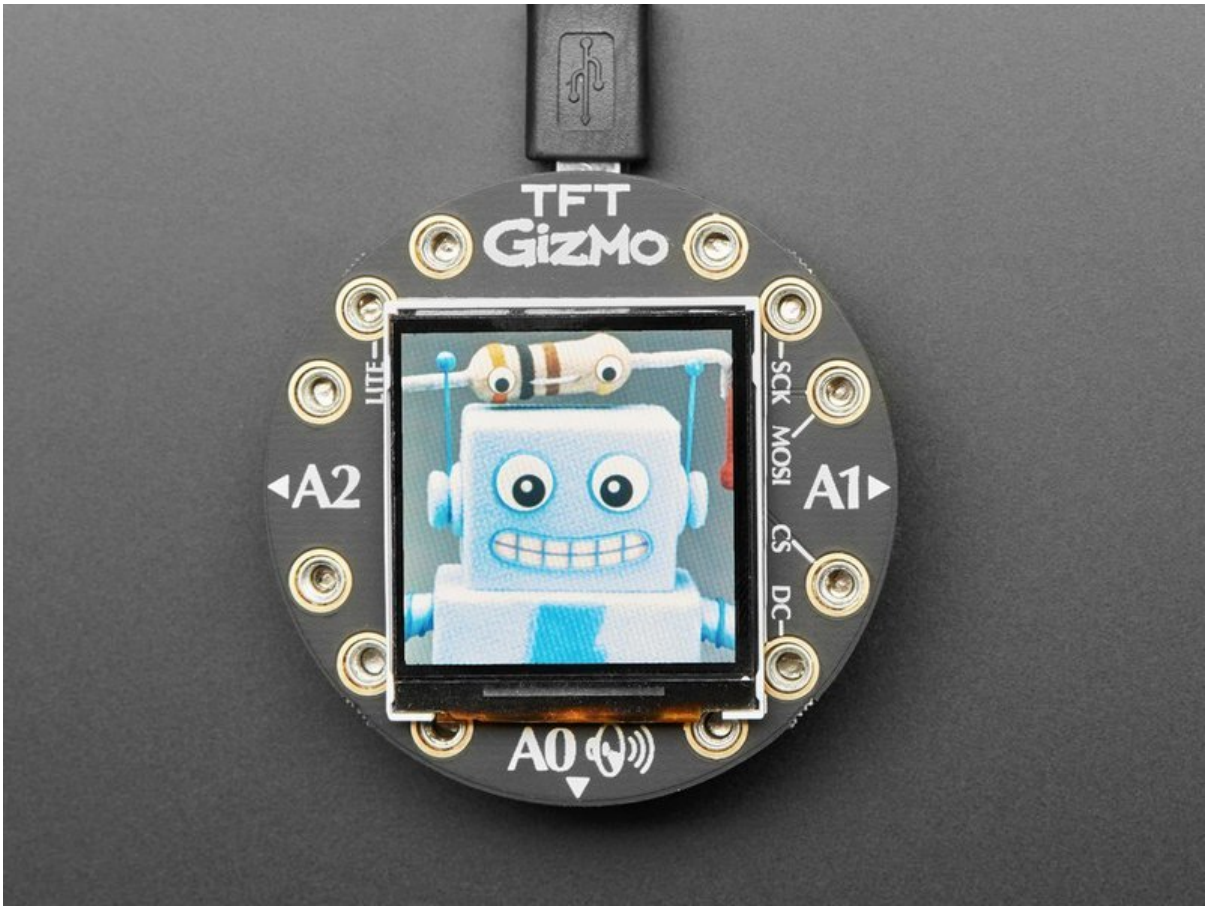
# Adafruit Circuit Playground TFT Gizmo

Created by Melissa LeBlanc-Williams

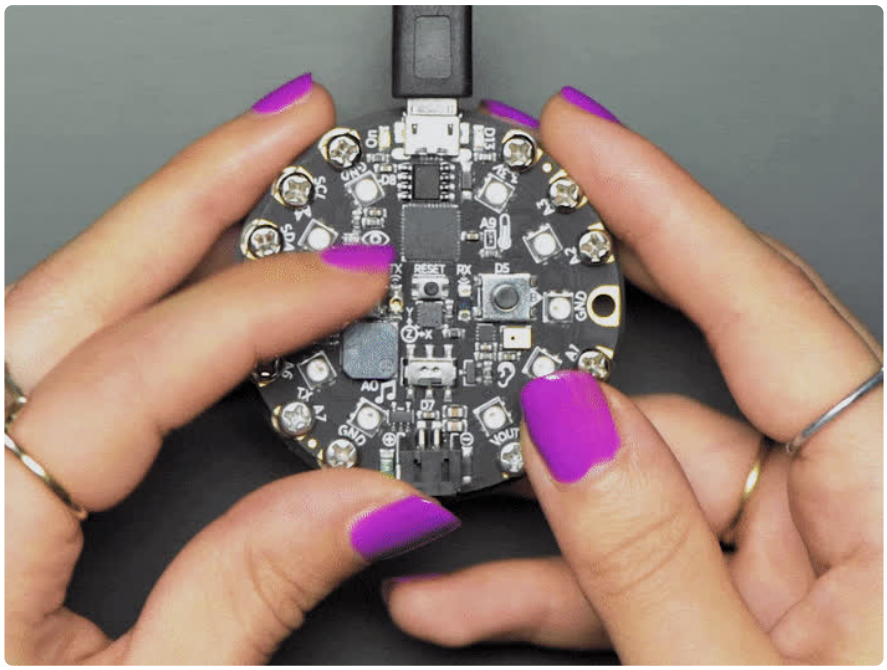



https://learn.adafruit.com/adafruit-tft-gizmo

Last updated on 2024-06-03 02:55:46 PM EDT

# Table of Contents

# Overview

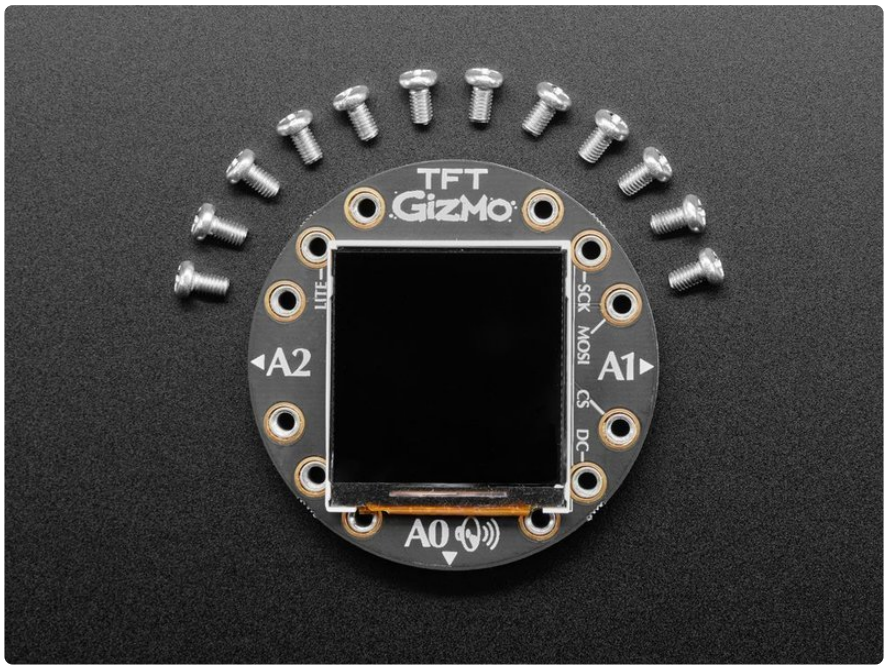


Extend and expand your Circuit Playground projects with a bolt on TFT Gizmo that lets you add a lovely color display in a sturdy and reliable fashion. This PCB looks just like a round TFT breakout but has permanently affixed M3 standoffs that act as mechanical and electrical connections.

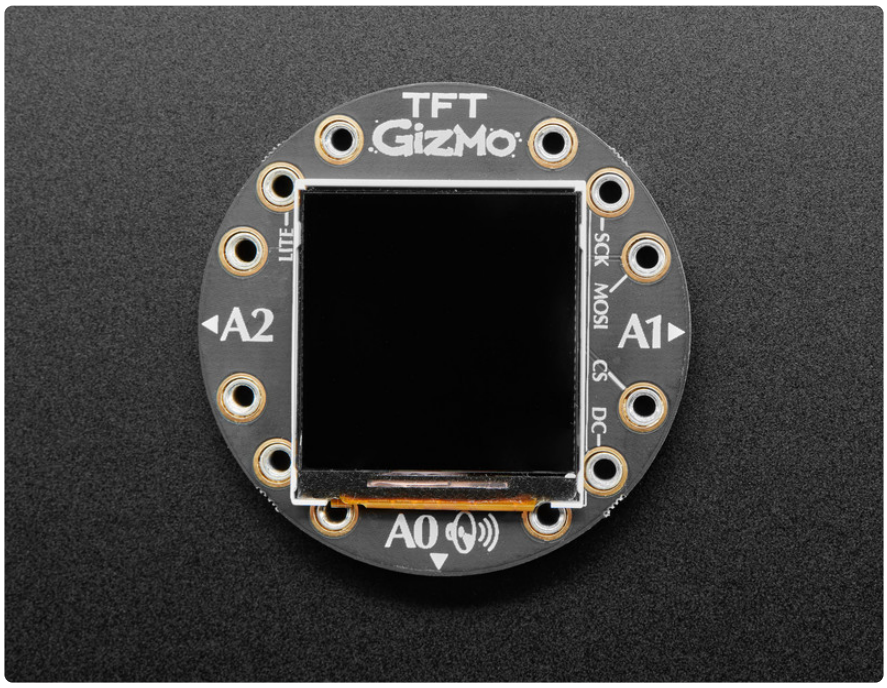
Once attached you'll get a 1.54" 240x240 IPS display with backlight control, two 3-pin STEMMA connectors for attaching NeoPixel strips (http://adafru.it/3919) or servos (http://adafru.it/4326), and a Class D audio amplifier with a Molex PicoBlade connector that can plug one of our lil speakers (https://adafru.it/FWr).

This is a great companion for our Circuit Playground Express or Bluefruit boards thanks to their fast SPI hardware speeds, and works in Arduino and CircuitPython. You can use it with the Circuit Playground Classic but it won't be very fast, as you have to bitbang the SPI - and the display has a lot of pixels - so it's not recommended.

Comes with a PCB that has pre-soldered standoffs attached, and 12x M3 screws for attachment. Fits all Circuit Playgrounds but like we mentioned earlier, the Express and Bluefruit are recommended.

