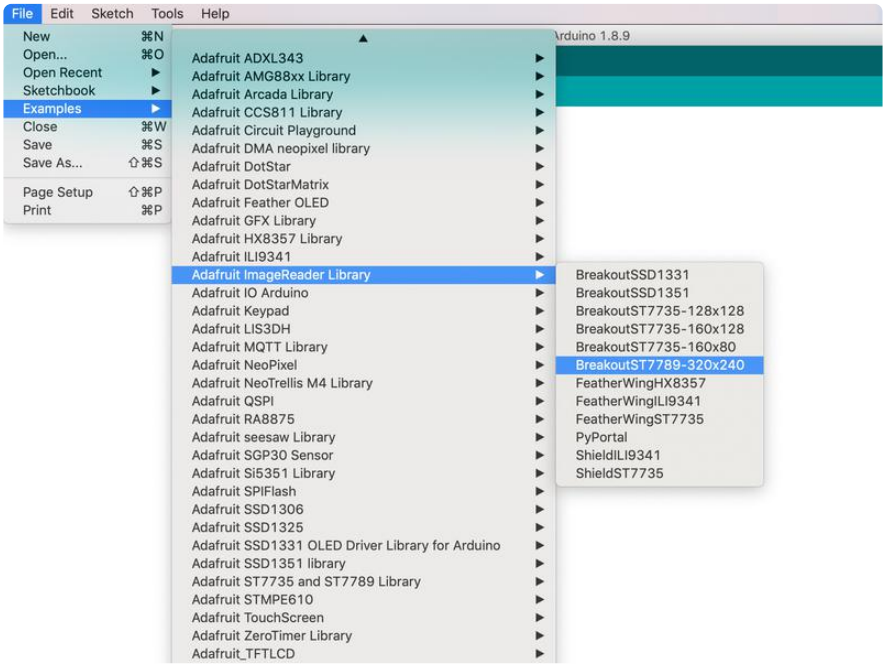
Two more wires are required to interface with the onboard SD card:

- You'll need to connect up the SO pin to the SPI MISO line on your microcontroller. On Arduino Uno/Duemilanove/328-based, that's Digital 12. On a Mega, it's Digital 50 and on Leonardo/Due it's ICSP-1 (See SPI Connections for more details ())
- Also, the CCS or CC pin to Digital 4 on your Arduino as well. You can change this pin later, but stick with this for now.




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

Open the File→examples→Adafruit ImageReader Library→BreakoutST7789 - 320x240 example:




You will need to change a couple of lines for this to work with the 280x240 display. First, we need to set this to the correct display size, so look for the following code:

```
tft.init(240, 320);           // Init ST7789 320x240
```

and change it to this:

```
tft.init(240, 280);           // Init ST7789 280x240
```

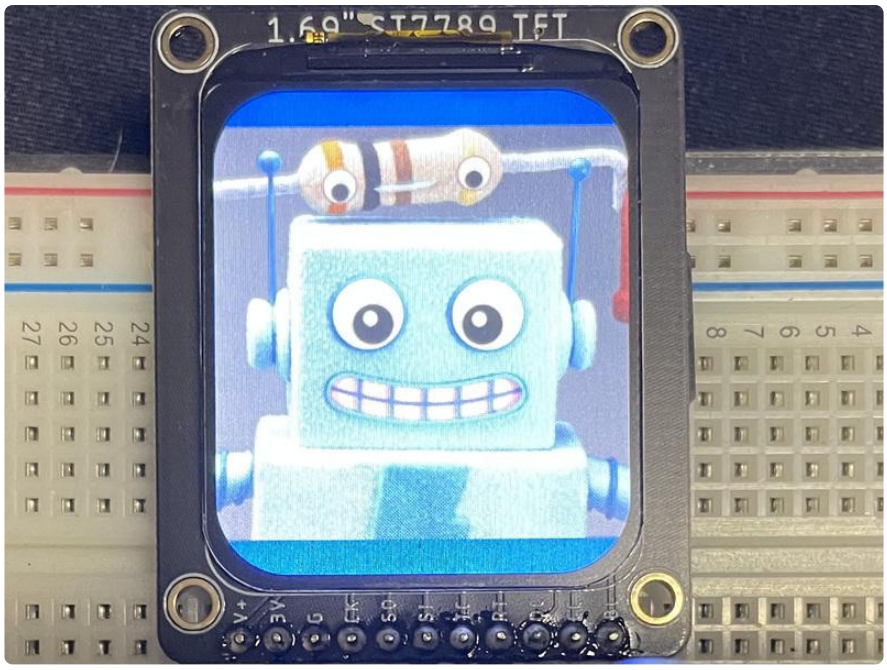
Second, we need to change the filename that we are loading, so look for the following lines of code.

```
Serial.print(F("Loading purple.bmp to screen..."));
stat = reader.drawBMP("/purple.bmp", tft, 0, 0);
```

