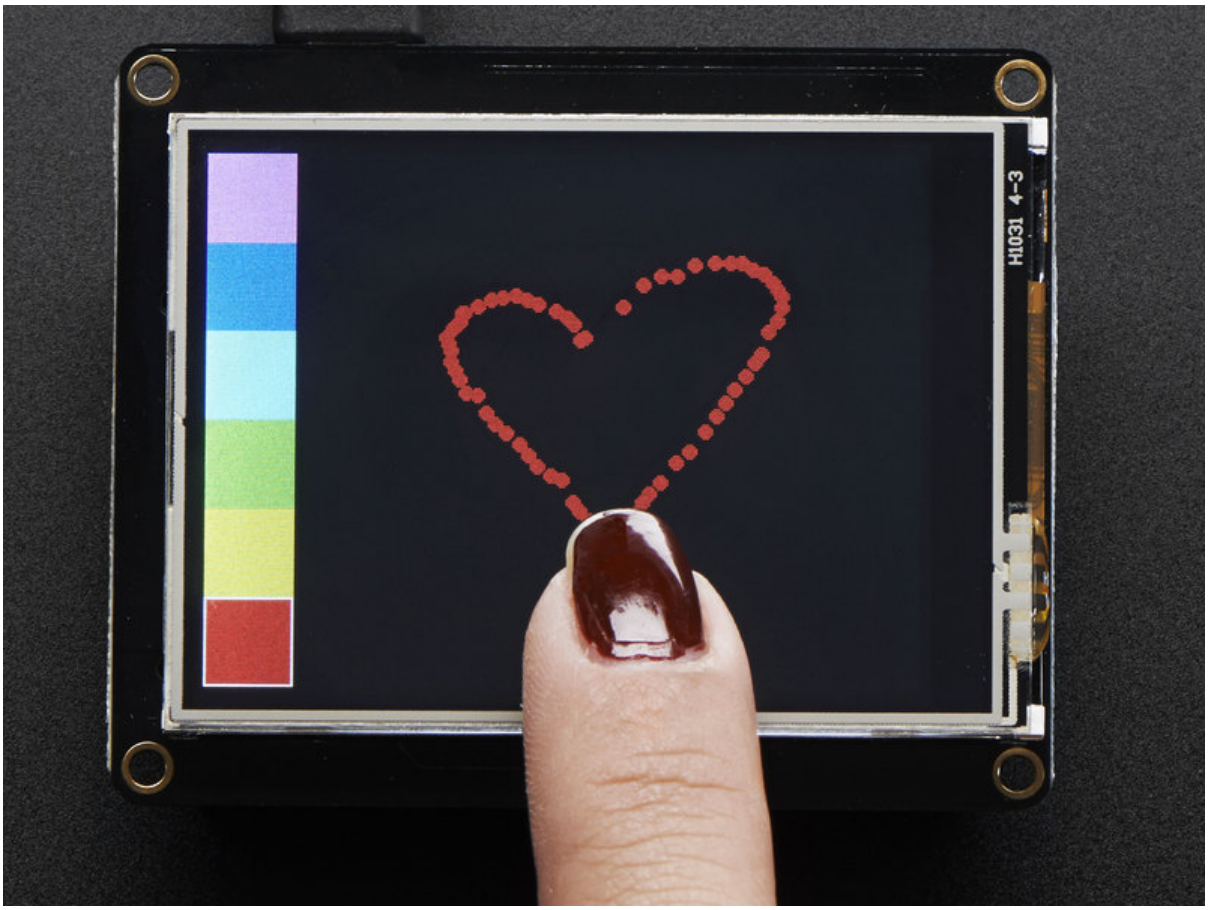
Adafruit 2.4" TFT FeatherWing

Created by lady ada

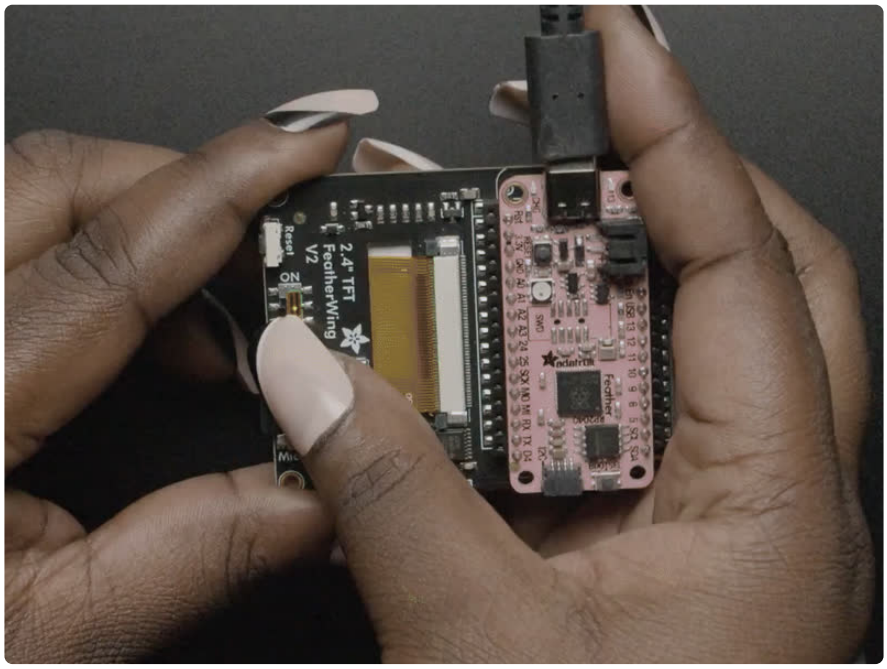


https://learn.adafruit.com/adafruit-2-4-tft-touch-screen-featherwing

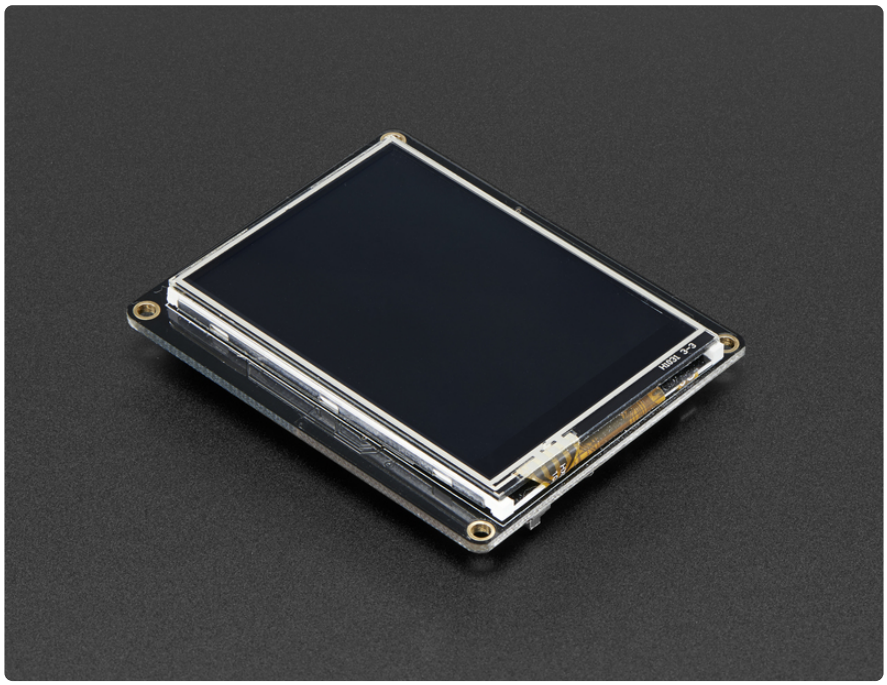
Last updated on 2025-03-21 11:02:18 PM EDT

Table of Contents

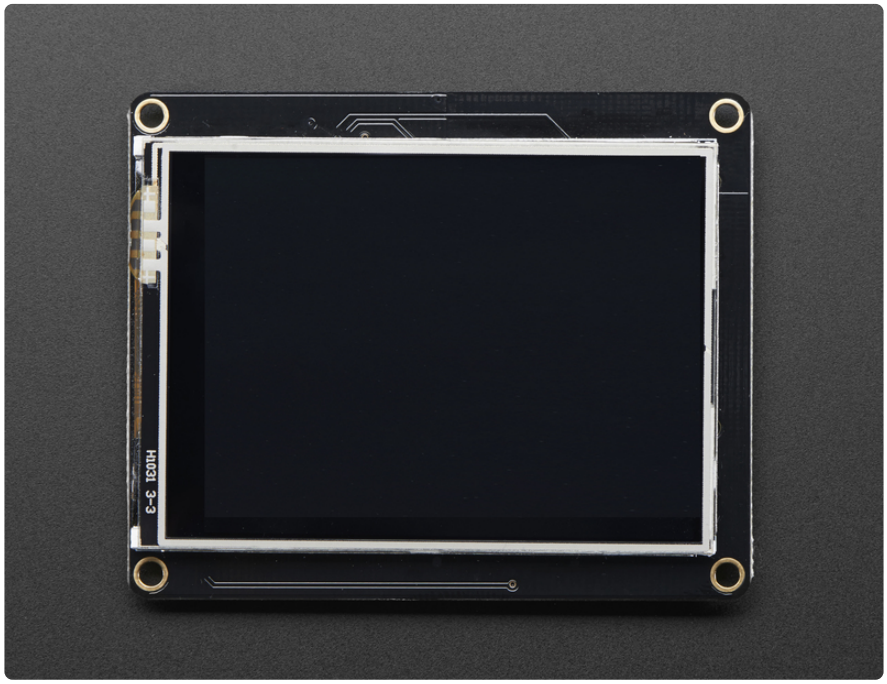
Overview



A Feather board without ambition is a Feather board without FeatherWings! Spice up your Feather project with a beautiful 2.4" touchscreen display shield with built in microSD card socket. This TFT display is 2.4" diagonal with a bright 4 white-LED backlight. You get 240x320 pixels with individual 16-bit color pixel control. It has way more resolution than a black and white 128x64 display. As a bonus, this display comes with a resistive touchscreen attached to it already, so you can detect finger presses anywhere on the screen.

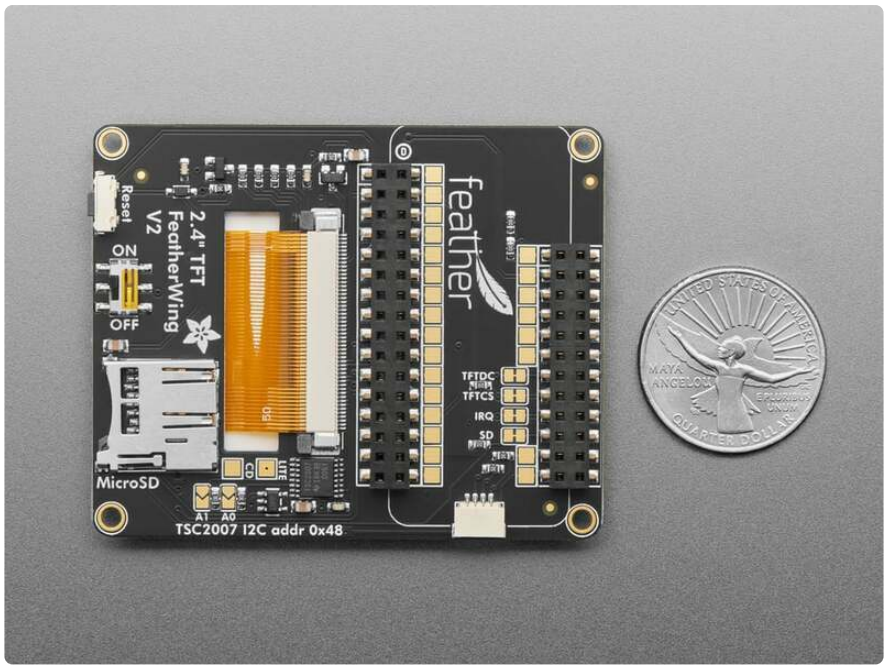


This FeatherWing uses a SPI display, touchscreen and SD card socket so it works nice and fast on all our Feathers (including **nRF52, ESP8266, 32u4, 328p, M0, M4, WICED** and **Teensy 3.x**) We also include an SPI resistive touchscreen controller so you only need one additional pin to add a high quality touchscreen controller. One more pin is used for an optional SD card that can be used for storing images for display.



This Wing comes fully assembled with dual sockets for your Feather to plug into. You get two sockets per pin so you can plug in wires if you want to connect to Feather pins. Alternatively, each pin has a large square pad on the PCB for direct soldering.

Four mounting holes make it easy to attach this Wing anywhere. We also include a big reset button and an on/off switch connected to the Feather Enable pin (note that the Teensy 3.x Feather does not use the Enable pin so the switch will not do anything with that type).



