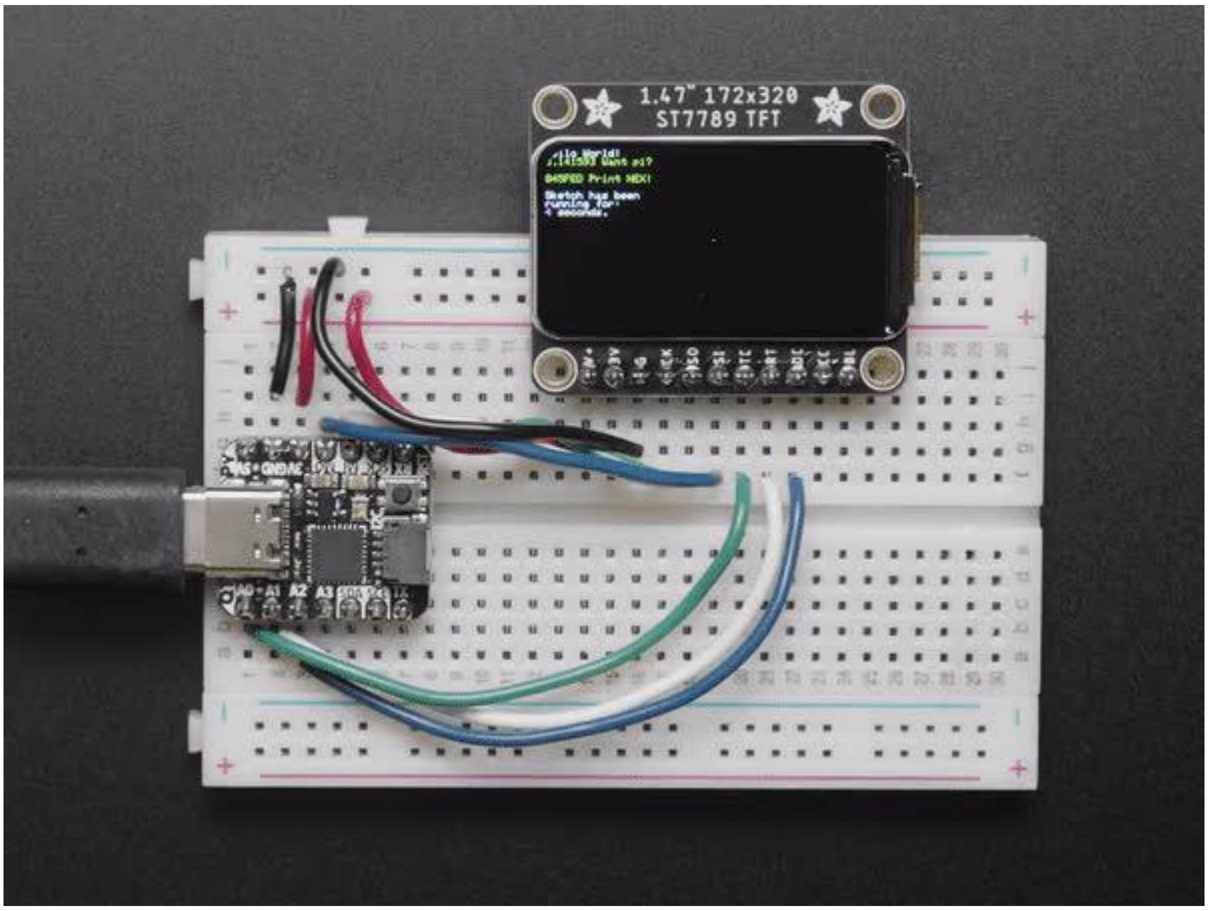
Adafruit 1.47" Round Rectangle TFT Display

Created by Melissa LeBlanc-Williams

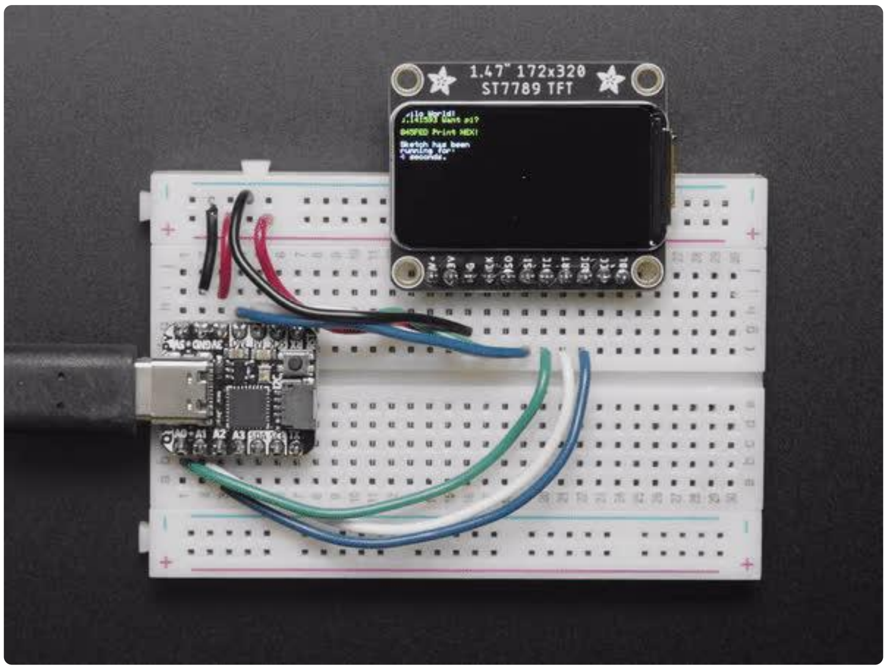


https://learn.adafruit.com/adafruit-1-47-round-rectangle-tft-display

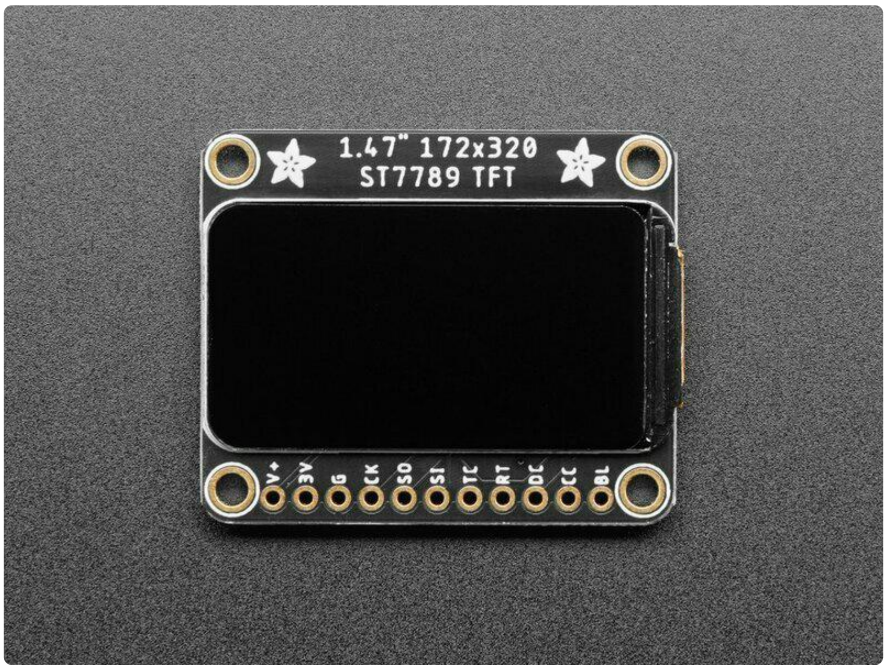
Last updated on 2022-12-01 02:16:26 PM EST