and change them to this:

```
Serial.print(F("Loading adabot240.bmp to screen..."));
stat = reader.drawBMP("/adabot240.bmp", tft, 0, 40);
```

Now upload the example sketch to the Arduino. You should see ADABOT appear! If you have any problems, check the serial console for any messages such as not being able to initialize the microSD card or not finding the image.




To make new bitmaps, make sure they are less than 280 by 240 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 - don't forget the names must be less than 8 characters long. Just copy the BMP drawing routines below `loop()` and call

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

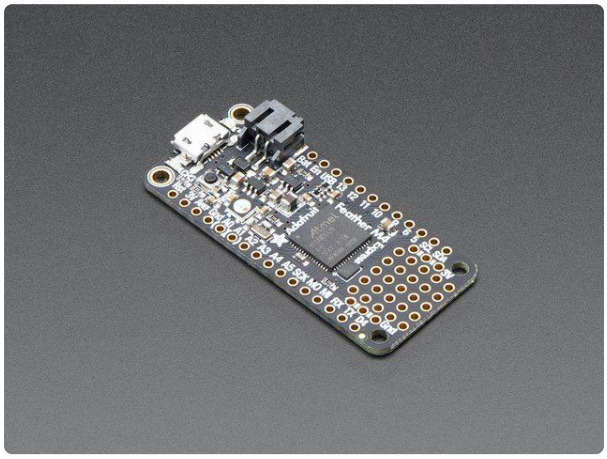
For each bitmap. They can be smaller than 280x240 and placed in any location on the screen.

---

# CircuitPython Usage

You will need a board capable of running CircuitPython such as the Metro M0 Express or the Metro M4 Express. You can also use boards such as the Feather M0 Express or the Feather M4 Express. We recommend either the Metro M4 or the Feather M4 Express because they are much faster and works better for driving a display.

This guide will be using a Feather M4 Express. The steps should be about the same for the Feather M0 Express or either of the Metros. If you haven't already, be sure to check out our Feather M4 Express () guide.