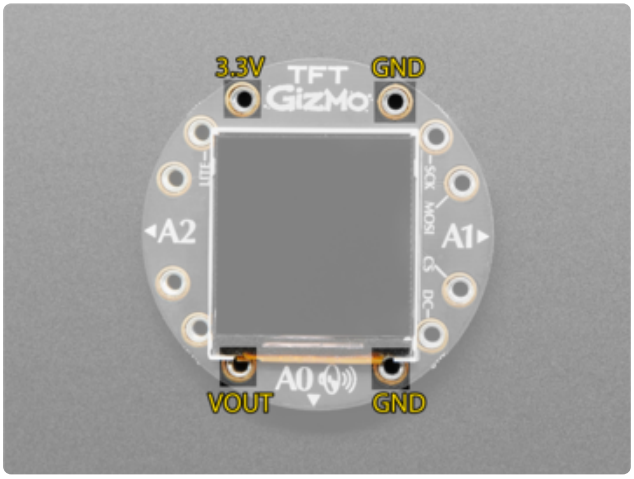



# Pinouts



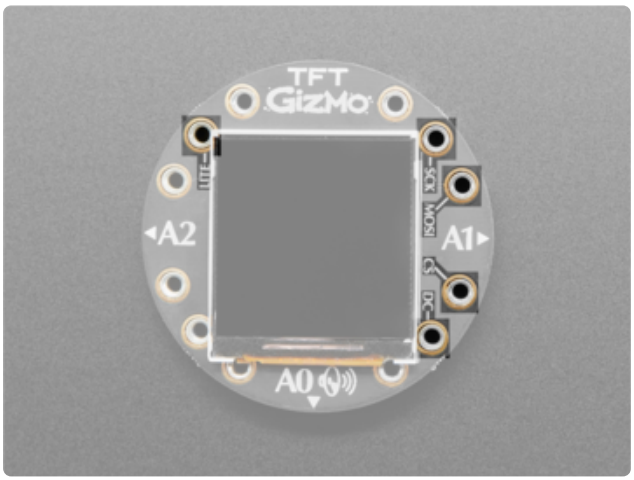
The TFT Gizmo uses SPI to receive image data. That means at least 5 pads are used - **clock**, **data in**, **tft cs** and **d/c**. An additional pin is used as a backlight control. These are already determined because of the bolt-on design. The TFT Gizmo also has a built-in audio amplifier and 2 JST connectors.

## Power Pins




**VOUT** - power input, connect to 3-5VDC.
**GND** - power and signal ground. Connect to your power supply and microcontroller ground.
**3.3V** - power input, connected to the 3.3V output of the Circuit Playground. Also, if you have a need for a clean 3.3V output, you can use this! It can provide at least 100mA output.

## SPI Display Pins

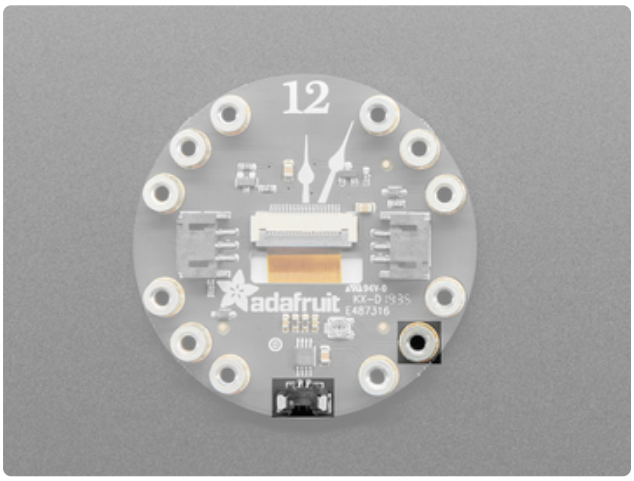



The display used on the TFT Gizmo is a 1.54" IPS TFT ST7789 display.

**SCK** - this is the SPI clock pin, its an input to the chip. This is connected to **A4** or **SCL**.
**MOSI** - this is the Microcontroller Out Serial In pin, for data sent from your processor to the TFT display. This is connected to **A5** or **SDA**.
**CS** - this is the chip select pin, drop it low to start an SPI transaction. Its an input to the chip. This is connected to **A6** or **RX**.
**DC** - this is the TFT SPI data or command selector pin. This is connected to **A7** or **TX**.
**LITE** - this is an input used to externally control the backlight. This is connected to **A3**.

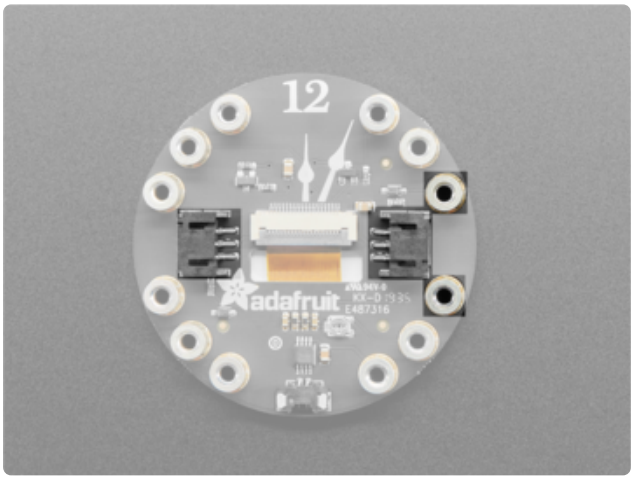
The SPI SCK and MOSI pins are not the default SPI pins for the CircuitPlayground Express or the CircuitPlayground Bluefruit.