Table of Contents

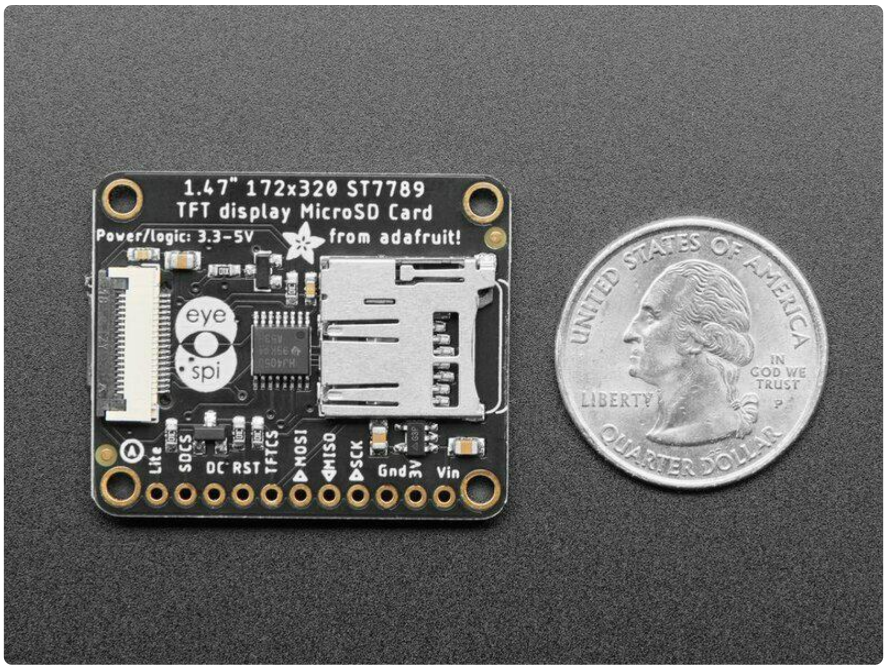
Overview



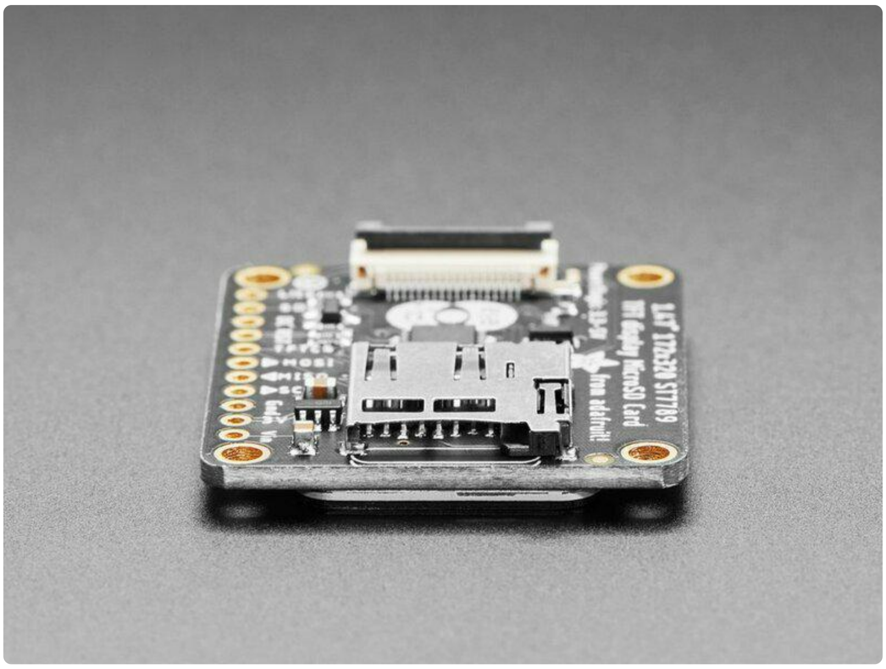
Don't be such a square - throw a curveball into your electronics with a curved-edge miniature display. Here's a new "round rect" TFT display - it's 1.47" diagonal and has high-density 250 ppi, 172x320 full-color pixels with IPS any-angle viewing. We've seen displays of this caliber used in smartwatches 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!



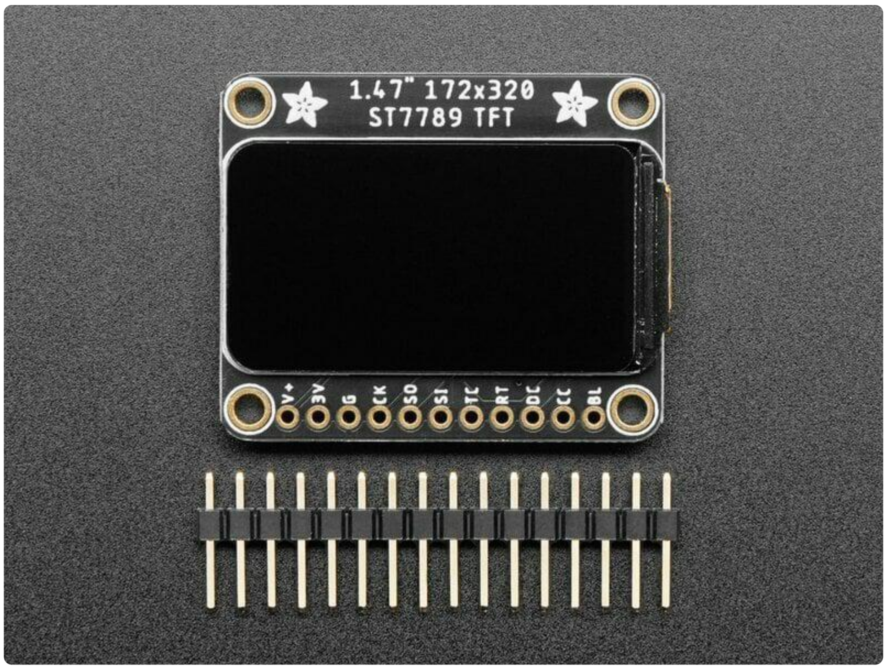
This lovely little display breakout is the best way to add a small, colorful, and very bright display to any project. Since the display uses 4-wire SPI to communicate and has its own pixel-addressable frame buffer, it can be used with every kind of microcontroller. Even a very small one with low memory and few pins available! The 1.47" display has 172x320 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.



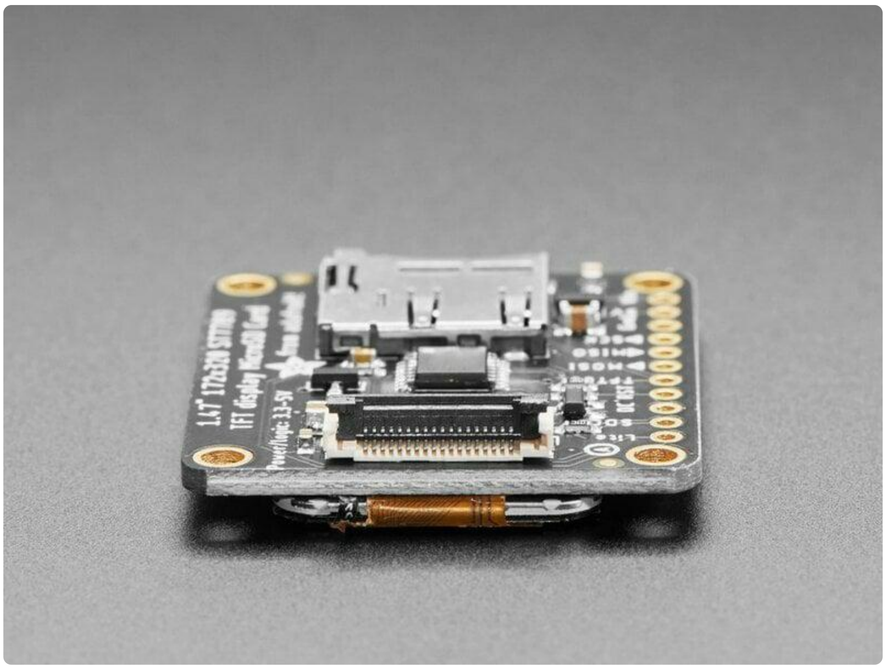
Note that the way we get the rounded corners is by deleting pixels. The corner pixels are still addressed in RAM, they just don't appear, so it isn't like you have to do some special radial-pixel mapping. Treat it like a rectangular display.