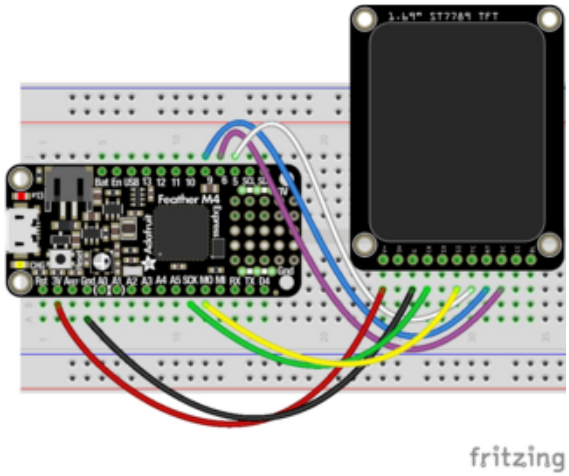


Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857

## Preparing the Breakout

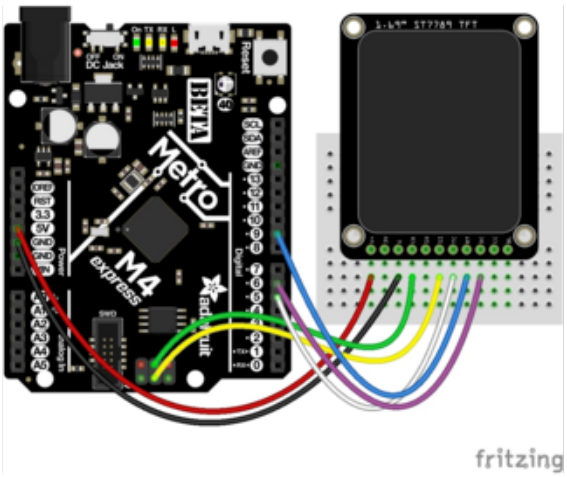
Before using the TFT Breakout, you will need to solder the headers or some wires to it. Be sure to check out the Adafruit Guide To Excellent Soldering (). After that the breakout should be ready to go.

## Feather Wiring




Vin connects to the Feather's 3V pin
GND connects to the Feather's Gnd pin
CLK connects to SPI clock. On the Feather, that's SCK
MOSI connects to SPI MOSI. On the Feather, that's also MO
CS connects to our SPI Chip Select pin. We'll be using D5
RST connects to our Reset pin. We'll be using D9.
DC connects to our SPI Chip Select pin. We'll be using D6.

## Metro M0/M4 Wiring




Vin connects to the Metro's 5V or 3.3 pin.
GND connects to any one of the Metro's Gnd pins.
CLK connects to SPI clock. On the Metro, that's Pin 3 on the ICSP Header.
MOSI connects to SPI MOSI. On the Metro, that's Pin 4 on the ISCP Header.
CS connects to our SPI Chip Select pin. We'll be using D5
RST connects to our Reset pin. We'll be using D9.
DC connects to our SPI Chip Select pin. We'll be using D6.

## CircuitPython Library Installation

First, make sure you are running the latest version of Adafruit CircuitPython () for your board.

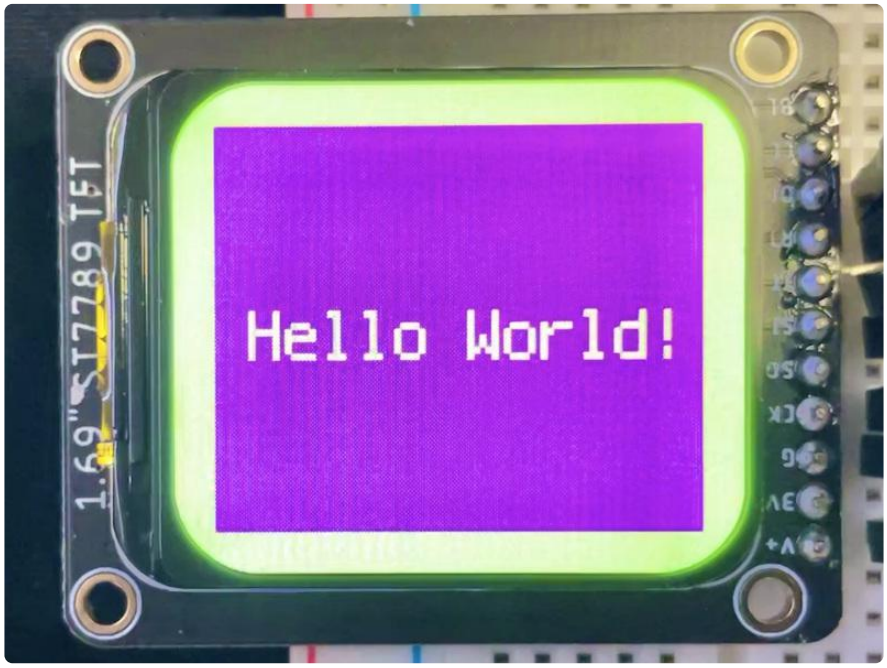
Next you'll need to install the Adafruit CircuitPython ST7789 () library on your CircuitPython board.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

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

"""
This test will initialize the display using displayio and draw a solid green
background, a smaller purple rectangle, and some yellow text.
"""
import board
import terminalio
import displayio
from adafruit_display_text import label
from adafruit_st7789 import ST7789