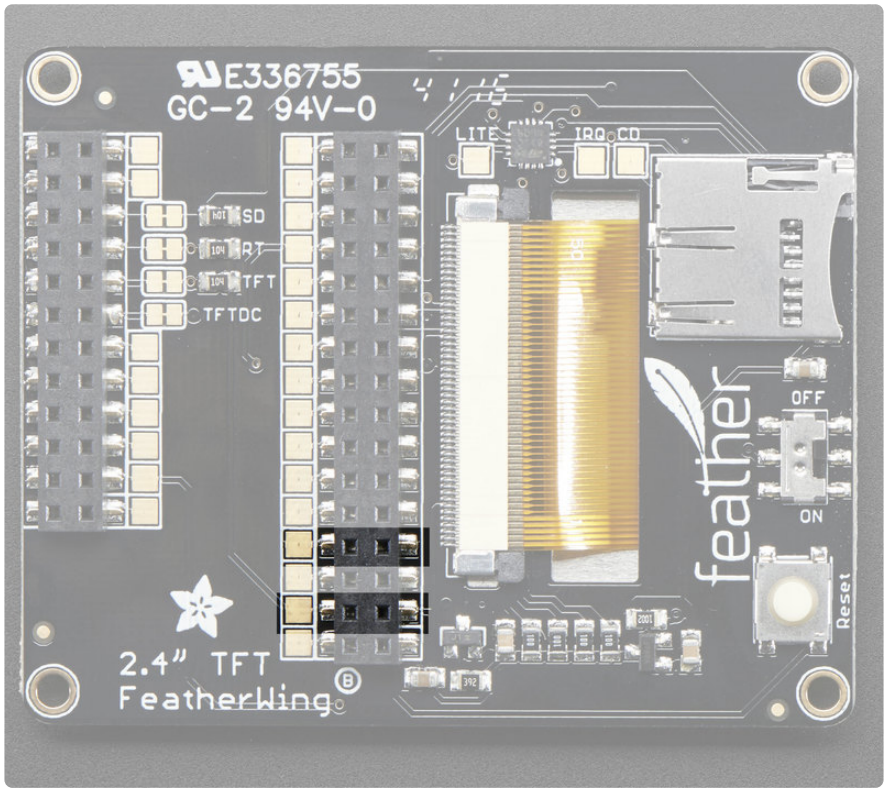
New! As of Oct 11, 2023 we've made a big redesign to this product: We now use the TSC2007 instead of the discontinued STMPE811 touchscreen controller. The screen and micro SD card are the same but any touchscreen code will need to be updated to use our Arduino or CircuitPython library. We've also updated the reset button to be right-angle and added a STEMMA QT port.

Pinouts - V1

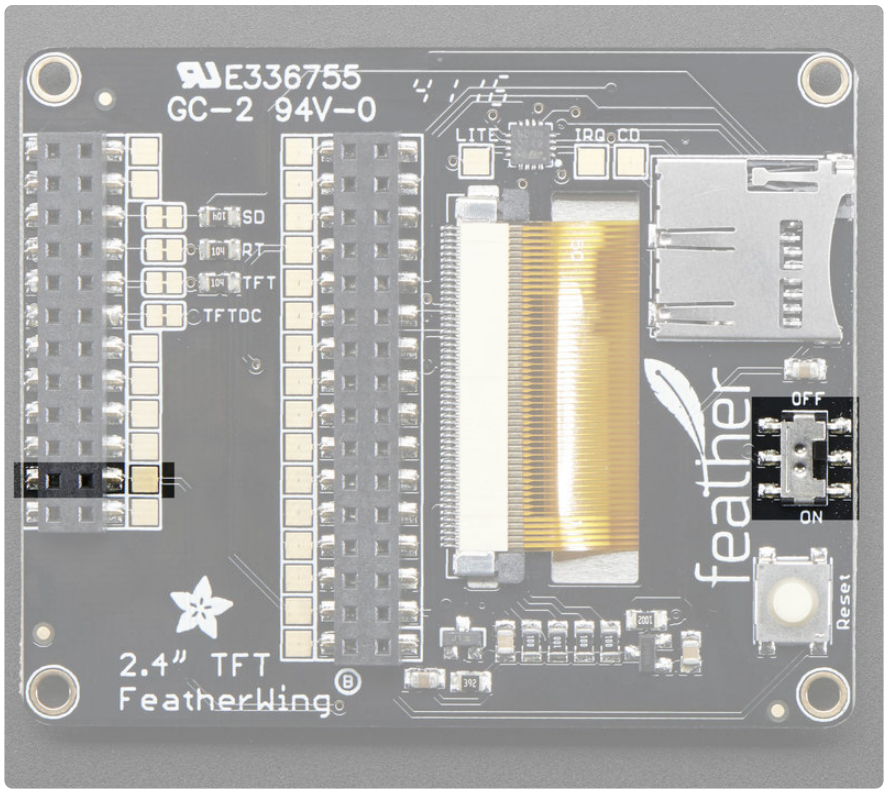
Unlike most FeatherWings, the TFT FeatherWing is fully assembled and has a dual socket set for your Feather to plug into.

This makes it really easy to use, but a little different to change if you don't want the default setup

Power Pins



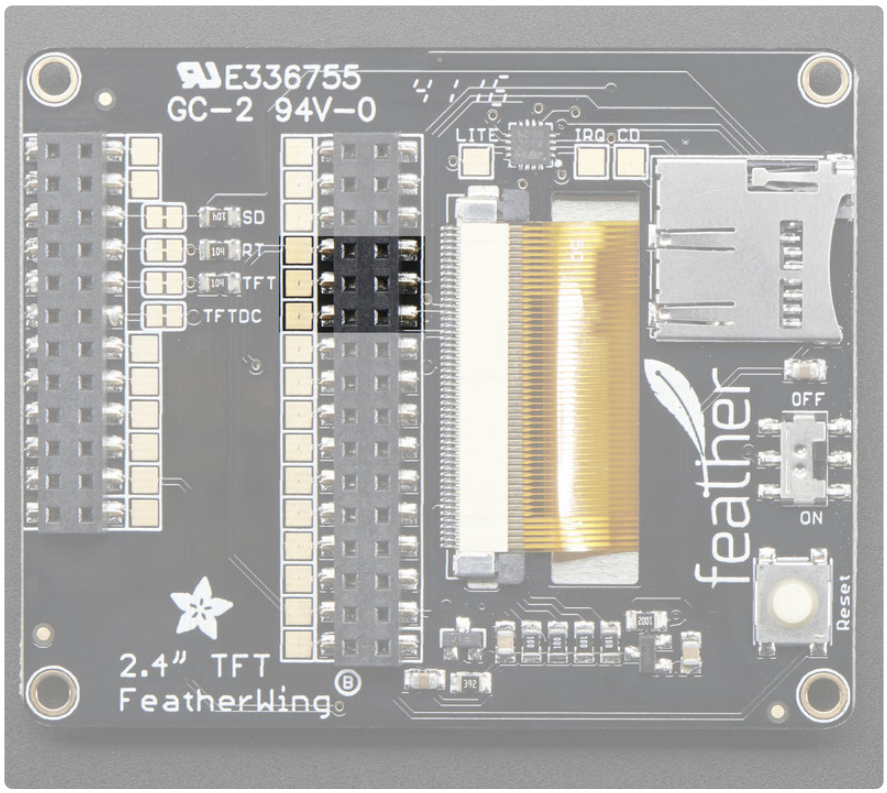
All power for the FeatherWing comes from the **3.3V** and **GND** pins. That includes the backlight (which can draw up to 100mA)!



You can turn off the 3.3V power supply with the **EN** pin or the switch attached to that pin. Note that on the Teensy 3x Feather Adapter, this pin doesn't do anything and on

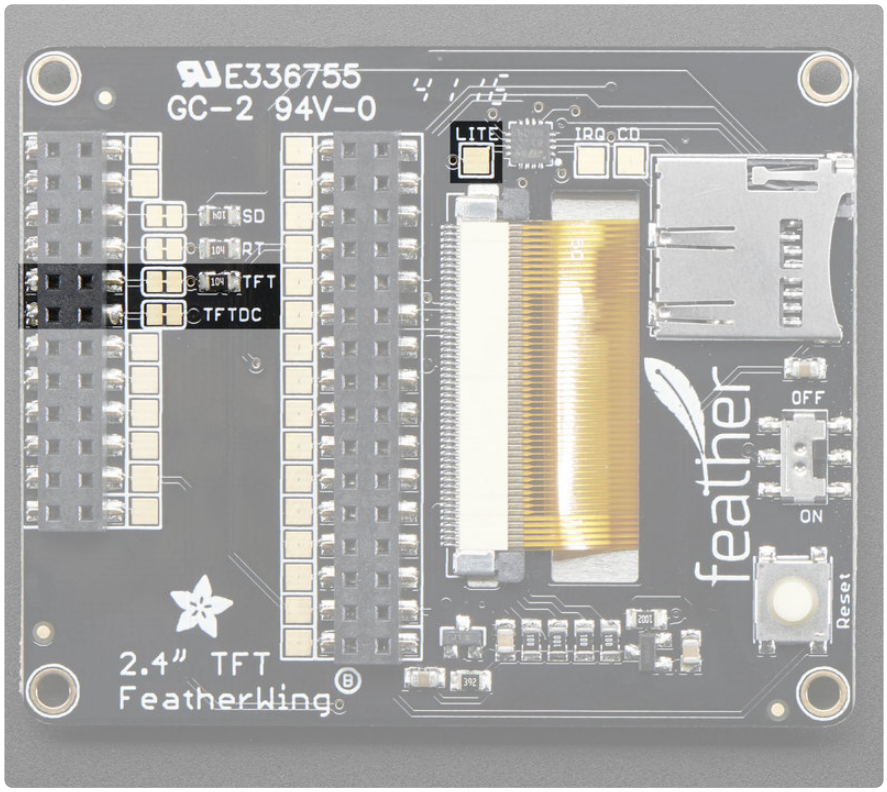
the FONA feather, this will not disable the VBAT power supply which is used to power the cellular module

SPI Pins



The TFT display, SD card and touch screen use the SPI interface to communicate. That means **MISO**, **MOSI** and **SCK** are used whenever either are accessed.

TFT Control Pins



In addition, for the TFT display there is are **D/C** (Data/Command) and **CS** (Chip Select) pins. These are used to select the display and tell it what kind of data is being sent. These pins can theoretically be changed by cutting the jumper trace and soldering a small wire from the right-hand pad to the pin you'd like to use.

On the **ESP8266**, TFT_CS is pin #0, TFT_DC is pin #15

On the **ESP32**, TFT_CS is pin #15, TFT_DC is pin #33

On the**Atmega32u4, ATmega328P, SAMD21 M0, nRF52840 or SAMD51 M4** Feather, TFT_CS is pin #9, TFT_DC is pin #10

On the **Teensy Feather**, TFT_CS is pin #4, TFT_DC is pin #10

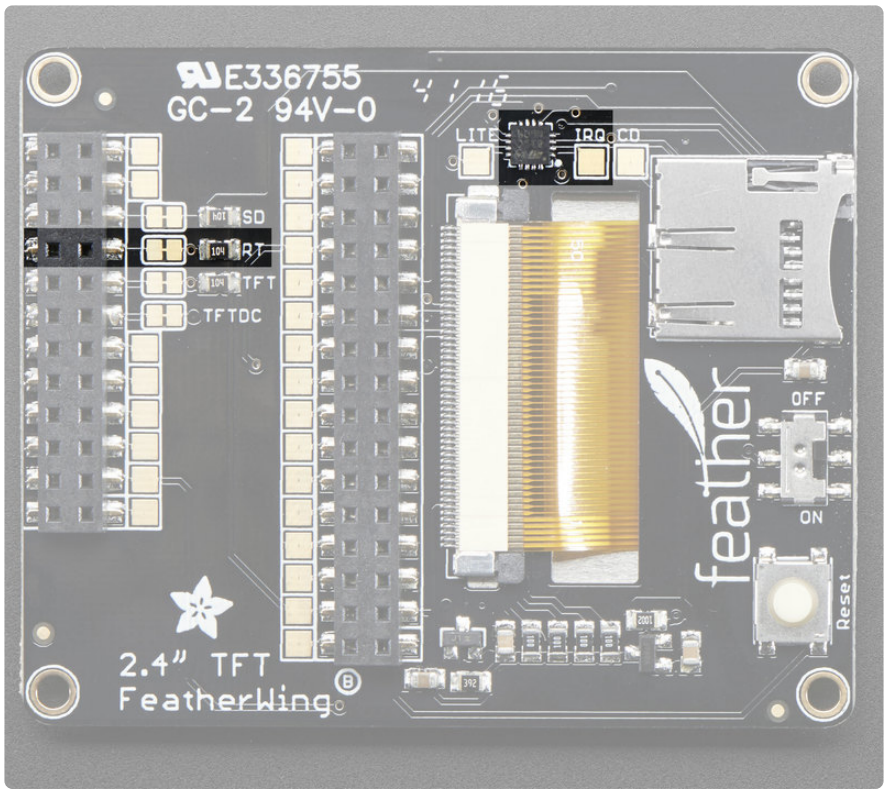
On the **WICED Feather**, TFT_CS is PA15 and TFT_DC is PB4

On the **nRF52832 Feather**, TFT_CS is #31 and TFT_DC is #11

There is also **LITE** pin which is not connected to any pads but you can use to control the backlight. Pull low to turn off the backlight. You can connect it to a PWM output pin.