The breakout has the TFT display soldered on (it uses a delicate flex-circuit connector) as well as an ultra-low-dropout 3.3V regulator, auto-reset circuitry, and a 3/5V level shifter so you can use it with 3.3V or 5V power and logic. We also had a little extra 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 ().

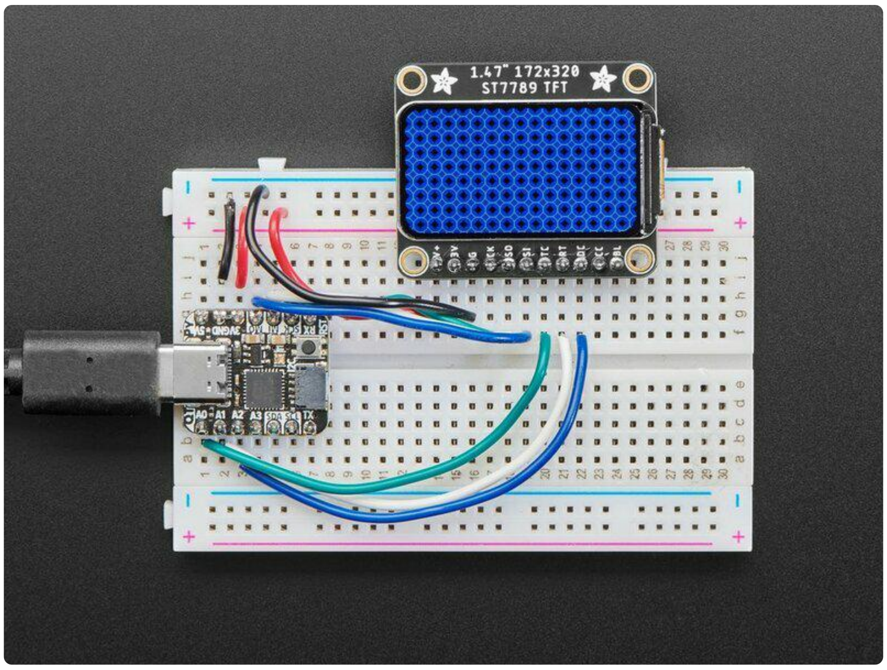


Of course, we wouldn't just leave you with a datasheet and a "good luck!" - we've written a full open-source graphics Arduino library that can draw pixels, lines, rectangles, circles, text, and bitmaps as well as example code (). The code is written for Arduino but can be easily ported to your favorite microcontroller! Wiring is easy, we strongly encourage using the hardware SPI pins of your Arduino as software SPI is noticeably slower when dealing with this size display. For Raspberry Pi or other Single Board Computer Python users, we have a user-space Pillow-compatible library (). For CircuitPython there's a displayio driver for native support ().

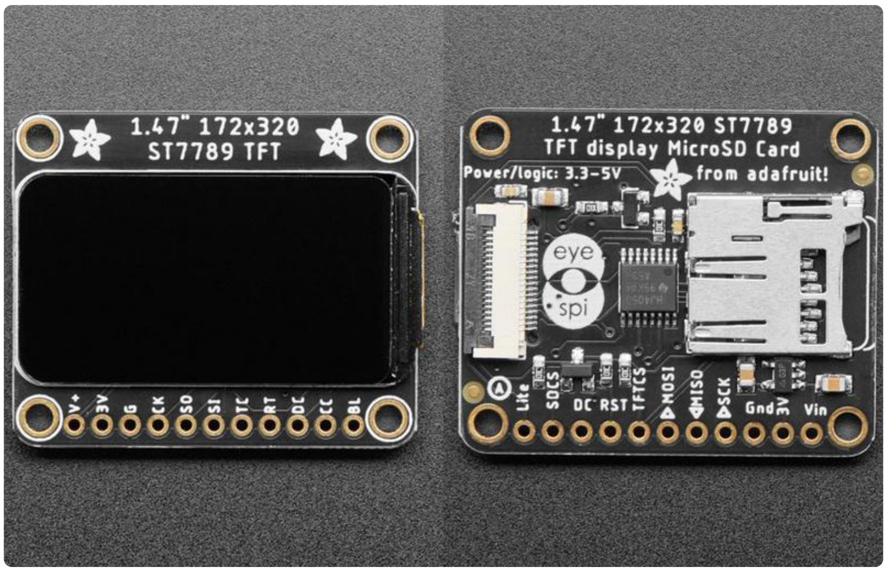


This display breakout also features an 18-pin "EYE SPI" standard FPC connector with a flip-top connector. You can use an 18-pin 0.5mm pitch FPC cable to connect to all the GPIO pins, for when you want to skip the soldering.

Please note! This display is designed originally for smartwatches and similar, where there's a glass over the screen. Without something gently holding the screen down, the backlight can eventually peel away from the TFT. (It's not destructive but it may be unattractive) You can prevent this by, ideally, adding a plastic or glass cover/overlay. If using bare, try dabbing a touch of E6000 or similar craft glue on the thin side edges, or using a thin piece of tape to keep the front TFT attached to the backlight.



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.
- 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
- 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)
- 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 microcontroller Reset pin, which works for most cases.

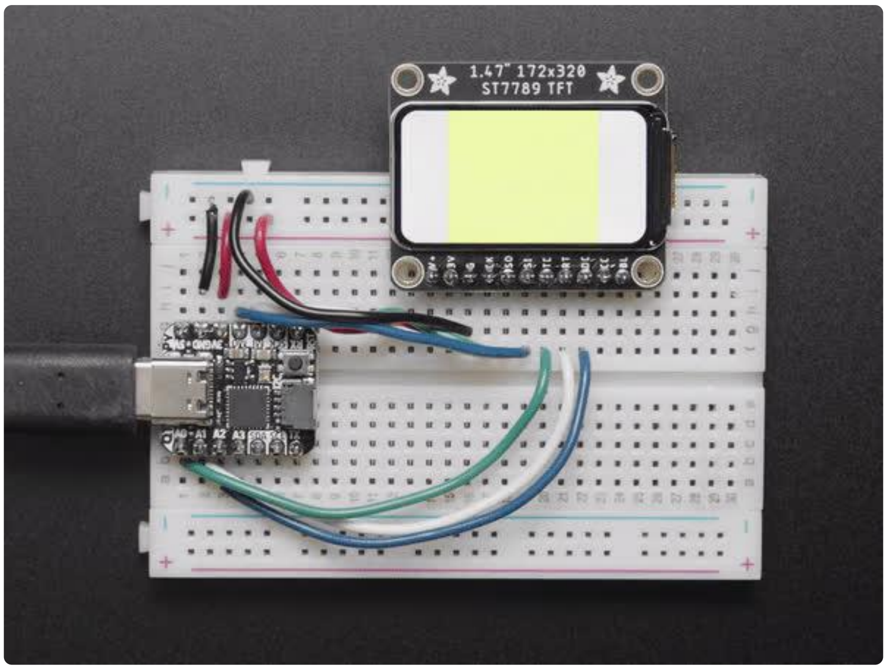
There is an automatic-reset chip connected so it will reset on power-up. Use 3-5V logic level

- 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

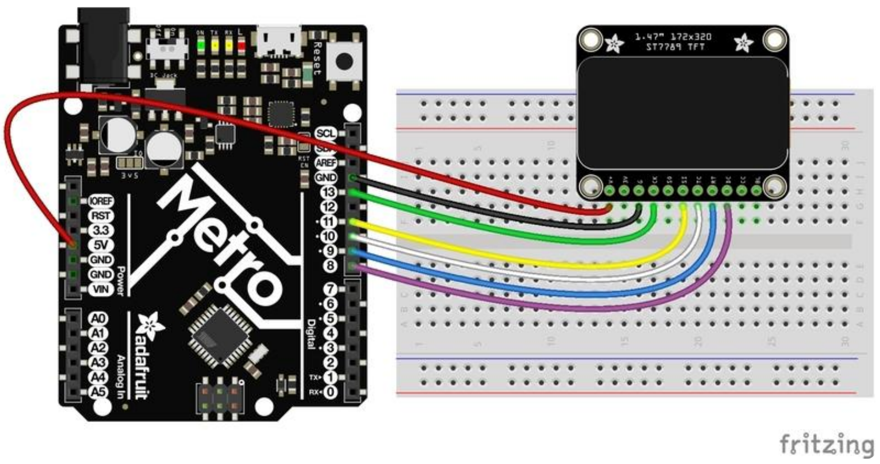


Basic Graphics Test Wiring

Wiring up the display in SPI mode is pretty easy as there's not that many pins! This is 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 it's 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 it's ICSP-4 (See SPI Connections for more details ())
- TCS or TC connects to the SPI Chip Select pin. This uses Digital 10 but you can later change this to any pin.
- RST or RT connects to the Display Reset pin. This uses 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.

