Adafruit 2.4" Color TFT Touchscreen Breakout
Created by lady ada
Add some jazz & pizazz to your project with a color touchscreen LCD. This TFT display is 2.4" diagonal with a bright (4 white-LED) backlight and it's 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 a resistive 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.

In addition, 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)).

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 resitive 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.
Pinouts

The 2.8” 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 at 3.3V

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, its 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
- **3.3V / 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) They can be used in 8-bit or SPI mode.

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)! If your microcontroller

- **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
- **Backkite** - 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 D7** - these are the 8 bits of parallel data sent to the TFT in 8-bit mode. **D0** is the least-significant-bit and **D7** is the MSB
Assembly

Prepare the header strip:
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down
Add the breakout board:
Place the breakout board over the pins so that the short pins poke through the breakout pads
And Solder!
Be sure to solder all pins for reliable electrical contact.

Solder the longer power/data strip first

(For tips on soldering, be sure to check out our Guide to Excellent Soldering [https://adafruit.it/aTk]).
You're done! Check your solder joints visually.

If your display isn't taped down to the breakout PCB, peel off the tape covering and press down so the TFT sticks to the circuit board.
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 & 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. It's 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!

![8-Bit Wiring](image)

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://adafruit.it/aHk) (this contains the low-level code specific to this device), and second is the Adafruit GFX Library (https://adafruit.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.

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://adafruit.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.
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!
IFT LCD test
Using Adafruit 2.8" TFT Breakout Board Pinout
Unknown LCD driver chip: 4141
If using the Adafruit 2.8" TFT Arduino shield, the line:
#define USE_ADAFRUIT_SHIELD_PINOUT
should appear in the library Reader (Adafruit_TFT.h).
If using the breakout board, it should NOT be #defined!
Also if using the breakout, double-check that all wiring
matches the tutorial.
SPI Wiring & Test

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 its ICSP-3 (See SPI Connections for more details (https://adafru.it/d5h))

- **MISO** connects to SPI MISO. On Arduino Uno/Duemilanove/328-based, that's Digital 12. On Mega’s, its Digital 50 and on Leonardo/Due its ICSP-1 (See SPI Connections for more details (https://adafru.it/d5h))

- **MOSI** connects to SPI MOSI. On Arduino Uno/Duemilanove/328-based, that's Digital 11. On Mega’s, its Digital 51 and on Leonardo/Due its ICSP-4 (See SPI Connections for more details (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://adafruit.it/d4d) (this contains the low-level code specific to this device), and second is the Adafruit GFX Library (https://adafruit.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!

![Library Manager](https://learn.adafruit.com/adafruit-2-4-color-tft-touchscreen-breakout)

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 (https://adafruit.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!

![Library Manager](https://learn.adafruit.com/adafruit-2-4-color-tft-touchscreen-breakout)

Repeat the search and install steps one more time, looking for Adafruit_ZeroDMA this time around. Install this 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.

![Example Sketch](https://learn.adafruit.com/adafruit-2-4-color-tft-touchscreen-breakout)
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.
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time (microseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen fill</td>
<td></td>
</tr>
</tbody>
</table>
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).

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

It's really easy to draw bitmaps. We have a library for it, Adafruit_ImageReader, which can be installed through the Arduino Library Manager (Sketch→Include Library→Manage Libraries...). Enter “imageread” in the search field and the library is easy to spot:

Let's start by downloading this image of pretty flowers

![Purple Flowers](image.png)

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 brown 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 ImageReader Library→ShieldILI9341` example to your Arduino + breakout. You will see the flowers appear!
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://adafruit.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!
Resistive Touchscreen

The LCD has a 2.8" 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.

![Image of a touchscreen device]

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.

Download Library

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

https://adafruit.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. 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 D4
- Y- to Arduino D5
- 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 top

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.
CircuitPython Displayio Quickstart

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 it's much faster and works better for driving a display. For this guide, we 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 (https://adafruit.it/EEm) guide.

For this guide, we'll assume you have a Feather M4 Express. The steps should be about the same for the Feather M0 Express. To start, if you haven't already done so, follow the assembly instructions for the Feather M4 Express in our Feather M4 Express guide (https://adafruit.it/EEm).

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 (https://adafruit.it/drI). Also, follow the SPI Wiring and Test page of this guide to be sure your display is setup for SPI. After that, the breakout should be ready to go.

Wiring the Breakout to the Feather

- 3.3V Vin connects to the Feather 3V pin
- GND connects to Feather ground
- CLK connects to SPI clock. On the Feather that's SCK.
- MISO connects to SPI MISO. On the Feather that's MI
- MOSI connects to SPI MOSI. On the Feather that's MO
- CS connects to our SPI Chip Select pin. We'll be using Digital 9 but you can later change this to any pin
- D/C connects to our SPI data/command select pin. We'll be using Digital 10 but you can later change this pin too.
- RST connects to our reset pin. We'll be using Digital 6 but you can later change this pin too.
**Required CircuitPython Libraries**

To use this display with displayio, there is only one required library.