## Audio Pins



There is a **speaker connector** with a mono 2.5 Watt Class D audio amp connected to **A0** or **AUDIO** - that's the DAC output on the SAMD21 and SAMD51. It is good for many simple sound effects or musical output.

## Extra Ports



There are two additional ports that can be used for connecting external sensors or NeoPixels.

On the **left** is a port, this is a **JST 3-PH port** and is connected to **A1**.
On the **right** is another port, this is a **JST 3-PH port** and is connected to **A2**.

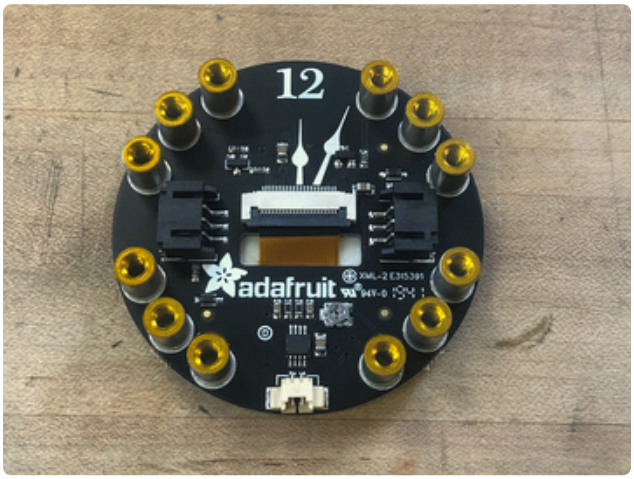
---

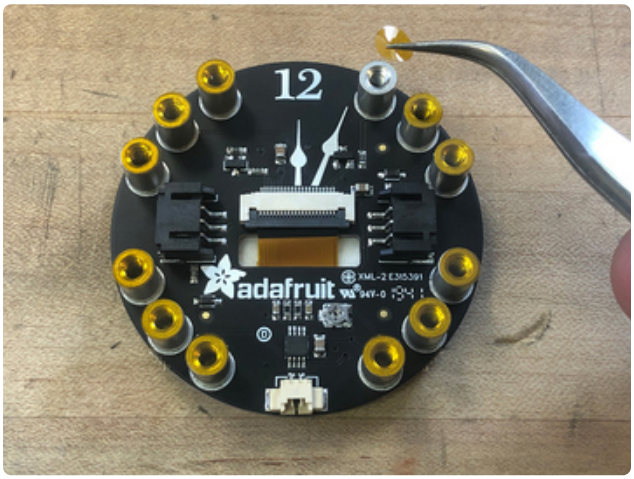
# Assembly

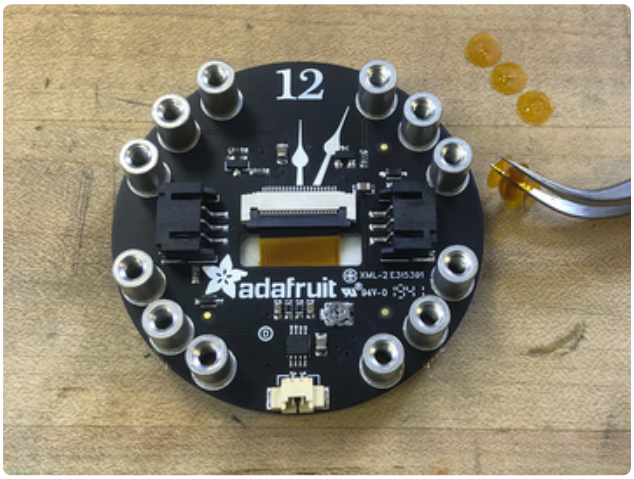
This page shows assembling the Circuit Playground TFT Gizmo, but the process is identical for the E-Ink Gizmo.

Placing the Circuit Playground TFT Gizmo on the Circuit Playground Express or Circuit Playground Bluefruit is pretty straightforward. All you need is a #2 Phillips screwdriver.

There may be plastic covers over the screw holes on the TFT Gizmo, which you will need to remove before assembly.