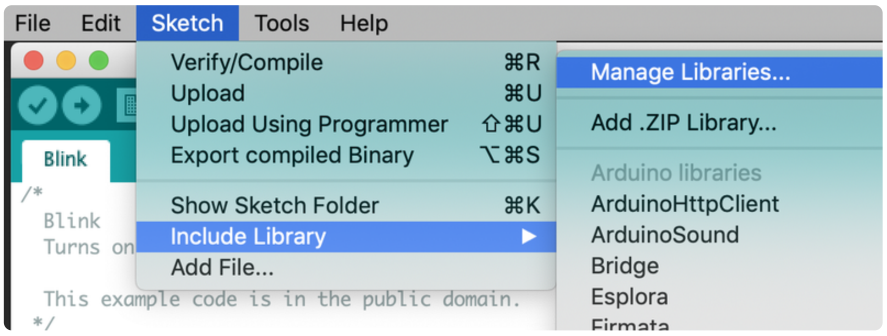
For the built-in level shifter, the board uses a CD74HC4050 () chip, which has a typical propagation delay of ~10ns



Install Arduino Libraries

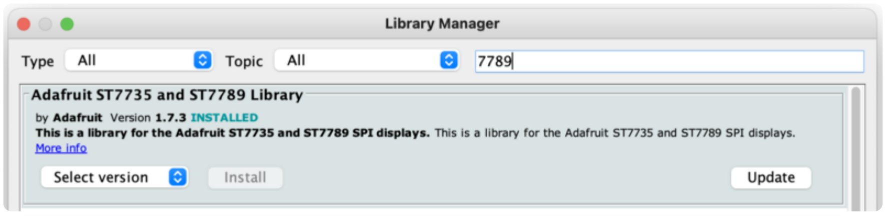
Adafruit has example code ready to go for use with these TFTs. It's written for Arduino, which should be portable to any microcontroller by adapting the C++ source.

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…”



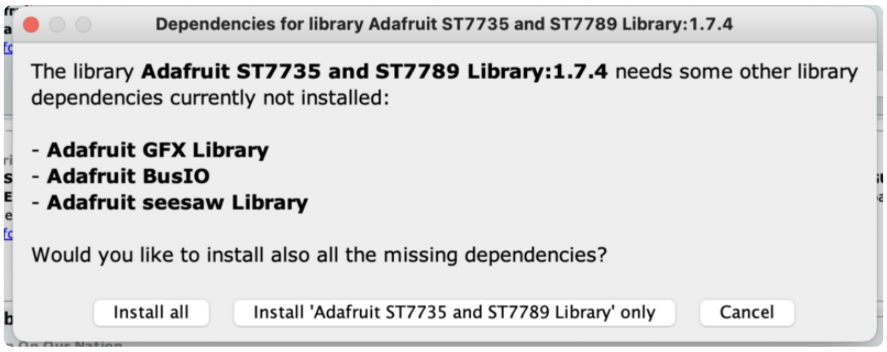
```
/*
  Blink
  Turns on

  This example code is in the public domain.
*/
```

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



Adafruit ST7735 and ST7789 Library
by Adafruit Version 1.7.3 INSTALLED
This is a library for the Adafruit ST7735 and ST7789 SPI displays. This is a library for the Adafruit ST7735 and ST7789 SPI displays.

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


Dependencies for library Adafruit ST7735 and ST7789 Library:1.7.4

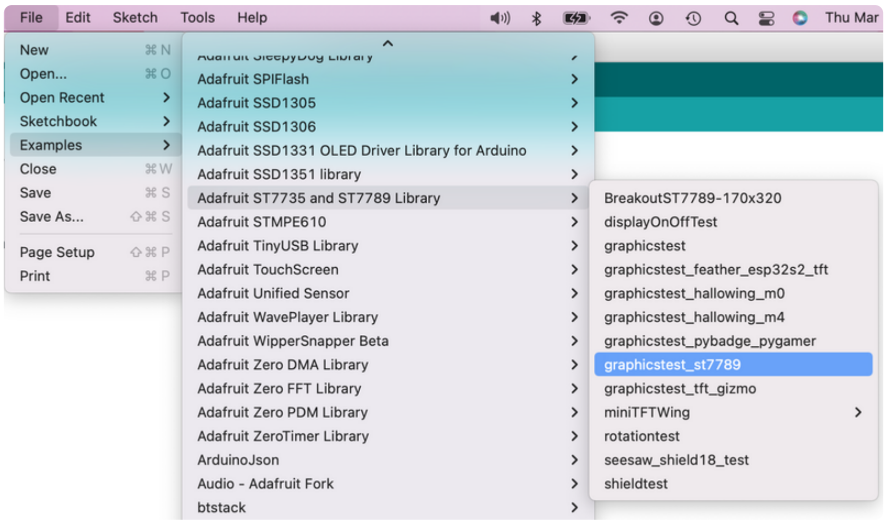
The library **Adafruit ST7735 and ST7789 Library:1.7.4** needs some other library dependencies currently not installed:

- **Adafruit GFX Library**
- **Adafruit BusIO**
- **Adafruit seesaw Library**

Would you like to install also all the missing dependencies?

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

After restarting the Arduino software, you should see a new example folder called Adafruit ST7735 and ST7789 Library, and inside, an example called graphicstest_st7789.



Since this example is written for several displays, there is just one change needed in order to use it with the this display.

You just need to set the correct initialization sequence. In the graphicstest_st7789 source code, look for the lines as follows:

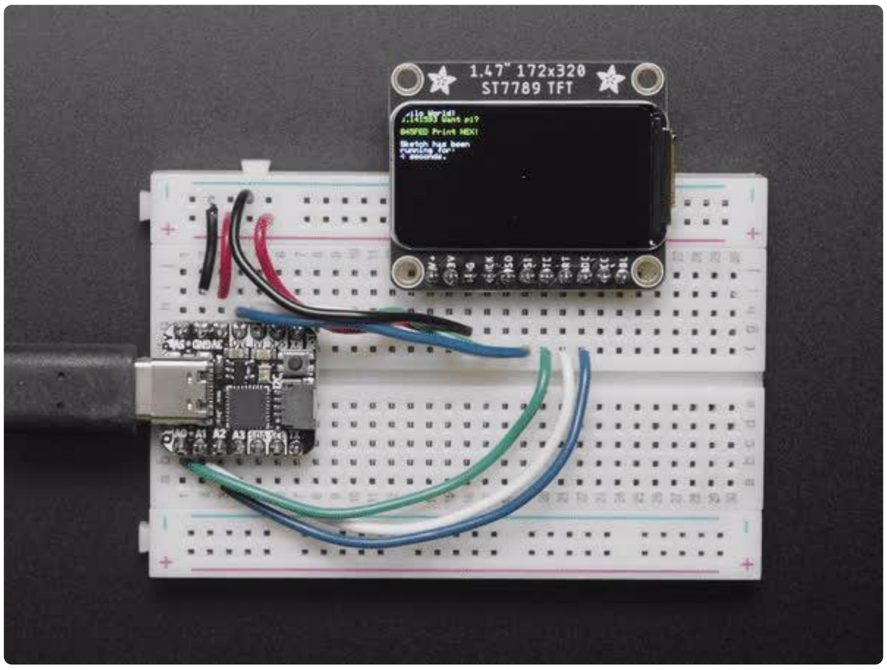
```
// Use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
tft.init(240, 240);           // Init ST7789 240x240

// 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

// OR use this initializer (uncomment) if using a 1.14" 240x135 TFT:
//tft.init(135, 240);           // Init ST7789 240x135

// OR use this initializer (uncomment) if using a 1.47" 172x320 TFT:
//tft.init(172, 320);           // Init ST7789 172x320
```

```
// OR use this initializer (uncomment) if using a 1.9" 170x320 TFT:
//tft.init(170, 320);           // Init ST7789 170x320
```

Comment out the first line, and uncomment the line that corresponds to this display, so it looks like:

```
// 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

// OR use this initializer (uncomment) if using a 1.14" 240x135 TFT:
//tft.init(135, 240);           // Init ST7789 240x135

// OR use this initializer (uncomment) if using a 1.47" 172x320 TFT:
tft.init(172, 320);           // Init ST7789 172x320

// OR use this initializer (uncomment) if using a 1.9" 170x320 TFT:
//tft.init(170, 320);           // Init ST7789 170x320
```

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.