First, make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next, you'll need to install the necessary libraries to use the hardware—carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.

Remember for non-express boards, you'll need to manually install the necessary libraries from the bundle:

- adafruit ili9341

Before continuing make sure your board's lib folder or root filesystem has the adafruit ili9341 file copied over.

**Code Example Additional Libraries**

For the Code Example, you will need an additional library. We decided to make use of a library so the code didn't get overly complicated.
Go ahead and install this in the same manner as the driver library by copying the adafruit_display_text folder over to
the lib folder on your CircuitPython device.

CircuitPython Code Example

Temporarily unable to load content:

Code Details

Let's take a look at the sections of code one by one. We start by importing the board so that we can initialize SPI, displayio, terminalio for the font, a label, and the adafruit_ili9341 driver.

```python
import board
import displayio
import terminalio
from adafruit_display_text import label
import adafruit_ili9341
```

Next, we set the SPI object to the board's SPI with the easy shortcut function board.SPI(). By using this function, it finds the SPI module and initializes using the default SPI parameters. Next we set the Chip Select and Data/Command pins that will be used.

```python
spi = board.SPI()
tft_cs = board.D9
tft_dc = board.D10
```

In the next two lines, we release the displays. This is important because if the Feather is reset, the display pins are not automatically released and this makes them available for use again. We set the display bus to FourWire which makes use of the SPI bus.

```python
displayio.release_displays()
display_bus = displayio.FourWire(spi, command=tft_dc, chip_select=tft_cs, reset=board.D6)
```

Finally, we initialize the driver with a width of 320 and a height of 240. If we stopped at this point and ran the code, we would have a terminal that we could type at and have the screen update.

```python
display = adafruit_ili9341.ILI9341(display_bus, width=320, height=240)
```
Next we create a background splash image. We do this by creating a group that we can add elements to and adding that group to the display. In this example, we are limiting the maximum number of elements to 10, but this can be increased if you would like. The display will automatically handle updating the group.

```python
splash = displayio.Group(max_size=10)
display.show(splash)
```

Next we create a Bitmap which is like a canvas that we can draw on. In this case we are creating the Bitmap to be the same size as the screen, but only have one color. The Bitmaps can currently handle up to 256 different colors. We create a Palette with one color and set that color to 0x00FF00 which happens to be green. Colors are Hexadecimal values in the format of RRGGBB. Even though the Bitmaps can only handle 256 colors at a time, you get to define what those 256 different colors are.

```python
color_bitmap = displayio.Bitmap(320, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00 # Bright Green
```

With all those pieces in place, we create a TileGrid by passing the bitmap and palette and draw it at \((0, 0)\) which represents the display's upper left.

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

This creates a solid green background which we will draw on top of.
Next we will create a smaller purple rectangle. The easiest way to do this is to create a new bitmap that is a little smaller than the full screen with a single color and place it in a specific location. In this case we will create a bitmap that is 20 pixels smaller on each side. The screen is 320x240, so we'll want to subtract 40 from each of those numbers.

We'll also want to place it at the position \((20, 20)\) so that it ends up centered.

```python
inner_bitmap = displayio.Bitmap(280, 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)
```

Since we are adding this after the first rectangle, it's automatically drawn on top. Here's what it looks like now.
Next let's add a label that says "Hello World!" on top of that. We're going to use the built-in Terminal Font and scale it up by a factor of three. To scale the label only, we will make use of a subgroup, which we will then add to the main group.

Labels are centered vertically, so we'll place it at 120 for the Y coordinate, and around 57 pixels make it appear to be centered horizontally, but if you want to change the text, change this to whatever looks good to you. Let's go with some yellow text, so we'll pass it a value of \texttt{0xFFFF00}.

```python
text_group = displayio.Group(max_size=10, scale=3, x=57, 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)
```

Finally, we place an infinite loop at the end so that the graphics screen remains in place and isn't replaced by a terminal.

```python
while True:
    pass
```
Using Touch

We won't be covering how to use the touchscreen with CircuitPython in this guide, but the libraries required to use it are:

- For enabling capacitive touch use the Adafruit_CircuitPython_FocalTouch (https://adafruit.it/Fsy) library.
- For enabling resistive touch use the Adafruit_CircuitPython_STMPE610 (https://adafruit.it/Fsz) library.

Where to go from here

Be sure to check out this excellent guide to CircuitPython Display Support Using displayio (https://adafruit.it/EGh)
Troubleshooting

- 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.
Downloads

Datasheets

- ILI9341 TFT controller chip datasheet (https://adafruit.it/d4l) (this is what you want to refer to if porting or if you want to look at the TFT command set)
- Datasheet for the raw 2.4" screen itself (https://adafruit.it/fgr)
- Adafruit ILI9341 Arduino driver library (https://adafruit.it/d4d)
- EagleCAD PCB files on GitHub (https://adafruit.it/rEl)
- Fritzing object in the Adafruit Fritzing Library (https://adafruit.it/aP3)

Schematic

Click to embiggen

Fabrication Print

Dimensions in Inches