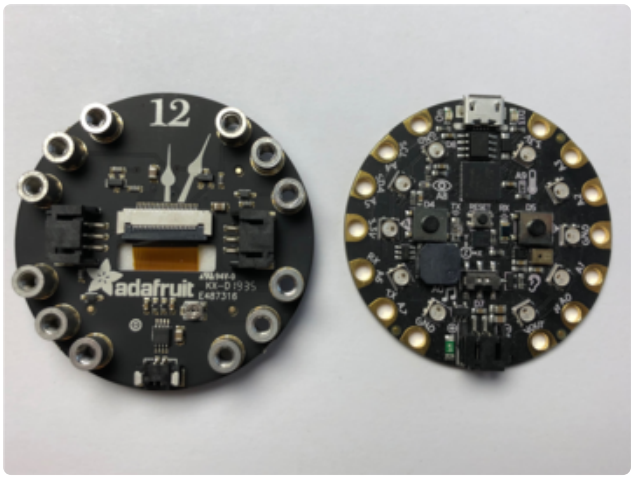


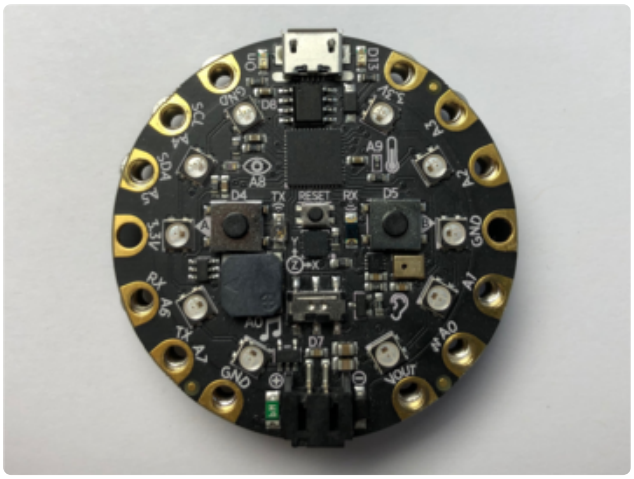



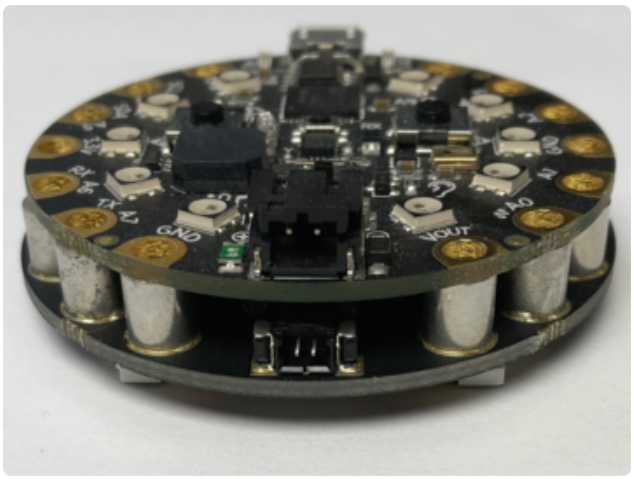

The amber colored Kapton tape dots must be removed from each of the twelve standoffs before assembling the boards. These are electrically insulating and will prevent the Gizmo from working properly if left in place.

You can use your fingernails or some tweezers or a pin to poke and lift each dot as shown here.

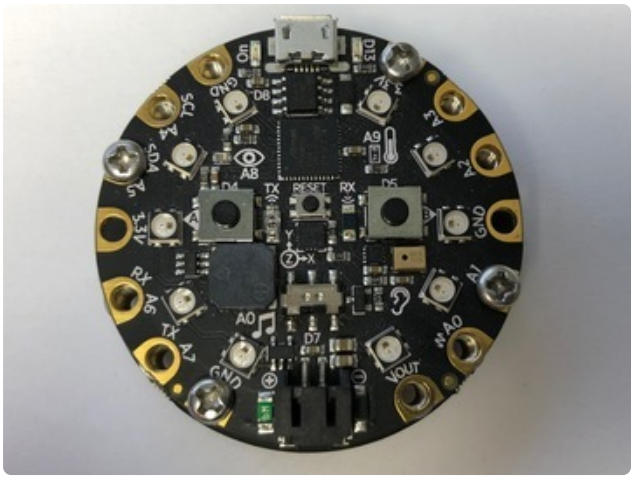



Start by aligning the two boards side by side like in the photo with the black plastic speaker connector and battery connectors pointing in the same direction.

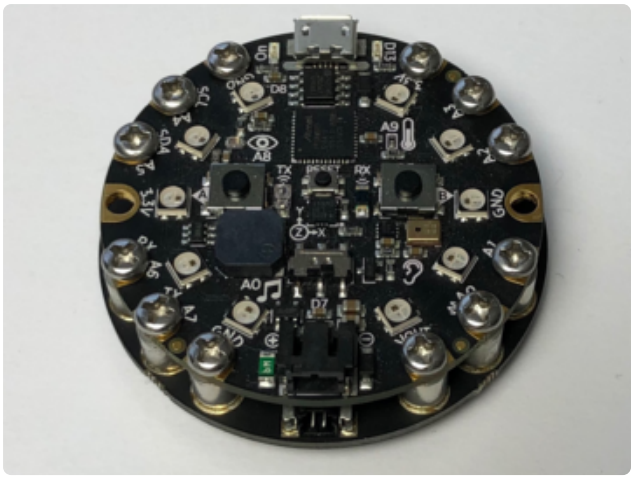



Place the Circuit Playground board on top of the Gizmo being sure that the connectors mentioned in the previous step are still aligned.

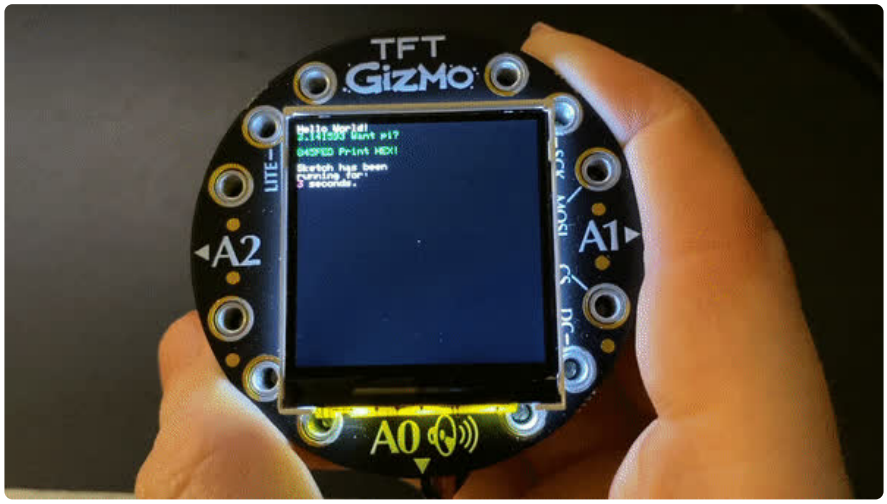


Install a few screws loosely, so that all of the holes are still aligned, before tightening them down.