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

//Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
TFT_RST);
```

Comment out option 1, and uncomment option 2. Then you can change the pin names that begin with TFT_ to whatever pins you'd like!

The 7789-based TFT displays have an auto-reset circuit on it, so you probably don't need to use the RST pin. You can change

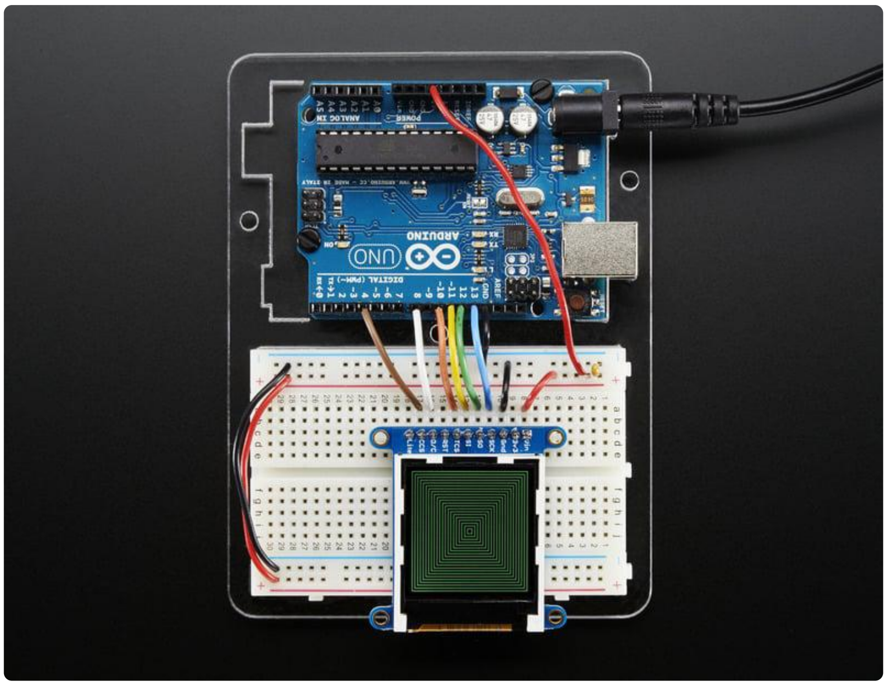
`#define TFT_RST    9`

to

`#define TFT_RST   -1`

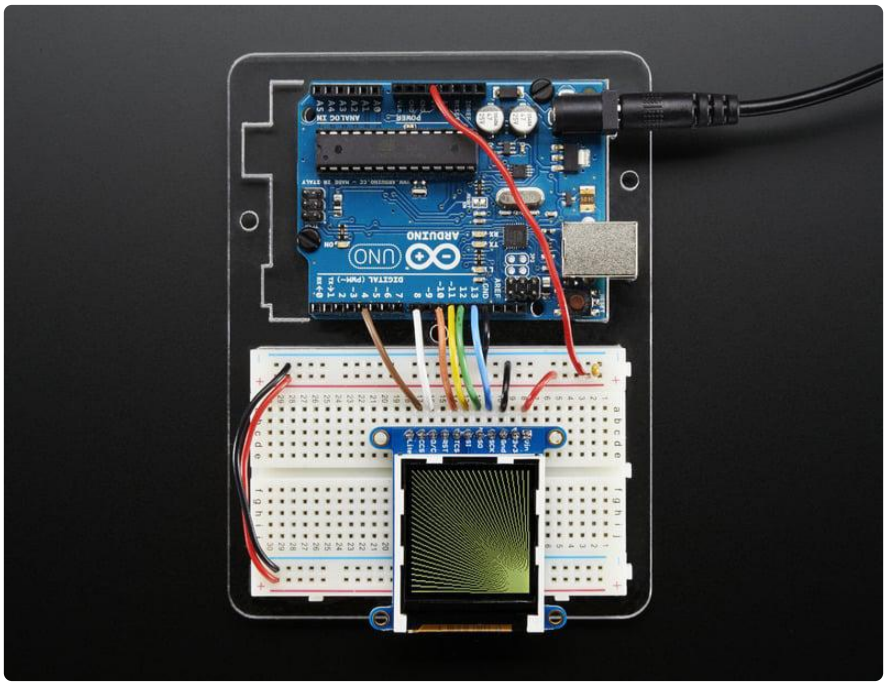
so that pin isn't used either. Or connect it up for manual TFT resetting!

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!

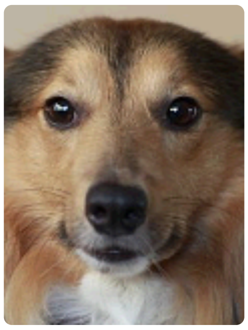
Drawing Bitmaps

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

It's really easy to draw bitmaps! Start by downloading this image of ADABOT

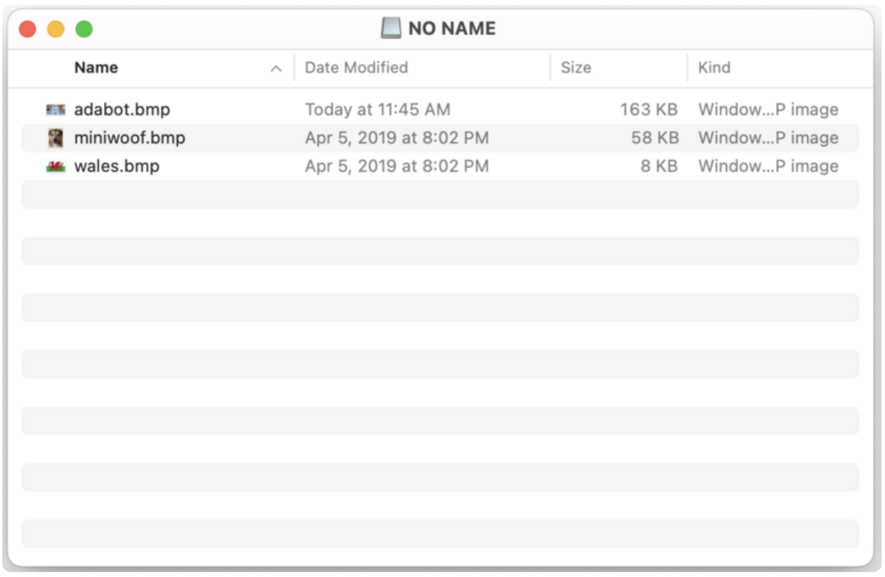
Rename the image to adabot.bmp.

You will want to download and 2 more images as well. Download and save the image below as miniwoof.bmp.



Download and save the image below as wales.bmp.

Copy all 3 images into the base directory of a microSD card and insert it into the microSD socket in the breakout. The contents of your SD card should look like this:

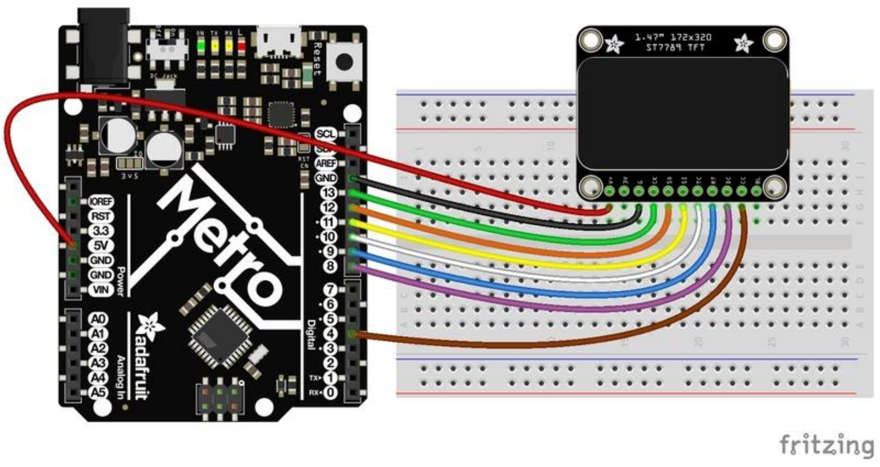


NO NAME

Name	Date Modified	Size	Kind
adabot.bmp	Today at 11:45 AM	163 KB	Window...P image
miniwoof.bmp	Apr 5, 2019 at 8:02 PM	58 KB	Window...P image
wales.bmp	Apr 5, 2019 at 8:02 PM	8 KB	Window...P image

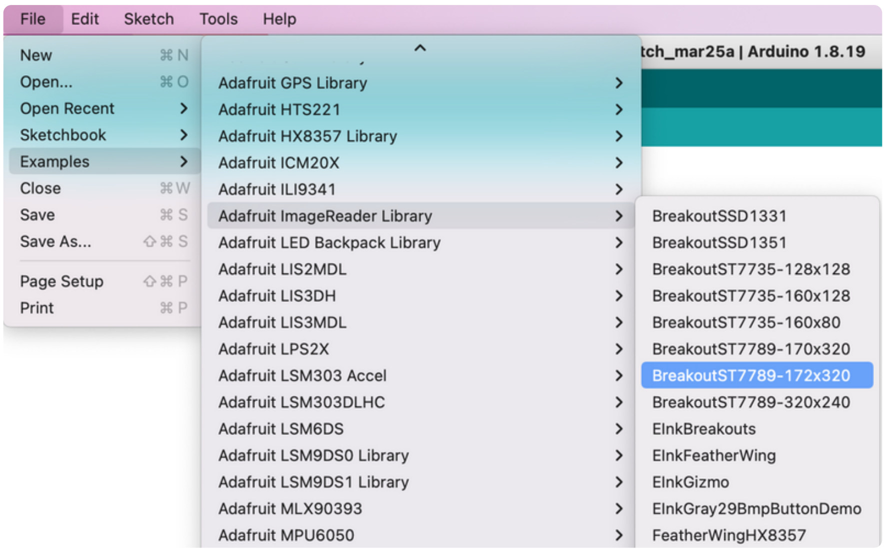
Two more wires are required to interface with the onboard SD card:

- 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 - 172x320 example:



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 no bigger than the display size in 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);
```