# Release any resources currently in use for the displays
displayio.release_displays()

spi = board.SPI()
tft_cs = board.D5
tft_dc = board.D6

display_bus = displayio.FourWire(
    spi, command=tft_dc, chip_select=tft_cs, reset=board.D9
)

display = ST7789(display_bus, width=280, height=240, rowstart=20, rotation=90)

# Make the display context
splash = displayio.Group()
display.show(splash)

color_bitmap = displayio.Bitmap(280, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

# Draw a smaller inner rectangle
inner_bitmap = displayio.Bitmap(240, 200, 1)
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(inner_bitmap, pixel_shader=inner_palette, x=20,
y=20)
splash.append(inner_sprite)

# Draw a label
text_group = displayio.Group(scale=3, x=37, y=120)
text = "Hello World!"
text_area = label.Label(terminalio.FONT, text=text, color=0xFFFF00)
text_group.append(text_area)  # Subgroup for text scaling
splash.append(text_group)

while True:
    pass
```

## Run the Script

Once everything is wired up correctly and the files are all copied over, the script should automatically run. If not, try pressing the reset button and you should see the following on the display:




---

# Python Usage

It's easy to use display breakouts with Python and the Adafruit Blinka Displayio () module.  This module allows you to easily write Python code to control the display.

We'll cover how to wire the display to your Raspberry Pi. First assemble your display.

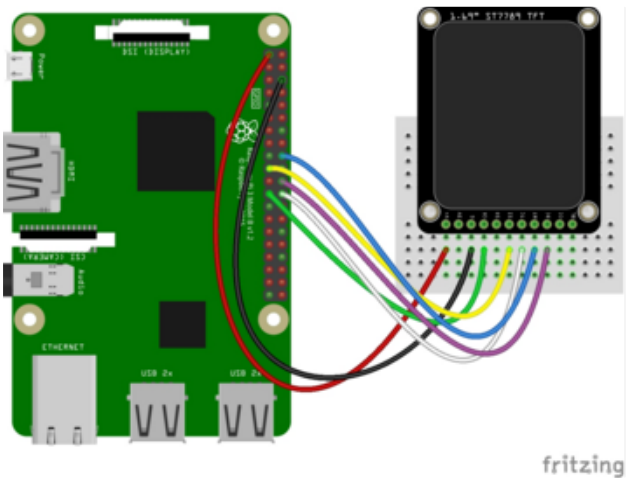
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Connect the display as shown below to your Raspberry Pi.

Note this is not a kernel driver that will let you have the console appear on the TFT. However, this is handy when you can't install an fbtft driver, and want to use the TFT purely from 'user Python' code!

You can only use this technique with Linux/computer devices that have hardware SPI support, and not all single board computers have an SPI device so check before continuing

# Wiring




Vin connects to the Raspberry Pi's 3V pin
GND connects to the Raspberry Pi's ground
CLK connects to SPI clock. On the Raspberry Pi, thats SLCK
MOSI connects to SPI MOSI. On the Raspberry Pi, thats also MOSI
CS connects to our SPI Chip Select pin. We'll be using CE0
RST connects to our Reset pin. We'll be using GPIO 24 but this can be changed later.
D/C connects to our SPI Chip Select pin. We'll be using GPIO 25, but this can be changed later as well.

# Setup

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling SPI 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 ()!

If you have previously installed the Kernel Driver with the PiTFT Easy Setup, you will need to remove it first in order to run this example.

## Python Installation of ST7789 Library

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

- `sudo pip3 install adafruit-circuitpython-st7789 adafruit-circuitpython-display-text`

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!

If that complains about pip3 not being installed, then run this first to install it:

- `sudo apt-get install python3-pip`

## Pillow Library

We also need PIL, the Python Imaging Library, to allow graphics and using text with custom fonts. There are several system libraries that PIL relies on, so installing via a package manager is the easiest way to bring in everything:

```
sudo apt-get install python3-pil
```

## NumPy Library

A recent improvement of the `RGB_Display` library makes use of NumPy for additional speed. This can be installed with the following command:

```
sudo apt-get install python3-numpy
```

## Script Download and Modifications

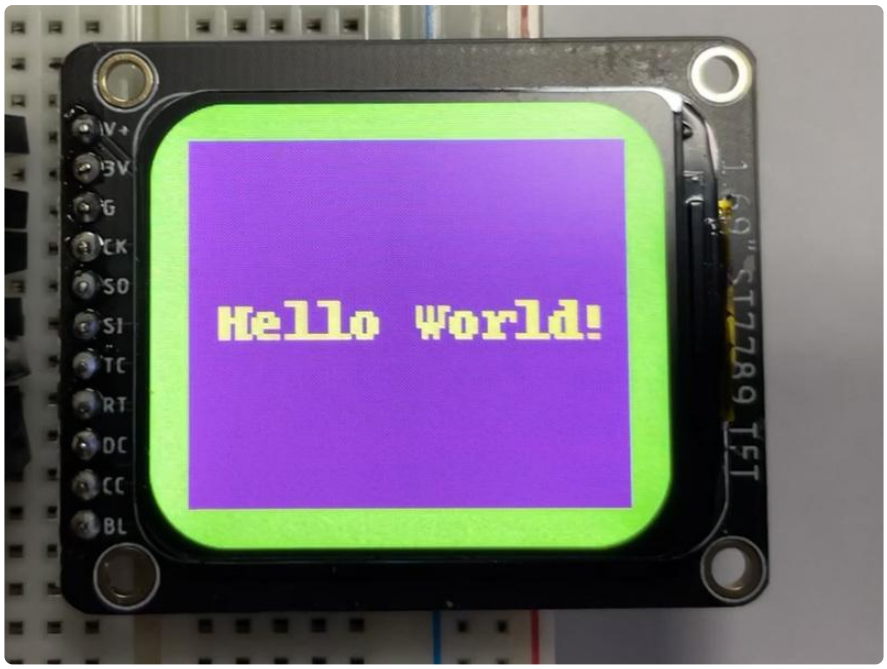
Download the script using the wget command:

```
cd ~
wget https://github.com/adafruit/Adafruit_CircuitPython_ST7789/raw/main/examples/
st7789_280x240_simpletest.py
```

Next, edit the script and make the following changes to use the correct pins:

```
spi = board.SPI()
tft_cs = board.CE0
tft_dc = board.D25

display_bus = displayio.FourWire(
    spi, command=tft_dc, chip_select=tft_cs, reset=board.D24
)
```

Now go ahead and run the script, the output should look like this:




## Full Example

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

"""
This test will initialize the display using displayio and draw a solid green
background, a smaller purple rectangle, and some yellow text.
"""
import board
import terminalio
import displayio
from adafruit_display_text import label
from adafruit_st7789 import ST7789

# Release any resources currently in use for the displays
displayio.release_displays()

spi = board.SPI()
tft_cs = board.D5
tft_dc = board.D6

display_bus = displayio.FourWire(
    spi, command=tft_dc, chip_select=tft_cs, reset=board.D9
)

display = ST7789(display_bus, width=280, height=240, rowstart=20, rotation=90)

# Make the display context
splash = displayio.Group()
display.show(splash)

color_bitmap = displayio.Bitmap(280, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

# Draw a smaller inner rectangle
inner_bitmap = displayio.Bitmap(240, 200, 1)
```

```
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(inner_bitmap, pixel_shader=inner_palette, x=20,
y=20)
splash.append(inner_sprite)

# Draw a label
text_group = displayio.Group(scale=3, x=37, y=120)
text = "Hello World!"
text_area = label.Label(terminalio.FONT, text=text, color=0xFFFF00)
text_group.append(text_area)  # Subgroup for text scaling
splash.append(text_group)

while True:
    pass
```