Note: Pin 9 is used for communication with the SIM800 chip on the Feather Fona. You will have to remap pin 9 to an unused pin when using with a Feather Fona.

Touch Screen control pins



The touch screen also has a Chip Select line, labeled **RT**. This pin can theoretically be changed by cutting the jumper trace and soldering a small wire from the right-hand pad to the pin you'd like to use.

On the **ESP8266**, RT is pin #16

On the **ESP32**, RT is pin #32

On the**Atmega32u4, ATmega328P, nRF52840 SAMD21 M0 or SAMD51 M4** Feather, RT is pin #6

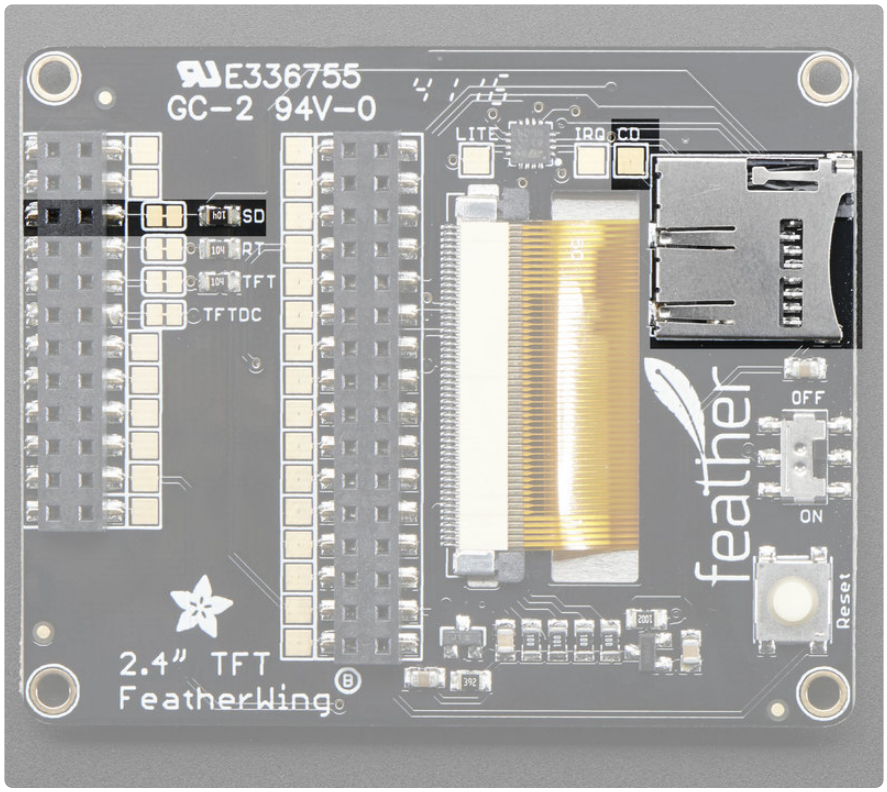
On the **Teensy Feather**, RT is pin #3

On the **WICED Feather**, RT is PC7

On the **nRF52832 Feather**, RT is #30

There is also an **IRQ** pin which is not connected to any pads but you can use to detect when touch events have occured.

SD Card control pins



The SD Card also has a Chip Select line, labeled **SD**. This pin can theoretically be changed by cutting the jumper trace and soldering a small wire from the right-hand pad to the pin you'd like to use.

On the **ESP8266**, SD is pin #2

On the **ESP32** SD is pin #14

On the **Atmega32u4, ATmega328P, nRF52840, SAMD21 M0 or SAMD51 M4** Feather, SD is pin #5

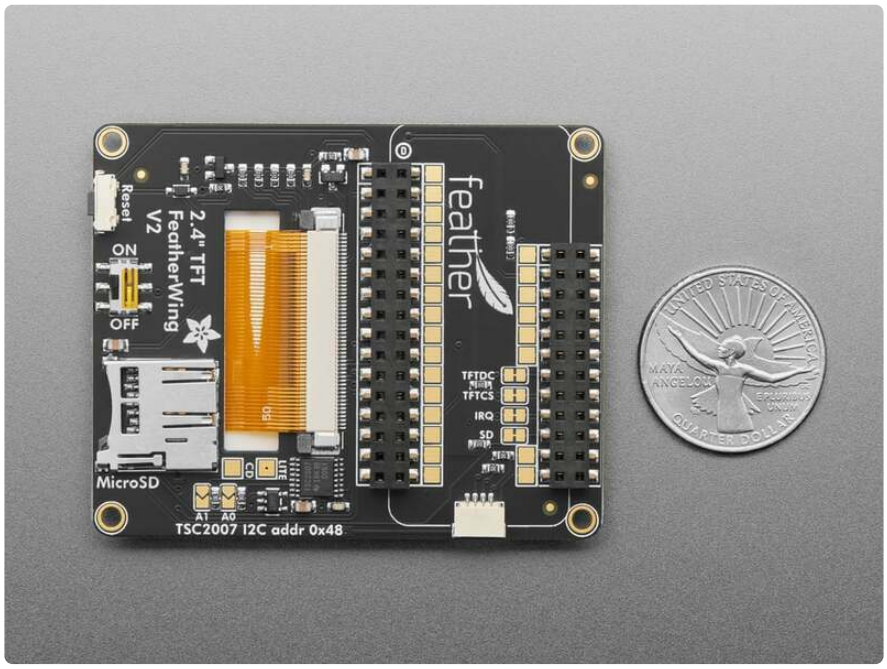
On the **Teensy Feather**, SD is pin #8

On the **WICED Feather**, SD is PC5

On the **nRF52 Feather**, SD is pin #27

There is also an **Card Detect (CD)** pin which is not connected to any pads but you can use to detect when a microSD card has been inserted have occured. It will be shorted to ground when a card is not inserted.

Pinouts - V2



The default I2C address for the TSC2007 touchscreen controller is **0x48**.

STEMMA QT Connector

- STEMMA QT **(https://adafru.it/Ft4)** - This connector, on the bottom of the FeatherWing, allows you to connect to sensors and breakout boards with STEMMA QT / Qwiic connectors or to other things with various associated accessories **(https://adafru.it/JRA)**.
- **SDA/SCL** - The I2C pins for the STEMMA QT connector are connected to the default I2C GPIO pins directly next to pin 5.
- **3.3V/GND** - The power for the STEMMA QT connector is 3.3V. Ground is the common ground for power and logic.

Default SPI Pins

The TFT (connected to an **ILI9341** chipset) and microSD card on the FeatherWing are controlled via SPI.

- **MOSI** - This is the SPI MOSI (**M**icrocontroller **O**ut / **S**erial **I**n) pin.
- **MISO** - This is the SPI MISO (**M**icrocontroller **I**n / **S**erial **O**ut) pin.
- **SCK** - This is the **S**PI clo**ck** input pin.

TFT Control Pins and Jumpers

- **TFTDC** - This is the display SPI **d**ata/**c**ommand selector pin. By default, it is connected to **pin 10**. To change this, cut the jumper. Then, connect the signal pad (closest to the label on the board silk) to one of the available and compatible GPIO pads.
- **TFTCS** - This is the TFT SPI **c**hip **s**elect pin. By default, is connected to **pin 9**. To change this, cut the jumper. Then, connect the signal pad (closest to the label on the board silk) to one of the available and compatible GPIO pads.
- **LITE** - This is the TFT backlight pad, located between the **CD** pad and the TSC2007 chip. It is not connected to any pins by default. Pull this pin low to turn off the backlight.

microSD Card Slot

- On the back of the FeatherWing, below the on/off switch, is the microSD card slot. You can use any microSD card that supports SPI mode with one CS pin.

SD Card Pins and Jumper

- **SD** - This is the SD card **c**hip **s**elect pin. By default, it is connected to **pin 5**. To change this, cut the jumper. Then, connect the signal pad (closest to the label on the board silk) to one of the available and compatible GPIO pads.
- **CD** - This is the **c**ard **d**etect pad, located to the left of the **LITE** pad and above the **A0** jumper. It is not connected to any pins by default. This pin will read low when a card is not inserted.

Touch Screen Interrupt Jumper

- **IRQ** - This is the touchscreen interrupt pin. By default, it is connected to **pin 6**. To change this, cut the jumper. Then, connect the signal pad (closest to the label on the board silk) to one of the available and compatible GPIO pads.

Settings an alternate I2C address on the TSC2007 requires a resistor across the jumper pad. Using a Pull-Up between 2k - 10k ohms is ideal. An 0805 SMD fits well on the jumper pad.

TSC2007 Address Jumpers

On the back of the board are **two address jumpers**, labeled **A0** and **A1,** to the right of the board name label on the silk. These jumpers allow you to change the default I2C

address of the TSC2007 on the FeatherWing. To do so, you solder the jumpers "closed" by connecting the two pads.

The default I2C address is **0x48**. The other address options can be calculated by “adding” the **A0/A1** to the base of **0x48**.

A0 sets the lowest bit with a value of **1**, **A1** sets the next bit with a value of **2**. The final address is **0x48** + **A1** + **A0** which would be **0x4B**.

If only **A0** is soldered closed, the address is **0x48** + **1** = **0x49**

If only **A1** is soldered closed, the address is **0x48** + **2** = **0x4A**

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

ADDR	A0	A1
0x48	L	L
0x49	H	L
0x4A	L	H
0x4B	H	H

Reset Button

- **Reset** - The reset button, located in the top left corner on the FeatherWing, is connected to the reset pin. It is mounted at a right angle so that it is easier to press.

On/Off Switch

- **ON/OFF** - The On/Off switch, located on the left side of the FeatherWing, is connected to the **EN pin**. You can disconnect 3.3V power by setting the switch to **OFF**.

TFT Graphics Test

The TFT FeatherWing is basically a combination of our 2.4" TFT Breakout (https://adafru.it/sjD) with the STMPE610 resistive touch-screen breakout attached (http://adafru.it/1571).

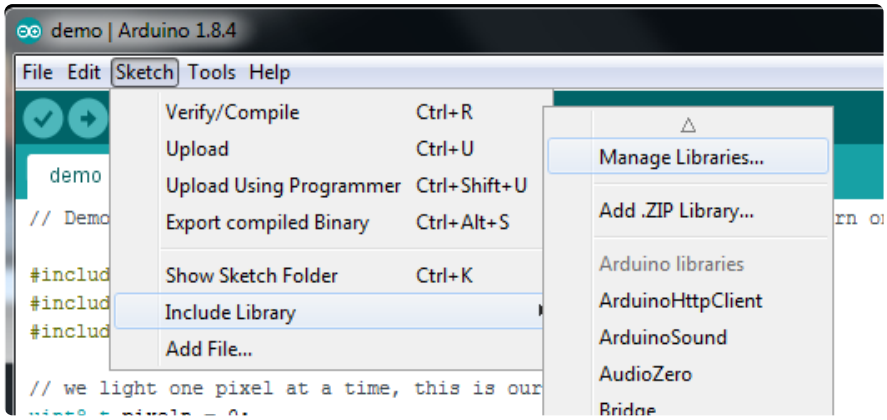
Install Libraries

You'll need a few libraries to use this FeatherWing!

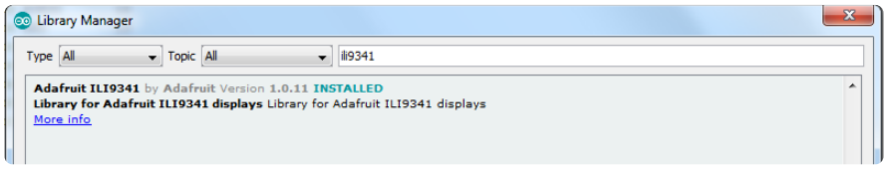
Install Adafruit ILI9341 TFT Library

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

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

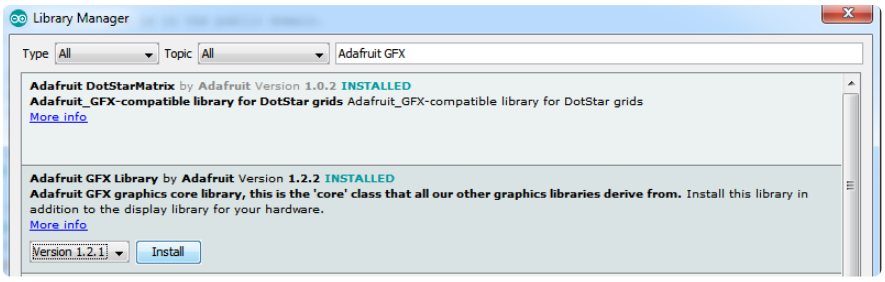


Search for **ILI9341** and install the **Adafruit ILI9341** library that pops up!