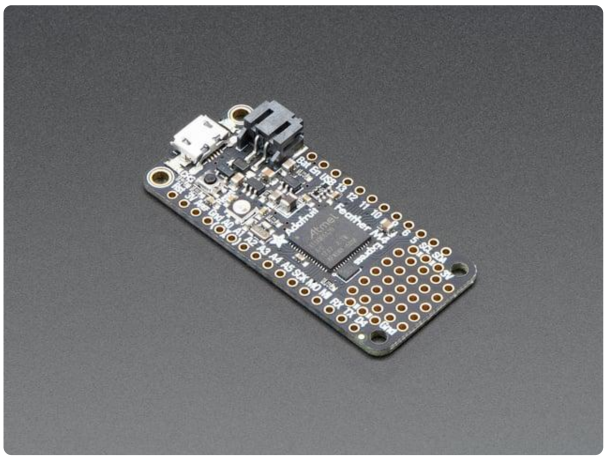
For each bitmap.

They can be smaller than 320x172 and placed in any location on the screen.

CircuitPython Usage

You will need a board capable of running CircuitPython, such as the Adafruit Metro M4 Express. You can also use boards such as the Adafruit Feather M4 Express. Adafruit recommends using at least an M4 processor 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 Metro. If you haven't already, be sure to check out the Feather M4 Express () g uide.

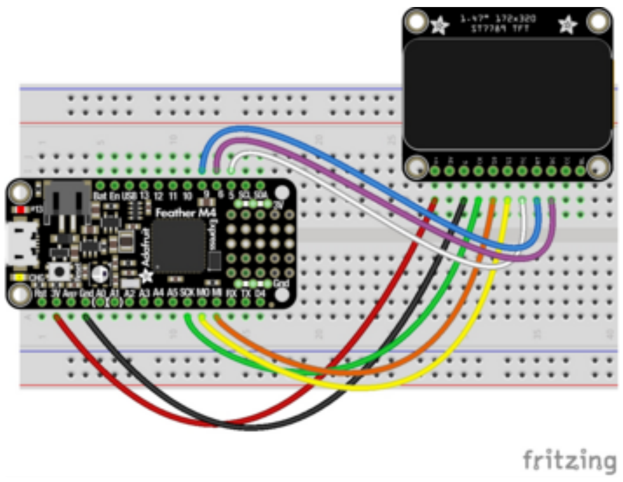


Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857

Preparing the Breakout

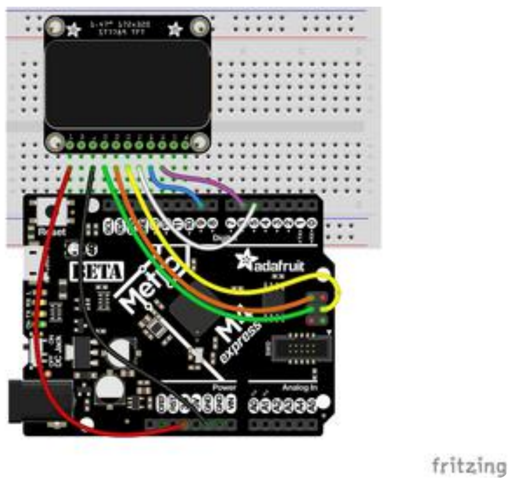
Before using the TFT Breakout, you will need to solder the headers or some wires to it. Be sure to check out the Adafruit Guide To Excellent Soldering (). After that the breakout should be ready to go.

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. This uses D5
RST connects to the Reset pin. This uses D9.
DC connects to our SPI Chip Select pin. This uses D6.

Metro 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

BORDER_WIDTH = 20
TEXT_SCALE = 3

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

spi = board.SPI()
tft_cs = board.D5
tft_dc = board.D6
tft_rst = board.D9

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

display = ST7789(display_bus, width=320, height=172, colstart=34, rotation=270)

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

color_bitmap = displayio.Bitmap(display.width, display.height, 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(
    display.width - (BORDER_WIDTH * 2), display.height - (BORDER_WIDTH * 2), 1
)
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(
    inner_bitmap, pixel_shader=inner_palette, x=BORDER_WIDTH, y=BORDER_WIDTH
)
splash.append(inner_sprite)

# Draw a label
text_area = label.Label(
    terminalio.FONT,
    text="Hello World!",
    color=0xFFFF00,
    scale=TEXT_SCALE,
    anchor_point=(0.5, 0.5),
    anchored_position=(display.width // 2, display.height // 2),
)
splash.append(text_area)
```

```
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 () mod ule. This module allows you to easily write Python code to control the display.

Below covers how to wire the display to a Raspberry Pi.