Finish installing the remaining screws. After that, you're done!

## TFT Gizmo Test




Once you've assembled the TFT Gizmo onto your Circuit Playground you can test it by loading one of these pre-made UF2 onto your Circuit Playground with the TFT Gizmo attached - it will perform a graphics test so you know that the Gizmo's attached!

Be sure to use the correct UF2 file for the Circuit Playground board you have.

## Circuit Playground Express

If you have a Circuit Playground Express (http://adafru.it/3333), download this UF2 file by clicking below:

**Circuit_Playground_Express_TFT_Giz**

https://adafru.it/IEA

## Circuit Playground Bluefruit

If you have a Circuit Playground Bluefruit (http://adafru.it/4333), download this UF2 file by clicking below:

**Circuit_Playground_Bluefruit___TFT_**

https://adafru.it/HDt

## Running Text Example UF2

Once you've downloaded the appropriate UF2 file from above, plug in your Circuit Playground to your computer with a data/sync cable.

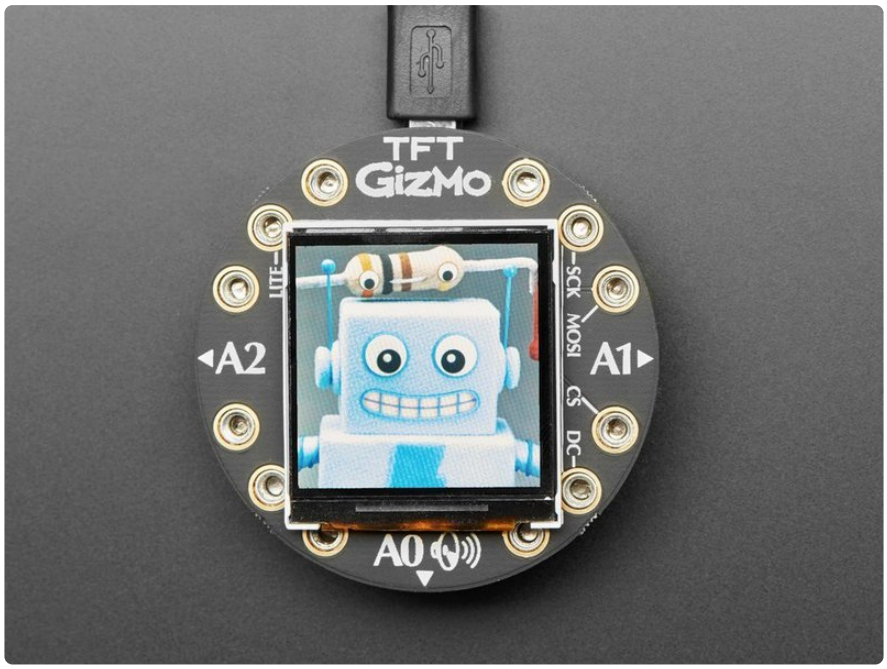
Double-click the reset button so that the boot drive folder shows up. This folder should be called **CPLAYBOOT** for the CP Express and **CPLAYBTBOOT** for the CP Bluefruit.

Drag-n-drop the downloaded UF2 onto **CPLAYBOOT** or **CPLAYBTBOOT** folder.

Press reset button to launch the code. If you don't get graphics on the TFT, check the assembly instructions on this page (https://adafru.it/Hb9)

This sketch is in Arduino, so after you've run this UF2, if you want to go back to CircuitPython, you'll need to re-install the CircuitPython UF2 (https://adafru.it/Em8)

# Arduino Libraries & Test

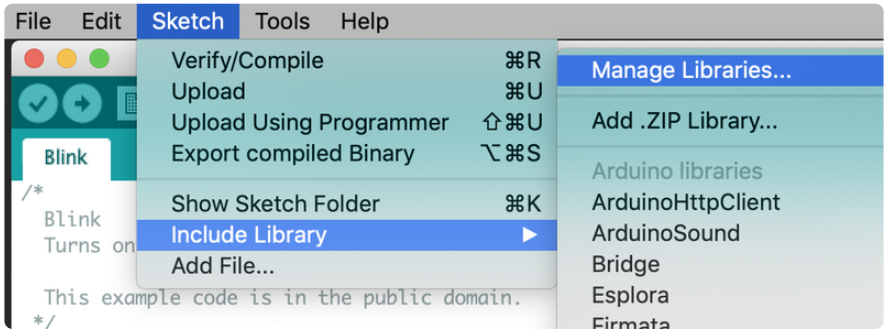



The CircuitPlayground Classic will only be able to run the Graphics Test in Software SPI mode.