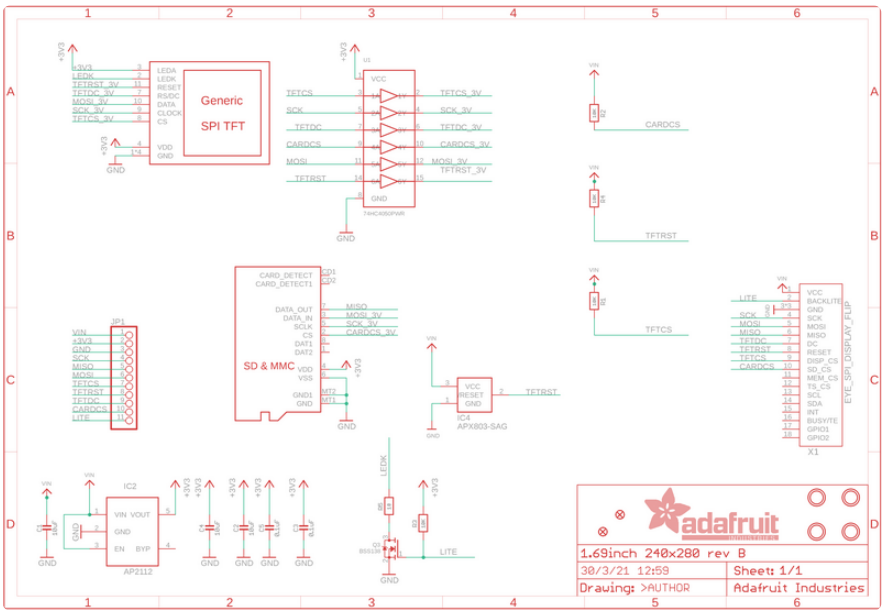
---

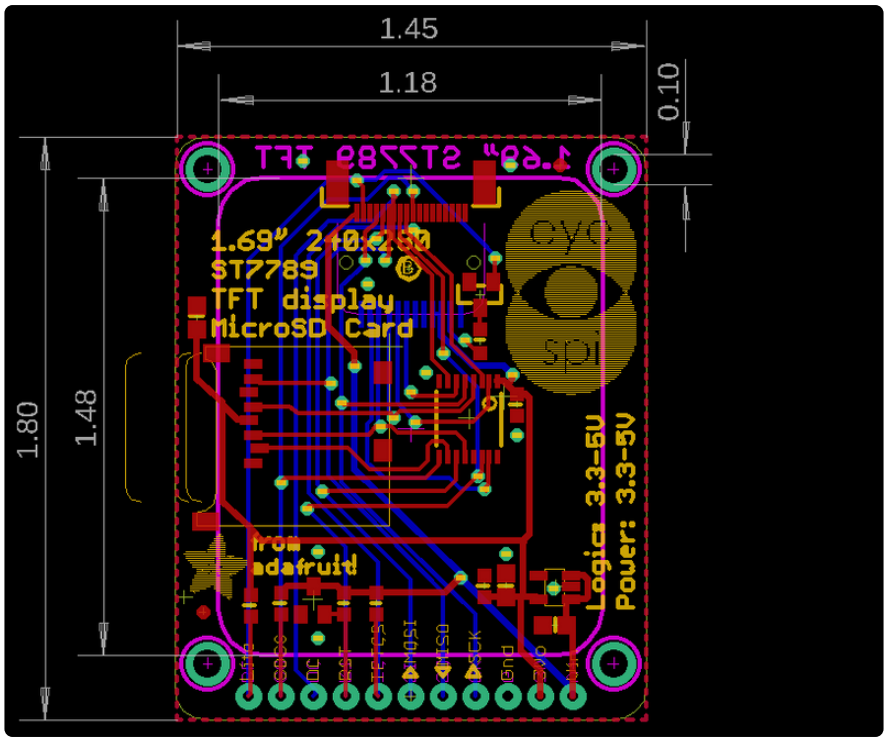
# Downloads

## Files

- ST7789VW datasheet ()
- EagleCAD PCB Files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

# Schematic and Fab Print




1.45
1.18
0.10
1.80
1.48
1.69" 240x280
ST7789
TFT display
MicroSD Card
from
adafruit
Logic 3.3-5V
Power: 3.3-5V
Gnd
Vin