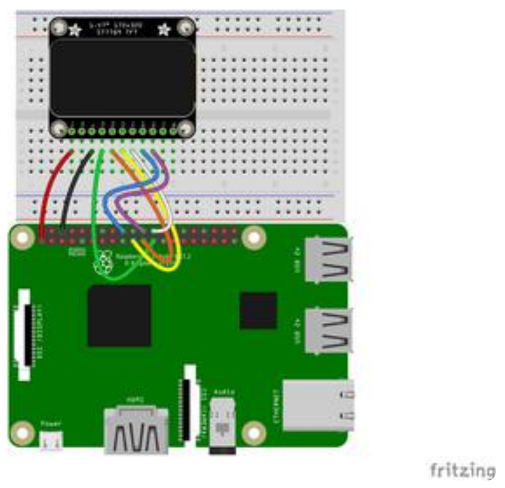
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

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, that's SLCK
MOSI connects to SPI MOSI. On the Raspberry Pi, that's also MOSI
CS connects to our SPI Chip Select pin. This uses CE0
RST connects to our Reset pin. This uses GPIO 24, but it can be changed later.
D/C connects to our SPI Chip Select pin. This uses GPIO 25, but it 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

PIL, the Python Imaging Library, will also be needed to display 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

You can install NumPy which allows for additional speed. This can be installed with the following command:

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

Running the Code

Next to run the code, you will need to do a simple modification so the CircuitPython code runs on Python.

Script Download and Modifications

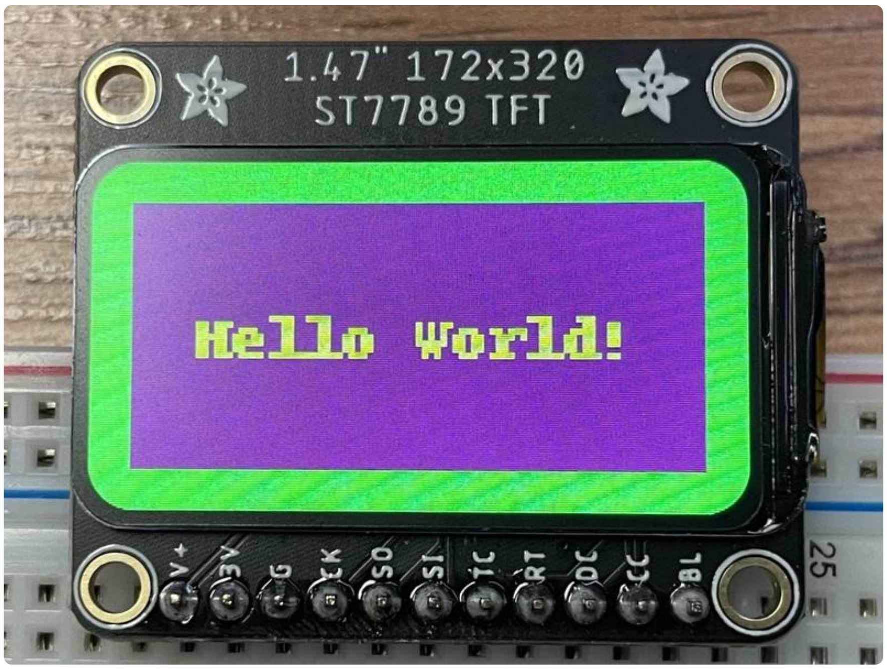
Download the script using the wget command:

```
cd ~
wget https://github.com/adafruit/Adafruit_CircuitPython_ST7789/raw/main/examples/
st7789_172x320_1.47_simpletest.py
```

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

```
tft_cs = board.CE0
tft_dc = board.D25
tft_rst = 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

BORDER_WIDTH = 20
TEXT_SCALE = 3

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

spi = board.SPI()
tft_cs = board.D5
tft_dc = board.D6
tft_rst = board.D9

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

display = ST7789(display_bus, width=320, height=172, colstart=34, rotation=270)

# Make the display context
```

```
splash = displayio.Group()
display.show(splash)

color_bitmap = displayio.Bitmap(display.width, display.height, 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(
    display.width - (BORDER_WIDTH * 2), display.height - (BORDER_WIDTH * 2), 1
)
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(
    inner_bitmap, pixel_shader=inner_palette, x=BORDER_WIDTH, y=BORDER_WIDTH
)
splash.append(inner_sprite)

# Draw a label
text_area = label.Label(
    terminalio.FONT,
    text="Hello World!",
    color=0xFFFF00,
    scale=TEXT_SCALE,
    anchor_point=(0.5, 0.5),
    anchored_position=(display.width // 2, display.height // 2),
)
splash.append(text_area)

while True:
    pass
```

Python Installation of RGB Display Library

The other library that you can use to draw to displays in Python is the RGB Display library. This library allows you to more directly use PIL so if you wanted to draw a JPG image, for instance, or other advanced graphics, then you can easily do so. To install the library, run this command:

```
sudo pip3 install adafruit-circuitpython-rgb-display
```

Displaying a JPG Image

To display an image other than a Bitmap such as a JPG, you just need to load it in with Pillow. To demonstrate this, download this image of blinka:

You can download it straight onto you Pi with this command:

```
wget https://cdn-learn.adafruit.com/assets/assets/000/110/219/original/
adafruit_products_blinka.jpg
mv adafruit_products_blinka.jpg blinka.jpg
```

You can download the script to your Pi with this command:

```
wget https://github.com/adafruit/Adafruit_CircuitPython_RGB_Display/raw/main/
examples/rgb_display_pillow_image.py
```

Run the Script

You will need to modify the script by commenting out a few lines of the ILI9341 display initializer. Leave everything else that including the line that contains `cs=cs_pin,` alone. The code with the commented out part should look like this:

```
#disp = ili9341.ILI9341(
#    spi,
#    rotation=90,  # 2.2", 2.4", 2.8", 3.2" ILI9341
    cs=cs_pin,
    dc=dc_pin,
```

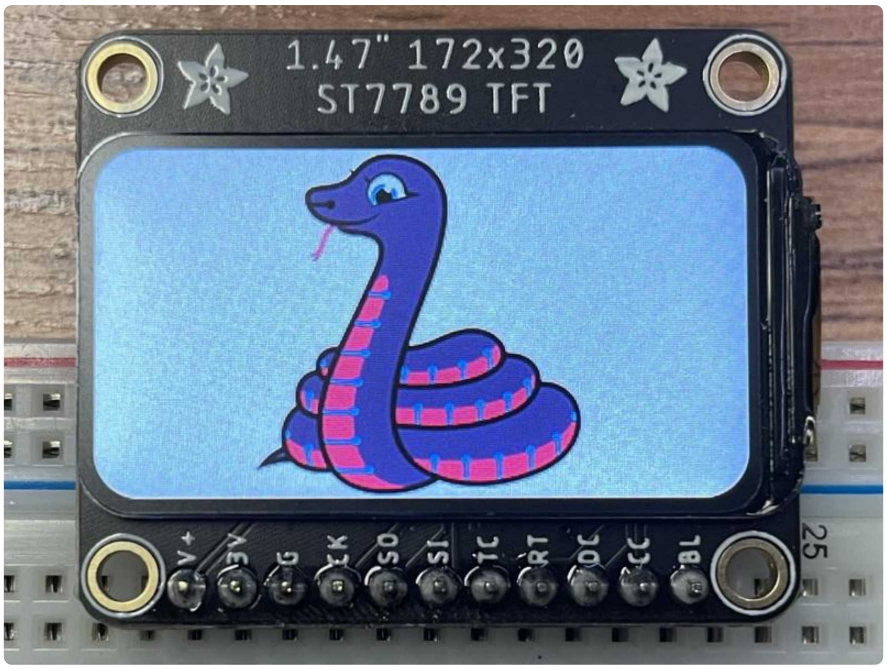
Then you will need to uncomment the line that for this display:

```
disp = st7789.ST7789(spi, rotation=90, width=172, height=320, x_offset=34, # 1.47"
ST7789
```

Now you can run the script with this command:

```
sudo python3 rgb_display_pillow_image.py
```

Your display should look like this:



Full Example

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

"""
Be sure to check the learn guides for more usage information.

This example is for use on (Linux) computers that are using CPython with
Adafruit Blinka to support CircuitPython libraries. CircuitPython does
not support PIL/pillow (python imaging library)!