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!



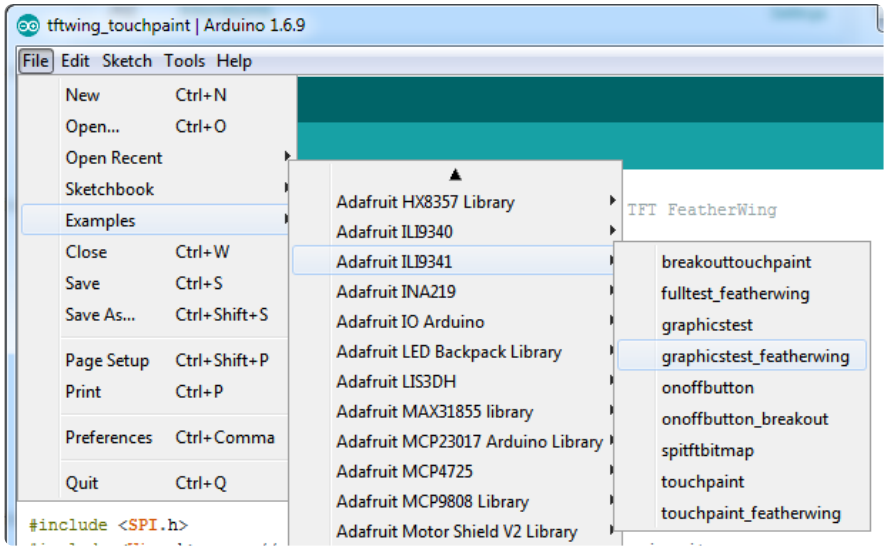
If using an earlier version of the Arduino IDE (pre-1.8.10), locate and install **Adafruit_BusIO** (newer versions handle this prerequisite automatically).

Repeat the search and install steps for the **Adafruit_ImageReader** library.

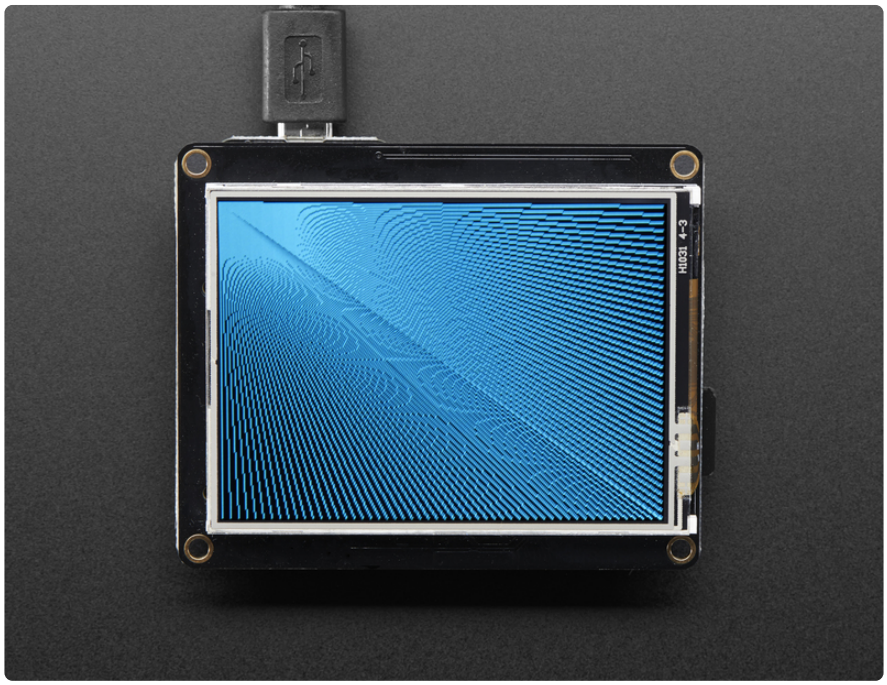
[For more details about this process, we have a tutorial introducing Arduino library concepts and installation](https://adafru.it/aYM) (https://adafru.it/aYM).

Basic Graphics Test

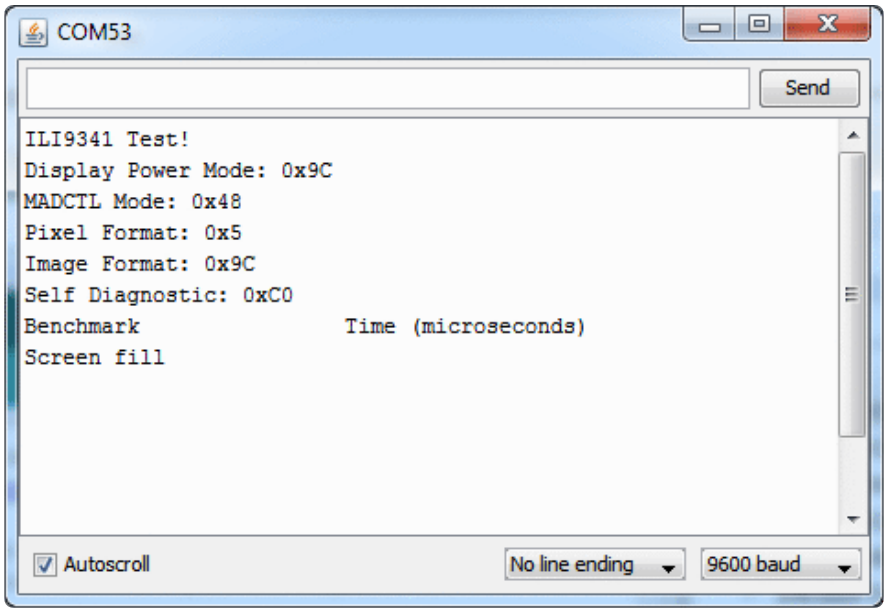
After installing these libraries, you should see a new **example** folder called **Adafruit_ILI9341** and inside, an example called **graphicstest_featherwing**.



Upload that sketch to your Feather. You should see a collection of graphical tests draw out on the TFT.

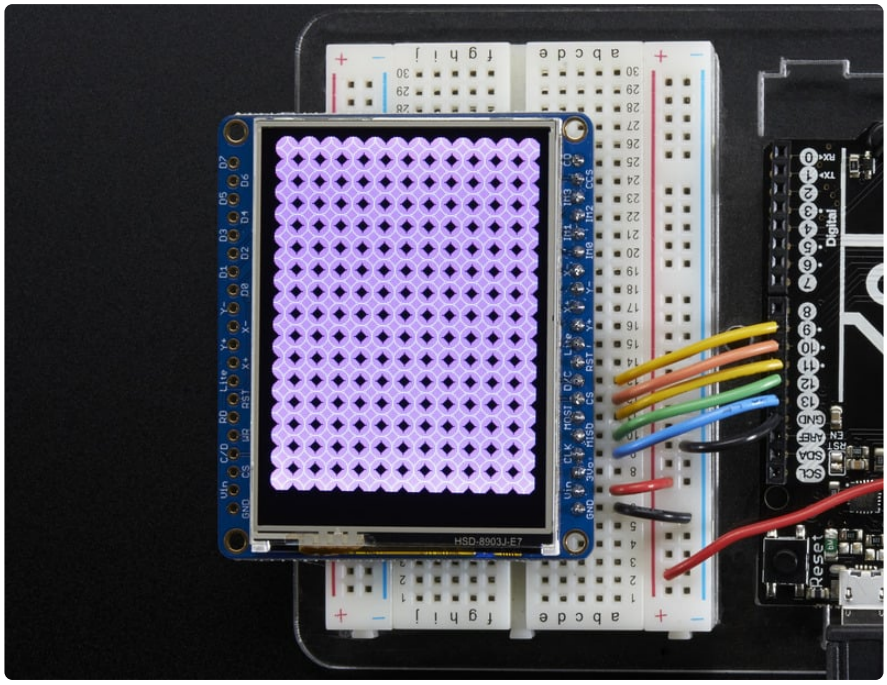


If you're having difficulties, check the serial console.The first thing the sketch does is read the driver configuration from the TFT, you should see the same numbers as below. That will help you determine if the TFT is found, if not, check your Feather soldering!



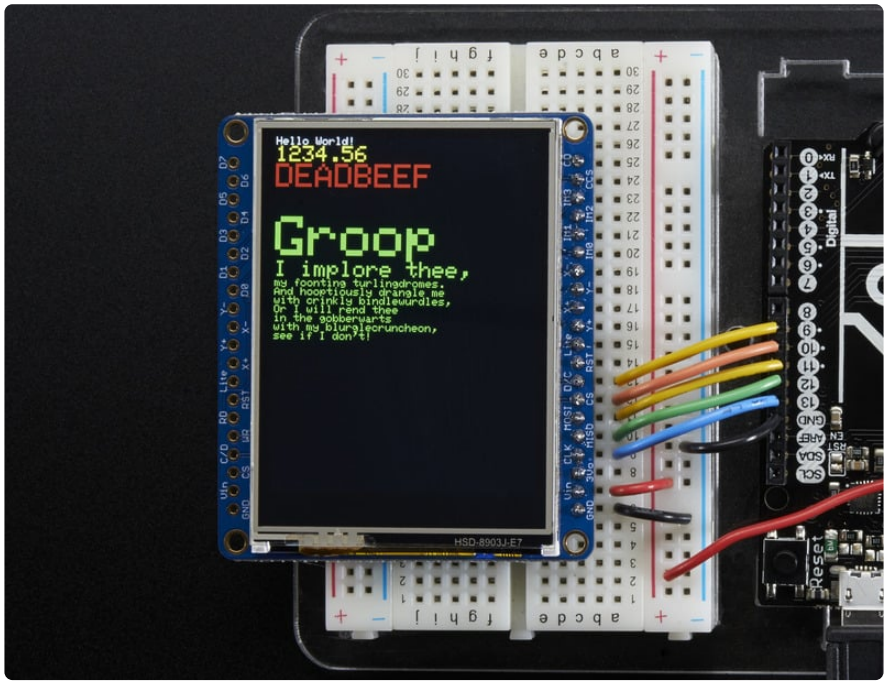
Once you've got the demo working, check out the detailed documentation over at http://learn.adafruit.com/adafruit-gfx-graphics-library (https://adafru.it/aPx) for more information on how to use the GFX library to draw whatever you like!

Adafruit GFX Library



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

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



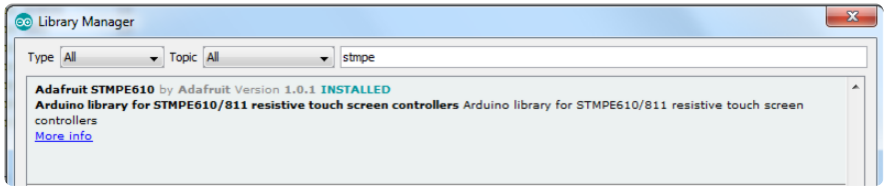
Check out our detailed tutorial here [http://learn.adafruit.com/adafruit-gfx-graphics-library](https://adafru.it/aPx)

It covers the latest and greatest of the GFX library. The GFX library is used in both 8-bit and SPI modes so the underlying commands (drawLine() for example) are identical!

Resistive Touch Screen - V1

The LCD has a 4-wire resistive touch screen glued onto it. You can use this for detecting finger-presses, stylus', etc. Normally, you'll need 4 pins to talk to the touch panel **but** we decided to go all snazzy and put a dedicated touch screen driver onto the shield. The driver shares the SPI pins with the TFT and SD card, so only one extra pin is needed. This allows you to query the controller when you're ready to read touchscreen data, and saves 3 pins.