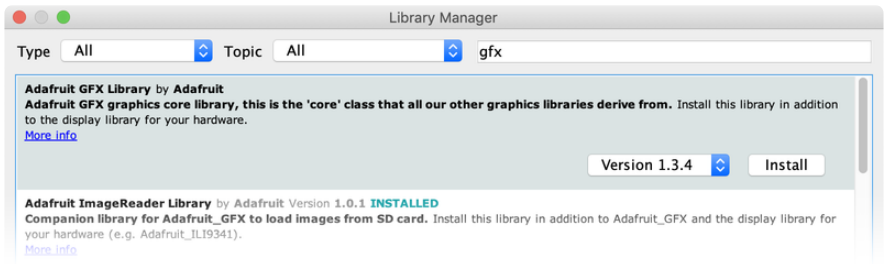
## Install Arduino Libraries

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

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

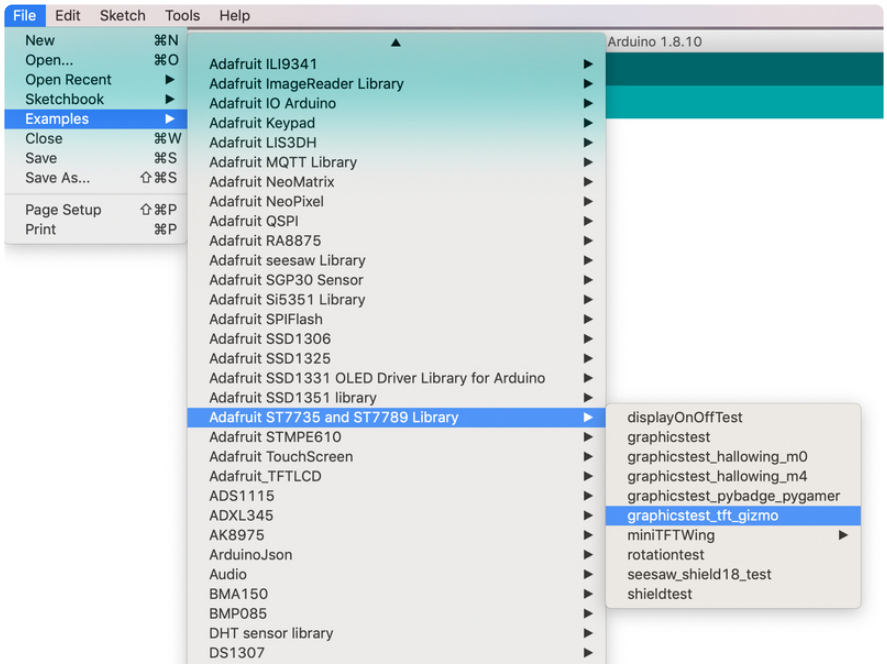



Type “gfx” in the search field to quickly find the first library — **Adafruit_GFX**:

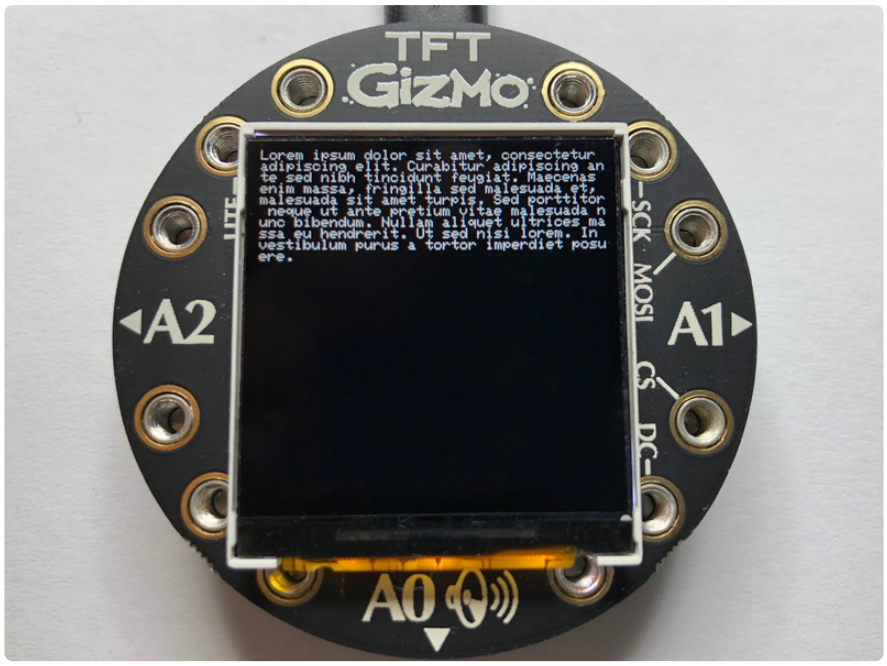



Repeat the search and install steps, looking for the **Adafruit BusIO**, **Adafruit Zero DMA**, **Adafruit ST7735 and ST7789**, and **Adafruit SPIFlash** libraries.

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




Now upload the sketch to your Circuit Playground board. You may need to press the Reset button to reset the Circuit Playground and TFT. You should see a collection of graphical tests draw out on the TFT.