Author(s): Melissa LeBlanc-Williams for Adafruit Industries
"""

import digitalio
import board
from PIL import Image, ImageDraw
from adafruit_rgb_display import ili9341
from adafruit_rgb_display import st7789  # pylint: disable=unused-import
from adafruit_rgb_display import hx8357  # pylint: disable=unused-import
from adafruit_rgb_display import st7735  # pylint: disable=unused-import
from adafruit_rgb_display import ssd1351  # pylint: disable=unused-import
from adafruit_rgb_display import ssd1331  # pylint: disable=unused-import

# Configuration for CS and DC pins (these are PiTFT defaults):
cs_pin = digitalio.DigitalInOut(board.CE0)
dc_pin = digitalio.DigitalInOut(board.D25)
reset_pin = digitalio.DigitalInOut(board.D24)

# Config for display baudrate (default max is 24mhz):
BAUDRATE = 24000000

# Setup SPI bus using hardware SPI:
spi = board.SPI()

# pylint: disable=line-too-long
# Create the display:
# disp = st7789.ST7789(spi, rotation=90,                            # 2.0" ST7789
# disp = st7789.ST7789(spi, height=240, y_offset=80, rotation=180,  # 1.3", 1.54"
ST7789
```

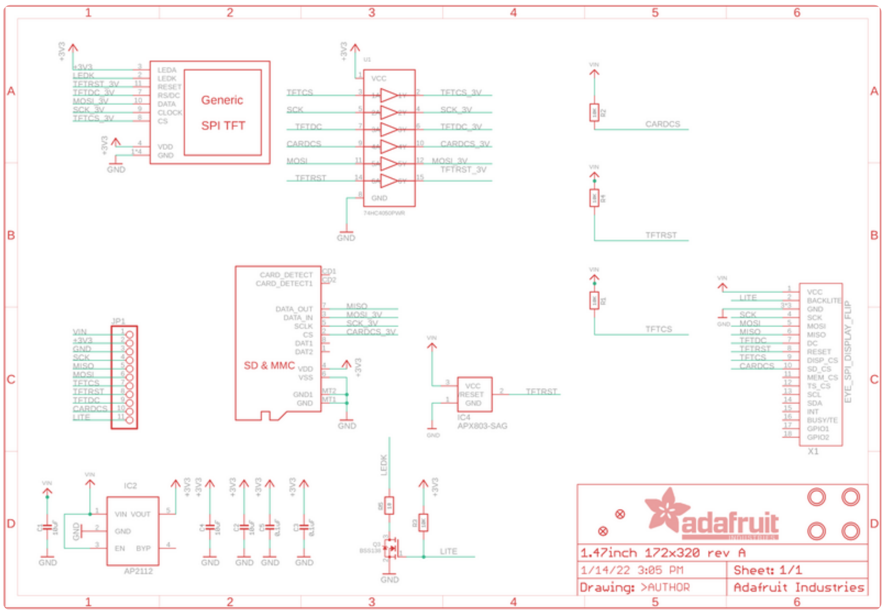
```
# disp = st7789.ST7789(spi, rotation=90, width=135, height=240, x_offset=53,
y_offset=40, # 1.14" ST7789
# disp = st7789.ST7789(spi, rotation=90, width=172, height=320, x_offset=34, #
1.47" ST7789
# disp = st7789.ST7789(spi, rotation=270, width=170, height=320, x_offset=35, #
1.9" ST7789
# disp = hx8357.HX8357(spi, rotation=180,                           # 3.5" HX8357
# disp = st7735.ST7735R(spi, rotation=90,                           # 1.8" ST7735R
# disp = st7735.ST7735R(spi, rotation=270, height=128, x_offset=2, y_offset=3,   #
1.44" ST7735R
# disp = st7735.ST7735R(spi, rotation=90, bgr=True,                 # 0.96" MiniTFT
ST7735R
# disp = ssd1351.SSD1351(spi, rotation=180,                         # 1.5" SSD1351
# disp = ssd1351.SSD1351(spi, height=96, y_offset=32, rotation=180, # 1.27" SSD1351
# disp = ssd1331.SSD1331(spi, rotation=180,                         # 0.96" SSD1331
disp = ili9341.ILI9341(
    spi,
    rotation=90,  # 2.2", 2.4", 2.8", 3.2" ILI9341
    cs=cs_pin,
    dc=dc_pin,
    rst=reset_pin,
    baudrate=BAUDRATE,
)
# pylint: enable=line-too-long

# Create blank image for drawing.
# Make sure to create image with mode 'RGB' for full color.
if disp.rotation % 180 == 90:
    height = disp.width  # we swap height/width to rotate it to landscape!
    width = disp.height
else:
    width = disp.width  # we swap height/width to rotate it to landscape!
    height = disp.height
image = Image.new("RGB", (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a black filled box to clear the image.
draw.rectangle((0, 0, width, height), outline=0, fill=(0, 0, 0))
disp.image(image)

image = Image.open("blinka.jpg")

# Scale the image to the smaller screen dimension
image_ratio = image.width / image.height
screen_ratio = width / height
if screen_ratio < image_ratio:
    scaled_width = image.width * height // image.height
    scaled_height = height
else:
    scaled_width = width
    scaled_height = image.height * width // image.width
image = image.resize((scaled_width, scaled_height), Image.Resampling.BICUBIC)

# Crop and center the image
x = scaled_width // 2 - width // 2
y = scaled_height // 2 - height // 2
image = image.crop((x, y, x + width, y + height))

# Display image.
disp.image(image)
```

Downloads

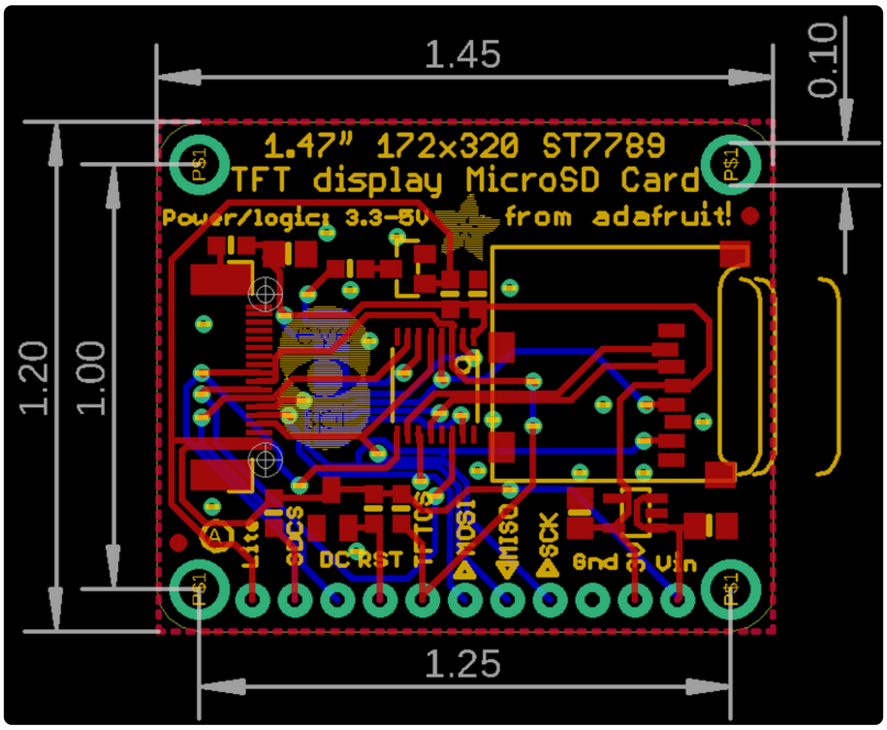
Files

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

Schematic and Fab Print



Generic SPI TFT
SD & MMC
1.47inch 172x320 rev A
1/14/22 3:05 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries



1.45
0.10
1.20
1.00
1.25
1.47" 172x320 ST7789
TFT display MicroSD Card
Power/logic: 3.3-5V
from adafruit!
Lite
SDCS
DC
RST
TCS
MOSI
MISO
SCK
Gnd
3V
Vin