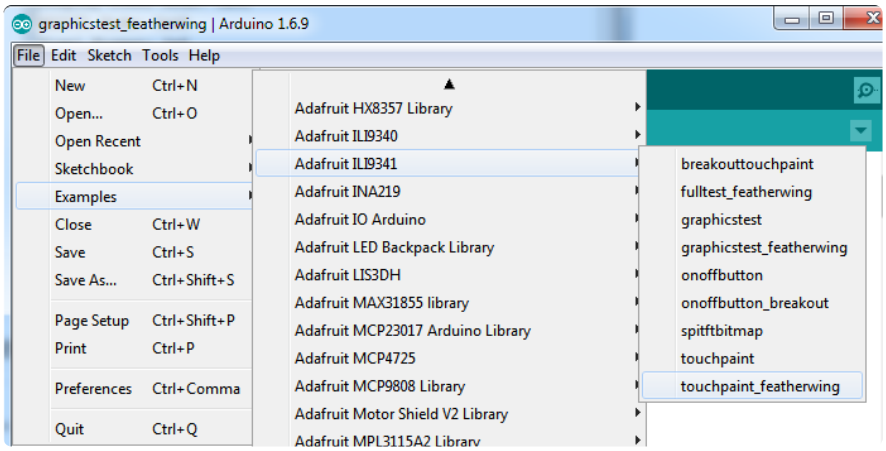
[To control the touchscreen you'll need one more library](https://adafru.it/d4f) - the STMPE610 controller library which does all the low level chatting with the STMPE610 driver chip. Use the library manager to install the **Adafruit STMPE610** library



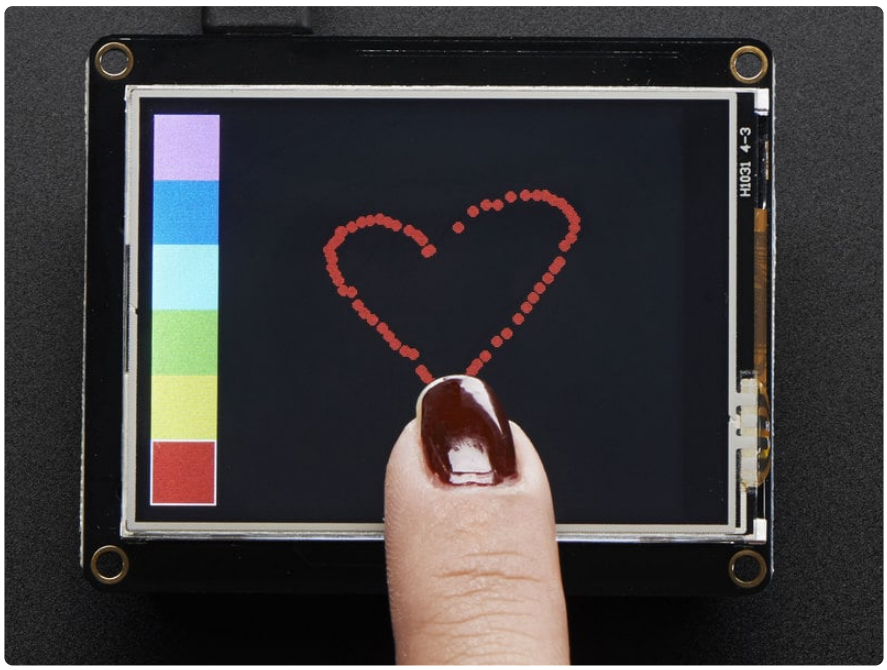
Touchscreen Paint Demo

Now that you've got the basic TFT graphics demo working, let's add in the touchscreen. Run and upload the **touchpaint_featherwing** demo

- If you have the **2.4" TFT** Featherwing, run the **Adafruit ILI9341->touchpaint_featherwing** demo
- If you have the **3.5" TFT** Featherwing, run the **Adafruit HX8357->touchpaint_featherwing** demo



Upload to your Feather and have fun!



The touch screen is made of a thin glass sheet, and its very fragile - a small crack or break will make the entire touch screen unusable. Don't drop or roughly handle the TFT and be especially careful of the corners and edges. When pressing on the touchscreen, sometimes people can use the tip of their fingers, or a fingernail. If you don't find the touchscreen responds well to your fingers, you can use a rounded stylus which will certainly work. Do not press harder and harder until the screen cracks!

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

```
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
```

We're using hardware SPI so the clock, mosi and miso pins are not defined here. Then you can start the touchscreen with

```
ts.begin()
```

Check to make sure this returns a True value, which means the driver was found. If it wasn't, make sure you have the Feather soldered right and the correct CS pin!

Now you can call

```
if (! ts.bufferEmpty())
```

to check if there's any data in the buffer. The touchscreen driver will store touchpoints at all times. When you're ready to get the data, just check if there's any data in the buffer. If there is, you can call

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

To get the oldest point from the buffer. TS_Point has **.x .y** and .**z** data points. The **x** and **y** points range from 0 to 4095. The STMPE610 does not store any calibration data in it and it doesn't know about rotation. **So if you want to rotate the screen you'll need to manually rotate the x/y points!** The z point is 'pressure' and ranges from 0 to 255, we don't use it here but you can experiment with it on your own, the harder you press, the lower the number.

Since data from the STMPE610 comes in 0-4095 but our screen is 320 pixels by 240 pixels, we can use **map** to convert 0-4095 to 0-320 or 0-240. Something like

```
p.x = map(p.x, 0, 4095, 0, tft.width());
p.y = map(p.y, 0, 4095, 0, tft.height());
```

However, the touchscreen is a bit bigger than the screen, so we actually need to ignore presses beyond the touchscreen itself. We found that these numbers reflected the true range that overlaps the screen

```
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000
```

So we use

```
p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
```

instead.

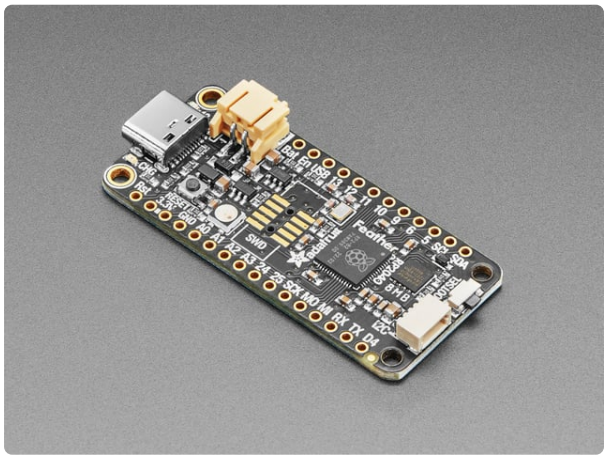
One last point (pun intended!) since the touchscreen driver stores points in a buffer, you may want to ask the driver "is the touchscreen being pressed RIGHT NOW?" You can do that with

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

Resistive Touch Screen - V2

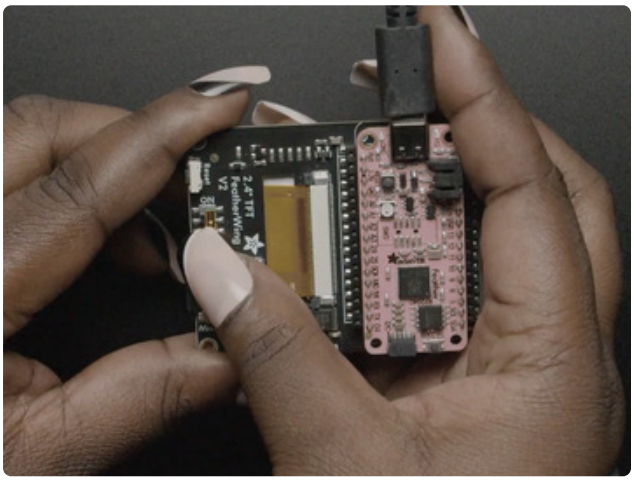
Using the 2.4" TFT FeatherWing V2 with Arduino involves plugging a Feather board into the FeatherWing. Then, you'll install the necessary libraries and upload the example code to the Feather board.

This page uses the Feather RP2040 for demonstrating Arduino usage. You can use the same concepts to get going with any Feather board.



Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884

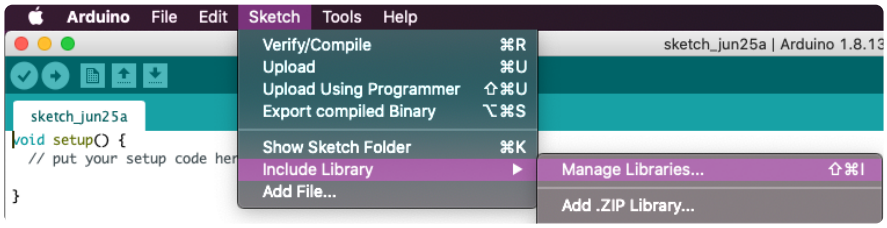
Hardware Setup



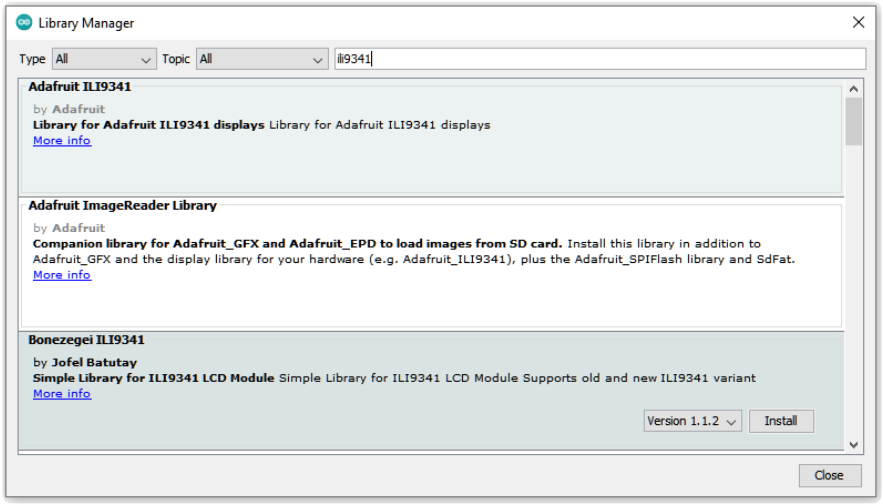
Plug your Feather into the FeatherWing. The bottom of the Feather will be above the FeatherWing STEMMA QT port.

Library Installation

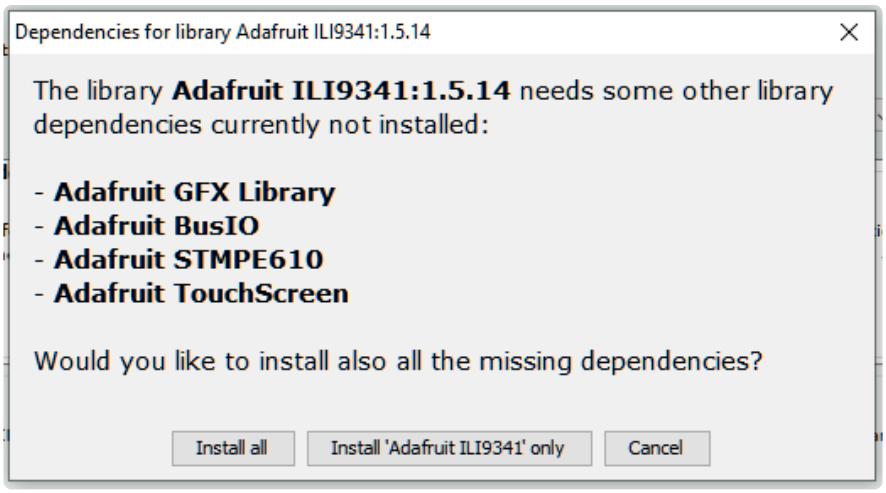
You can install the **Adafruit ILI9341** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries...** menu item, search for **Adafruit ILI9341**, and select the **Adafruit_ILI9341** library:

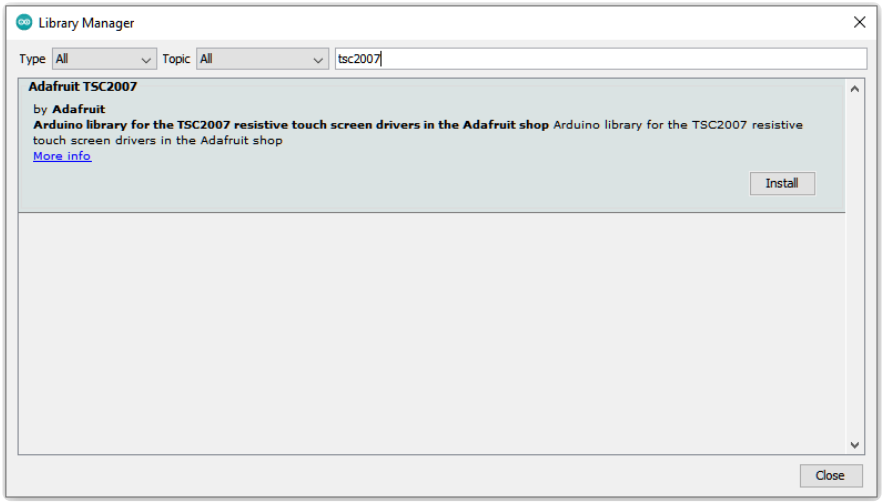


If asked about dependencies, click "Install all".

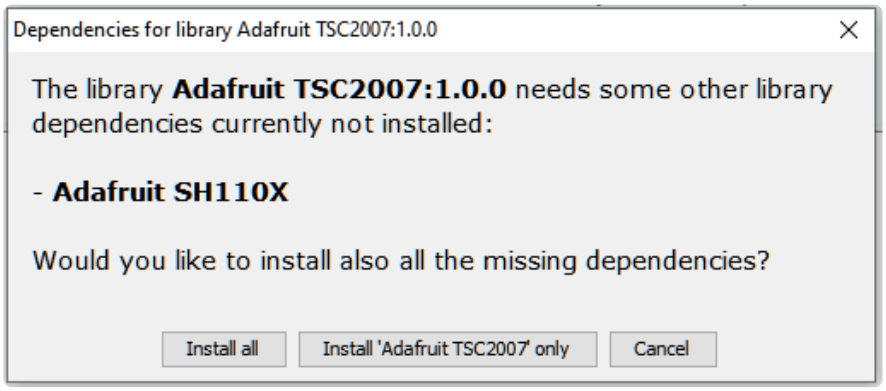


If the "Dependencies" window does not come up, then you already have the dependencies installed.

Then you'll install the Adafruit TSC2007 library for the touch screen. Click the **Manage Libraries...** menu item, search for **Adafruit TSC2007**, and select the **Adafruit TSC2007** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

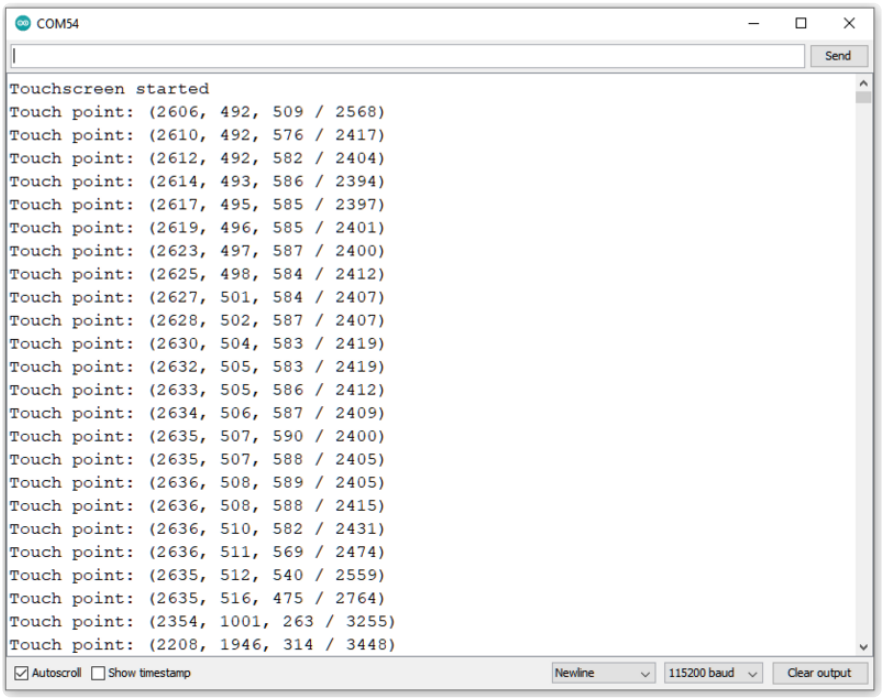
> If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

```
// SPDX-FileCopyrightText: 2023 Limor Fried/Ladyada for Adafruit Industries
// SPDX-License-Identifier: MIT

/*************************************************
  This is our touchscreen painting example for the updated Adafruit
  TFT FeatherWing V2 with TSC2007
  ----> http://www.adafruit.com/products/3315

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 *************************************************/


#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_TSC2007.h>

// This is calibration data for the raw touch data to the screen coordinates
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000
#define TS_MIN_PRESSURE 200

Adafruit_TSC2007 ts;

// The display also uses hardware SPI, plus #9 & #10
#define TFT_CS 9
#define TFT_DC 10
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// Size of the color selection boxes and the paintbrush size
#define BOXSIZE 40
#define PENRADIUS 3
int oldcolor, currentcolor;

void setup(void) {

  Serial.begin(115200);
  // while (!Serial) delay(10);

  tft.begin();

  if (!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
```

```
    while (1);
  }
  Serial.println("Touchscreen started");

  tft.fillScreen(ILI9341_BLACK);

  // make the color selection boxes
  tft.fillRect(0, 0, BOXSIZE, BOXSIZE, ILI9341_RED);
  tft.fillRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, ILI9341_YELLOW);
  tft.fillRect(BOXSIZE*2, 0, BOXSIZE, BOXSIZE, ILI9341_GREEN);
  tft.fillRect(BOXSIZE*3, 0, BOXSIZE, BOXSIZE, ILI9341_CYAN);
  tft.fillRect(BOXSIZE*4, 0, BOXSIZE, BOXSIZE, ILI9341_BLUE);
  tft.fillRect(BOXSIZE*5, 0, BOXSIZE, BOXSIZE, ILI9341_MAGENTA);

  // select the current color 'red'
  tft.drawRect(0, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
  currentcolor = ILI9341_RED;
}

void loop(){
  uint16_t x, y, z1, z2;
  if (ts.read_touch(&x, &y, &z1, &z2) && (z1 > TS_MIN_PRESSURE)) {

    Serial.print("Touch point: (");
    Serial.print(x); Serial.print(", ");
    Serial.print(y); Serial.print(", ");
    Serial.print(z1); Serial.print(" / ");
    Serial.print(z2); Serial.println(")");

    // Scale from ~0->4000 to tft.width using the calibration #'s
    x = map(x, TS_MINX, TS_MAXX, 0, tft.width());
    y = map(y, TS_MINY, TS_MAXY, 0, tft.height());

    if (y < BOXSIZE) {
       oldcolor = currentcolor;

       if (x < BOXSIZE) {
         currentcolor = ILI9341_RED;
         tft.drawRect(0, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       } else if (x < BOXSIZE*2) {
         currentcolor = ILI9341_YELLOW;
         tft.drawRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       } else if (x < BOXSIZE*3) {
         currentcolor = ILI9341_GREEN;
         tft.drawRect(BOXSIZE*2, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       } else if (x < BOXSIZE*4) {
         currentcolor = ILI9341_CYAN;
         tft.drawRect(BOXSIZE*3, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       } else if (x < BOXSIZE*5) {
         currentcolor = ILI9341_BLUE;
         tft.drawRect(BOXSIZE*4, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       } else if (x < BOXSIZE*6) {
         currentcolor = ILI9341_MAGENTA;
         tft.drawRect(BOXSIZE*5, 0, BOXSIZE, BOXSIZE, ILI9341_WHITE);
       }

       if (oldcolor != currentcolor) {
          if (oldcolor == ILI9341_RED)
            tft.fillRect(0, 0, BOXSIZE, BOXSIZE, ILI9341_RED);
          if (oldcolor == ILI9341_YELLOW)
            tft.fillRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, ILI9341_YELLOW);
          if (oldcolor == ILI9341_GREEN)
            tft.fillRect(BOXSIZE*2, 0, BOXSIZE, BOXSIZE, ILI9341_GREEN);
          if (oldcolor == ILI9341_CYAN)
            tft.fillRect(BOXSIZE*3, 0, BOXSIZE, BOXSIZE, ILI9341_CYAN);
          if (oldcolor == ILI9341_BLUE)
            tft.fillRect(BOXSIZE*4, 0, BOXSIZE, BOXSIZE, ILI9341_BLUE);
          if (oldcolor == ILI9341_MAGENTA)
            tft.fillRect(BOXSIZE*5, 0, BOXSIZE, BOXSIZE, ILI9341_MAGENTA);
```

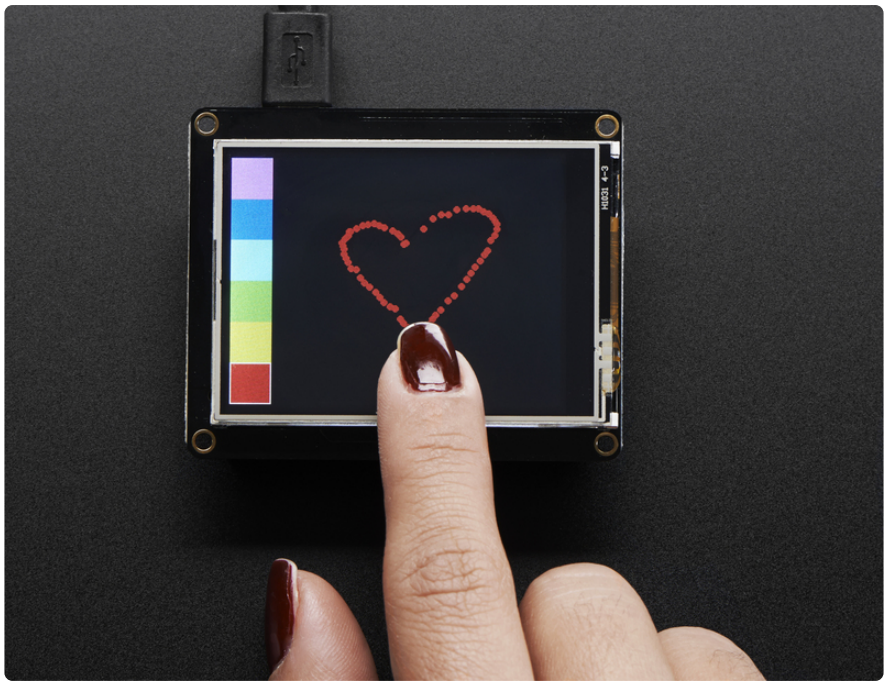
```
        }
      }
      if (((y-PENRADIUS) > BOXSIZE) && ((y+PENRADIUS) < tft.height())) {
        tft.fillCircle(x, y, PENRADIUS, currentcolor);
      }
    }
  }
```



Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. In the Serial Monitor, you should see the values from the touch screen being printed out.

The first number is the X coordinate, the second number is the Y coordinate and the last two numbers are Z "pressure" coordinates that can tell you how hard the touch pad is being pressed.

On the TFT, you'll be able to doodle with the different colors on the left side of the screen.

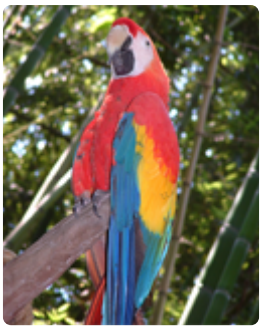


Drawing Bitmaps

There is a built-in microSD card slot on the FeatherWing, 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), and an SD card reader on whatever computer you're currently reading this with.

Its really easy to draw bitmaps. Lets start by downloading this image of pretty flowers:

Download these two smaller images as well:



The files should be renamed (if needed) to “**purple.bmp**”, “**parrot.bmp**” and “**wales.bmp**”, respectively, and copied to the base directory of the microSD card (not inside a folder).

(If it’s easier, you can also find these images in the “images” folder within the Adafruit_ImageReader library folder.)

Insert the microSD card into the socket in the shield. Now select the sketch **file→examples→Adafruit_ImageReader→FeatherWingILI9341** and upload this example to your Feather + Wing. You will see the flowers appear! (Plus parrots...and if you’re using one of the more powerful Feather boards, a whole lot of dragons.)

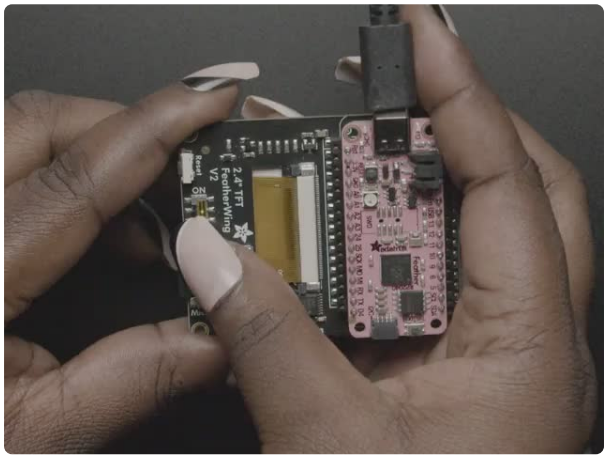
The Adafruit_ImageReader library, which is being used here to display .BMP images, is fully explained in its own page of the Adafruit_GFX guide (https://adafru.it/DpM).

CircuitPython Displayio Quickstart - V1

We'll start with the 2.4" TFT FeatherWing which has an ILI9341 display on it. If you would like more information on this display, be sure to check out our Adafruit 2.4" TFT FeatherWing guide (https://adafru.it/vvE).

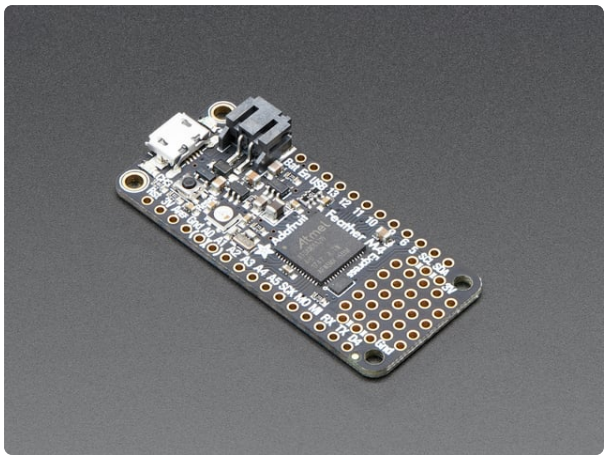
Parts

To use this display with displayio, you will only need two main parts. First, you will need the TFT FeatherWing itself.



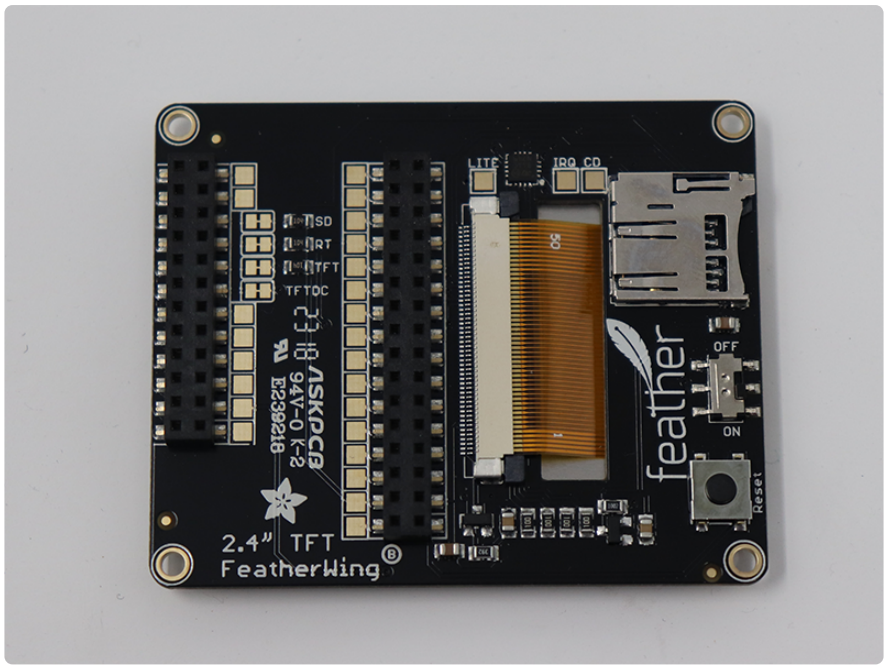
TFT FeatherWing - 2.4" 320x240 Touchscreen For All Feathers
A Feather board without ambition is a Feather board without FeatherWings! Spice up your Feather project with a beautiful 2.4" touchscreen display shield with built in microSD card...
https://www.adafruit.com/product/3315

And second, you will need a Feather such as the Feather M0 Express or the Feather M4 Express. We recommend the Feather M4 Express because it's much faster and works better for driving a display.

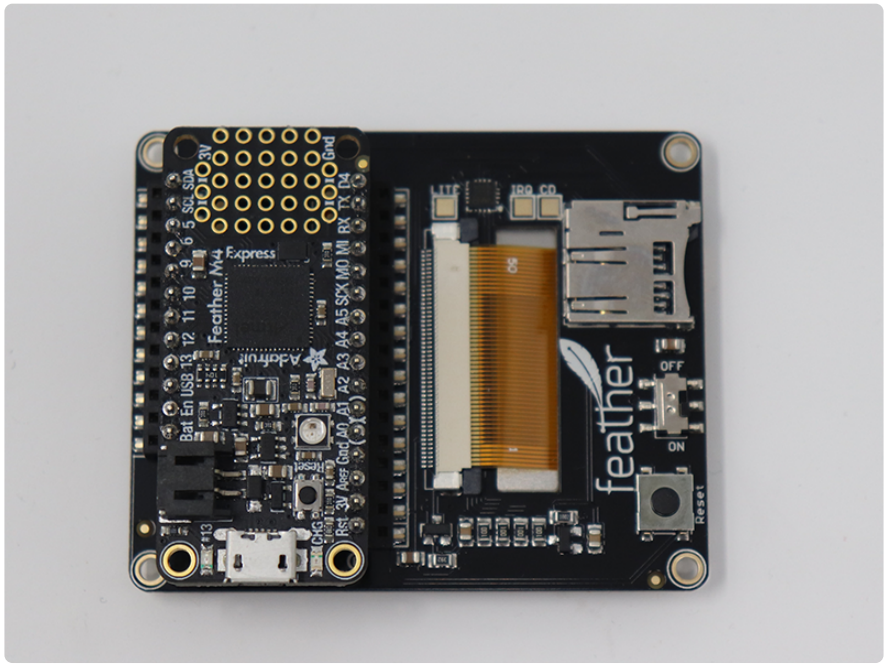


Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857

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://adafru.it/EEm). We'll start by looking at the back of the 2.4" TFT FeatherWing.



After that, it's just a matter of inserting the Feather M4 Express into the back of the TFT FeatherWing.



Required CircuitPython Libraries

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

Adafruit_CircuitPython_ILI9341

https://adafru.it/EGe

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.

Adafruit_CircuitPython_Display_Text

https://adafru.it/FiA

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

```
# 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. All drawing is done
using native displayio modules.

Pinouts are for the 2.4" TFT FeatherWing or Breakout with a Feather M4 or M0.
"""

import board
import displayio
import terminalio
from adafruit_display_text import label
from fourwire import FourWire

import adafruit_ili9341

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

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

```
display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs, reset=board.D6)
display = adafruit_ili9341.ILI9341(display_bus, width=320, height=240)

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

# Draw a green background
color_bitmap = displayio.Bitmap(320, 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(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)

# Draw a label
text_group = displayio.Group(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)

while True:
    pass
```

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.

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

Next we release any previously used 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.

```
displayio.release_displays()
```

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.

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

In the next line, we set the display bus to FourWire which makes use of the SPI bus. The reset parameter is actually not needed for the FeatherWing, but was added to make it compatible with the breakout displays. You can either leave it or remove it if you need access to an additional GPIO pin.

```
display_bus = fourwire.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.

```
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. The display will automatically handle updating the group.

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

```
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.

```
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.

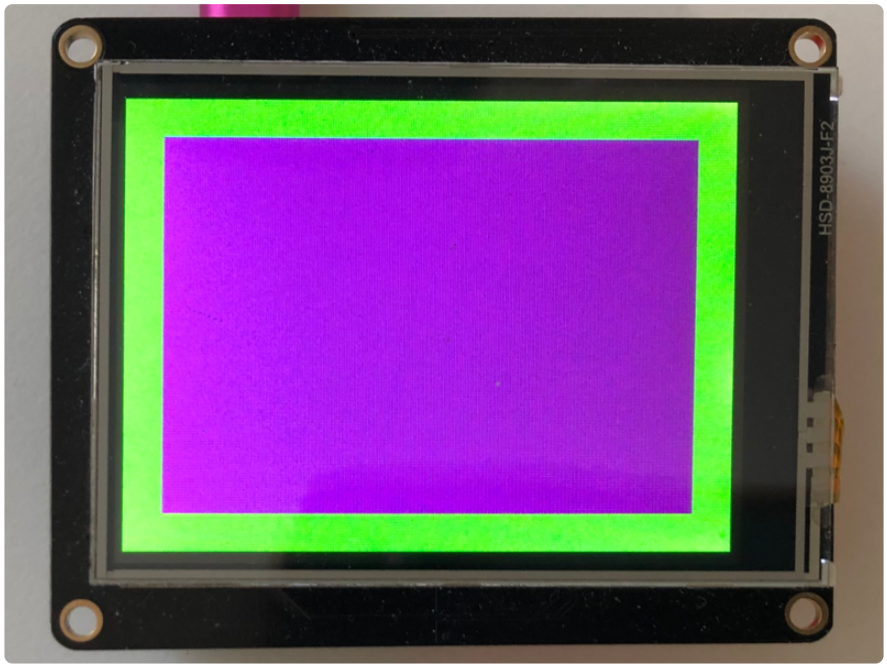
Next we will create a smaller purple rectangle. The easiest way to do this is the create a new bitmap that is a little smaller than the full screen with a single color and place it in a specific location. In this case we will create a bitmap that is 20 pixels smaller on each side. The screen is 320x240, 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.

```
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 `0xFFFF00`.

```
text_group = displayio.Group(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.

```
while True:
    pass
```

Using Touch

We won't be covering how to use the touchscreen on the shield with CircuitPython in this guide, but the library required for enabling resistive touch is the Adafruit_CircuitPython_STMPE610 (https://adafru.it/Fsz) library.

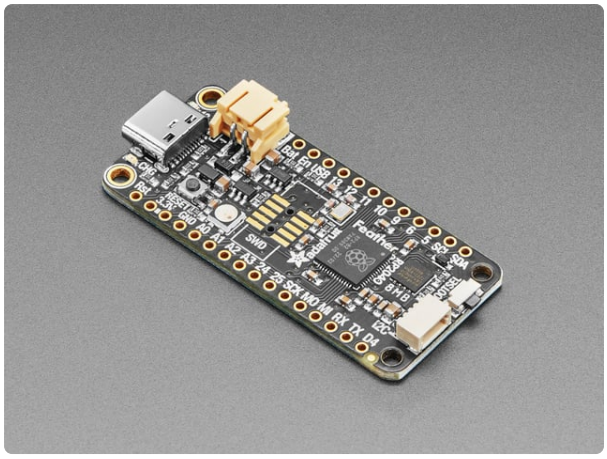
Where to go from here

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

CircuitPython Displayio Quickstart - V2

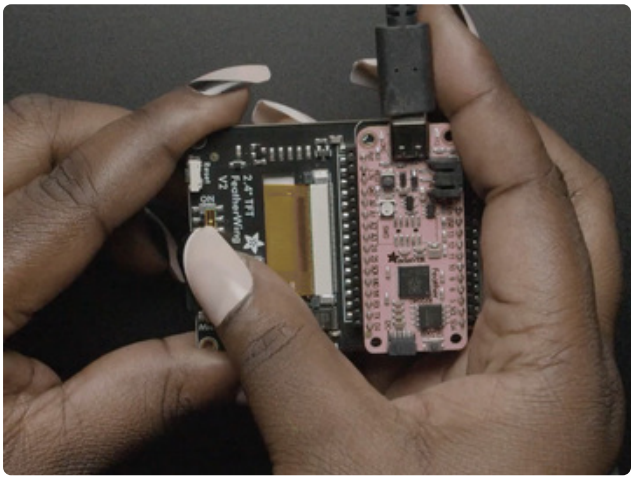
Using the 2.4" TFT FeatherWing V2 with CircuitPython involves plugging a Feather board into the FeatherWing. Then, you load the code and necessary libraries onto your Feather board to run the example.

This page uses the Feather RP2040 for demonstrating CircuitPython usage. You can use the same concepts to get going with any classic Feather board.



Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884

Hardware Setup



Plug your Feather into the FeatherWing. The bottom of the Feather will be above the FeatherWing STEMMA QT port.

CircuitPython Usage

To use with CircuitPython, you need to first install the necessary libraries, and their dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

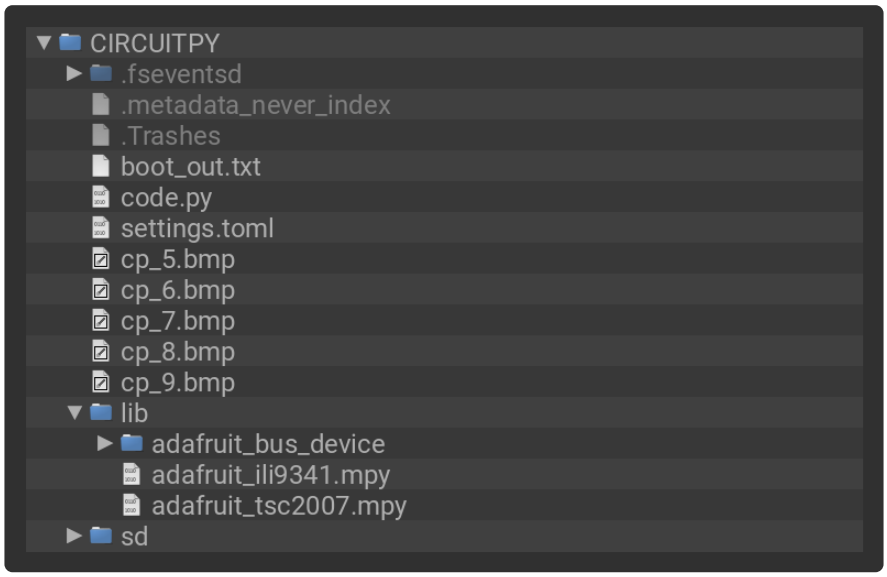
Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file.

Connect the microcontroller to your computer via a known-good USB power+data cable. The board shows up as a thumb drive named **CIRCUITPY**. Copy the **entire lib folder**, the **bitmap image files**, and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folder and files:

- **/adafruit_bus_device**
- **adafruit_ili9341.mpy**
- **adafruit_tsc2007.mpy**

Once you have copied over the necessary folders and files, your **CIRCUITPY** drive should resemble the following:



Example Code

```
# SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This test will initialize the display using displayio and display
a bitmap image. The image advances when the touch screen is touched.

Pinouts are for the 2.4" TFT FeatherWing V2
"""
import os
import board
import displayio
import fourwire
import adafruit_ili9341
import adafruit_tsc2007

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

# Use Hardware SPI
spi = board.SPI()

tft_cs = board.D9
tft_dc = board.D10

display_width = 320
display_height = 240

display_bus = fourwire.FourWire(spi, command=tft_dc, chip_select=tft_cs)
display = adafruit_ili9341.ILI9341(display_bus, width=display_width,
height=display_height)

i2c = board.STEMMA_I2C()

irq_dio = None
tsc = adafruit_tsc2007.TSC2007(i2c, irq=irq_dio)

groups = []
images = []
for filename in os.listdir('/'):
```

```
    if filename.lower().endswith('.bmp') and not filename.startswith('.'):
        images.append("/"+filename)
print(images)

for i in range(len(images)):
    splash = displayio.Group()
    bitmap = displayio.OnDiskBitmap(images[i])
    tile_grid = displayio.TileGrid(bitmap, pixel_shader=bitmap.pixel_shader)
    splash.append(tile_grid)
    groups.append(splash)

index = 0
touch_state = False

display.root_group = groups[index]

while True:
    if tsc.touched and not touch_state:
        point = tsc.touch
        print("Touchpoint: (%d, %d, %d)" % (point["x"], point["y"],
point["pressure"]))
        # left side of the screen
        if point["y"] < 2000:
            index = (index - 1) % len(images)
            display.root_group = groups[index]
        # right side of the screen
        else:
            index = (index + 1) % len(images)
            display.root_group = groups[index]
        touch_state = True
    if not tsc.touched and touch_state:
        touch_state = False
```

Once everything is saved to the **CIRCUITPY** drive, [connect to the serial console](https://adafru.it/Bec) to see the data printed out!

The code will open all of the bitmap files that are on the **CIRCUITPY** drive and add them to the `images` array. The first image will then be displayed on the TFT. In the loop, if you touch the screen you'll see the coordinates and pressure print to the serial console.

If you press on the left side of the screen, the display will show the previous bitmap in the array. If you press on the right side of the screen, the display will show the next bitmap in the array.

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.

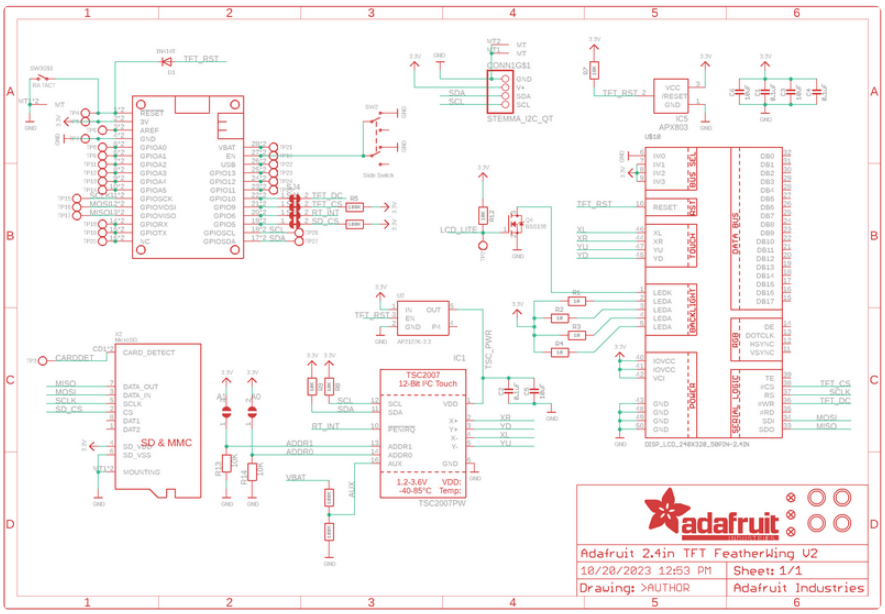
Downloads

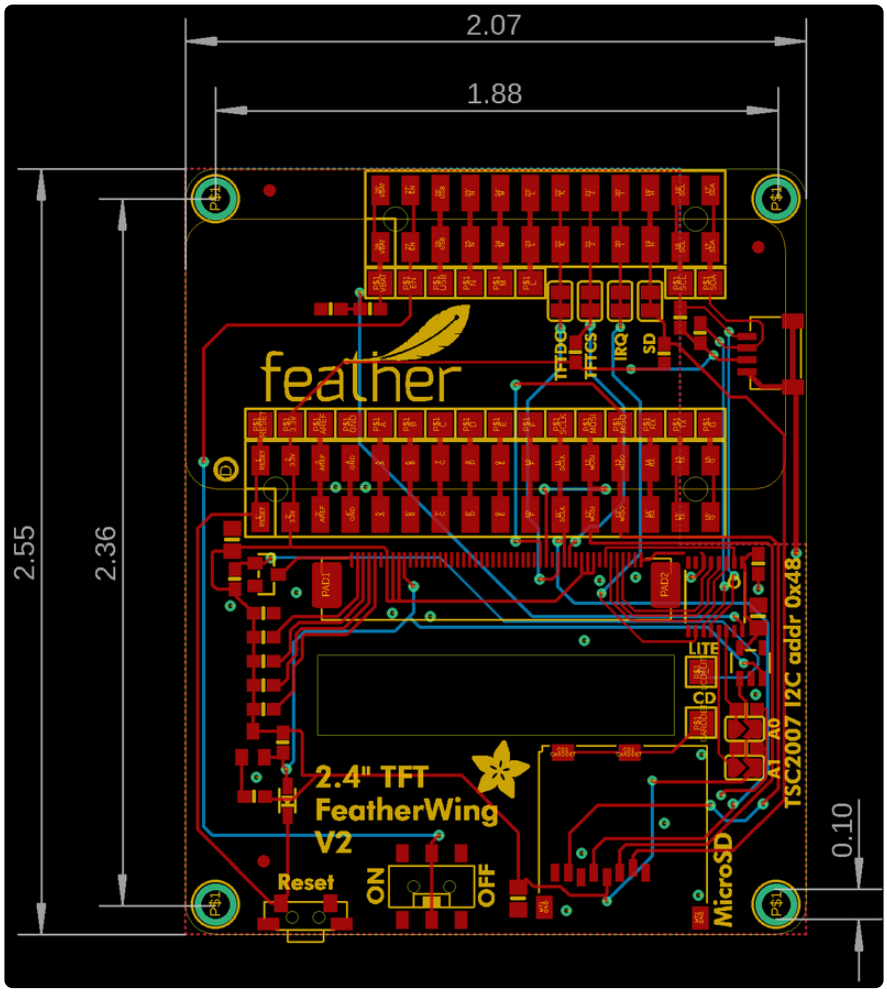
Datasheets & More

- V1 Fritzing object in Adafruit Fritzing Library (https://adafru.it/19Di)
- V2 Fritzing object in Adafruit Fritzing Library (https://adafru.it/19Dk)
- STMPE610 Touch Controller Datasheet (https://adafru.it/d4k)
- TSC2007 Touch Controller Datasheet (https://adafru.it/Zic)
- ILI9341 (TFT controller) Datasheet (https://adafru.it/d4l)
- Datasheet for TFT module itself (https://adafru.it/zAj)
- EagleCAD PCB files on GitHub (https://adafru.it/ska)
- 3D Models on GitHub (https://adafru.it/Z1A)

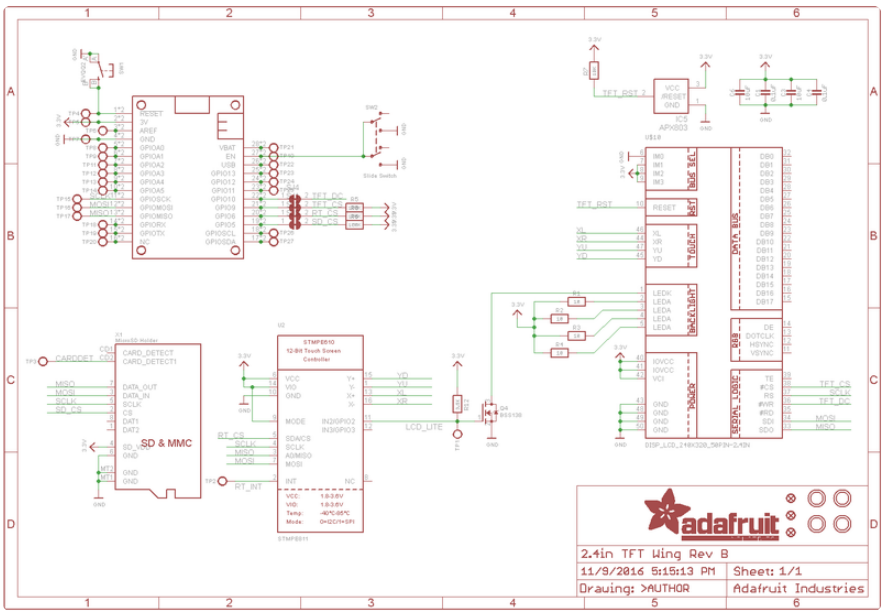
Schematic and Fab Print

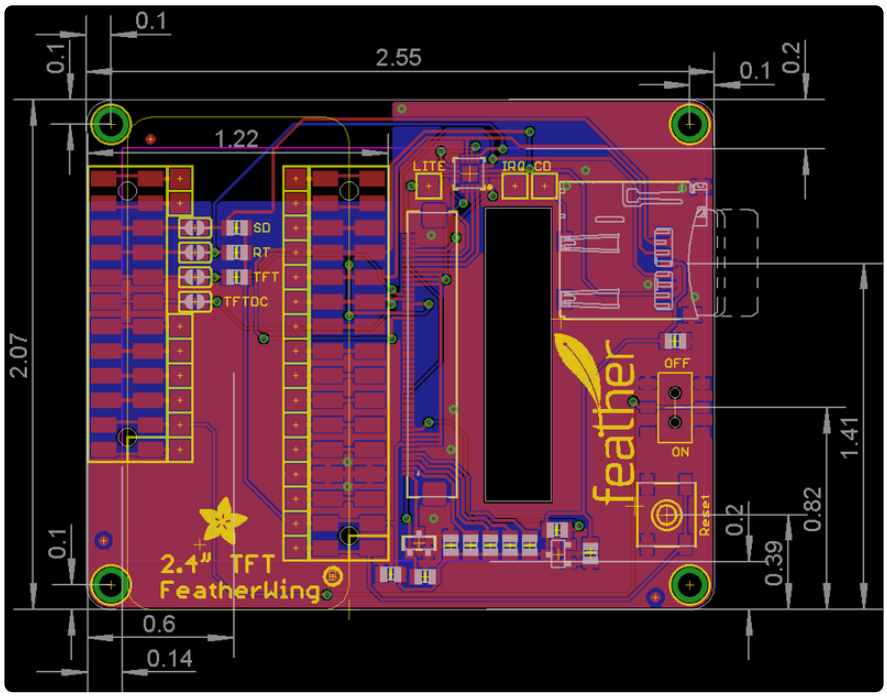
V2





V1


adafruit
2.4in TFT Wing Rev B
11/9/2016 5:15:13 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries
SD & MMC


0.1
2.55
0.2
1.22
2.07
1.41
0.82
0.39
0.6
0.14
SD
RT
TFT
TFTDC
LITE
IRQ
CD
OFF
ON
Reset
feather
2.4" TFT
FeatherWing