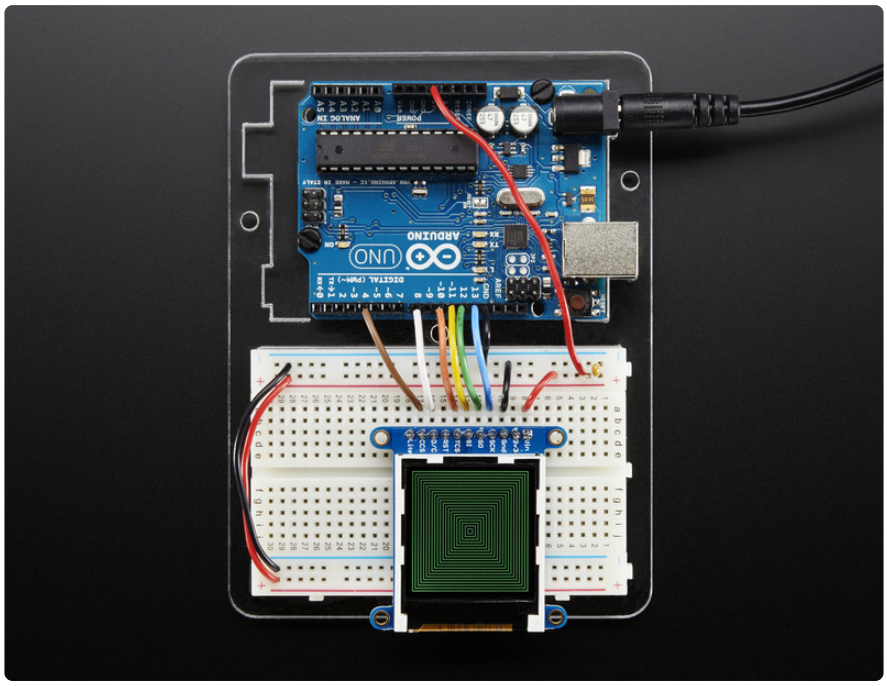

## Running in Software SPI mode

If you have a CircuitPlayground Classic or just want to run it as Software SPI for whatever reason, it's pretty easy to change. Due to the nature of the circular bolt-on style board that the gizmo is, physically changing pins isn't so easy, but we can change to software SPI, which is a bit slower. Find these lines:

```
// OPTION 1 (recommended) is to use the HARDWARE SPI pins, which are unique
// to each board and not reassignable.
Adafruit_ST7789 tft = Adafruit_ST7789(spi, 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      PIN_WIRE_SDA  // Data out
//#define TFT_SCLK      PIN_WIRE_SCL  // Clock out
//Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
TFT_RST);
```

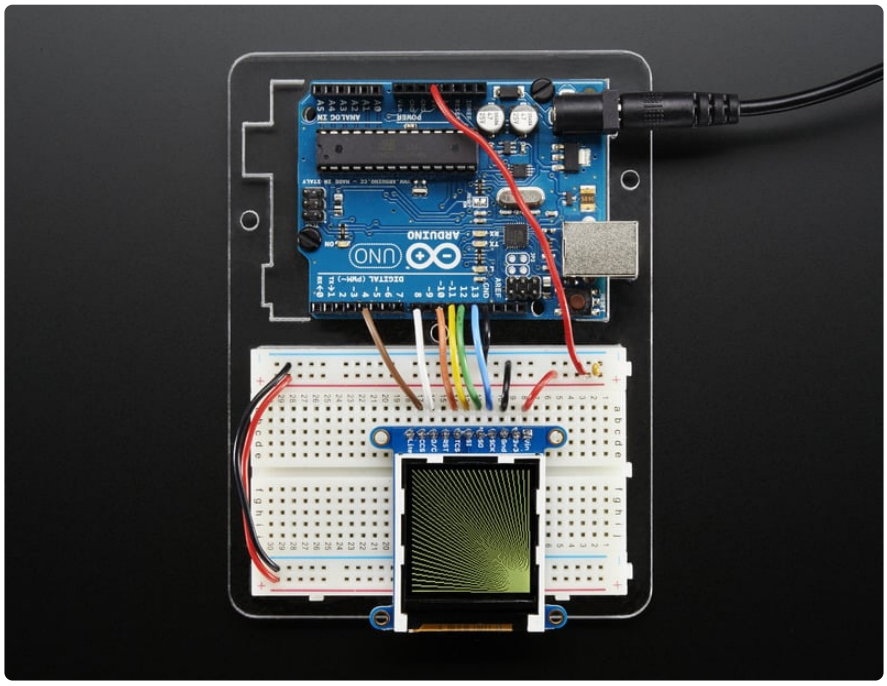
Comment out option 1, and uncomment option 2 Then you can change the **TFT_** pins to whatever pins you'd like!

# 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!

---

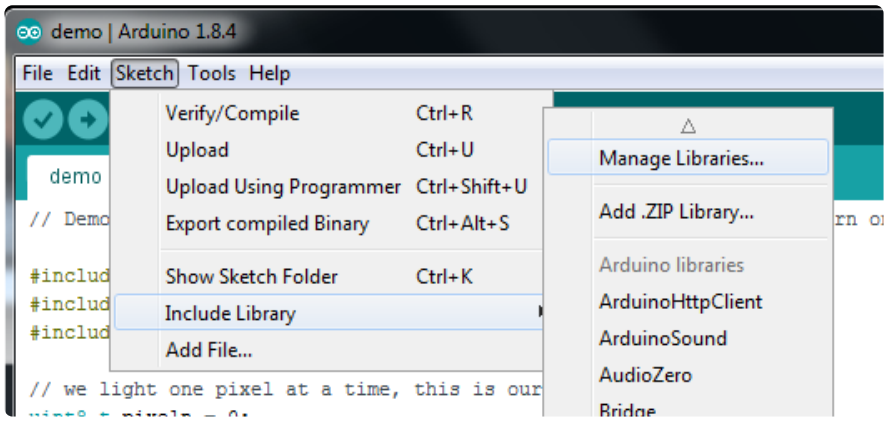
# Arcada Libraries

OK now that you have Arduino IDE set up, drivers installed if necessary and you've practiced uploading code, you can start installing all the Libraries we'll be using to program it.

**There's a lot of libraries!**

# Install Libraries

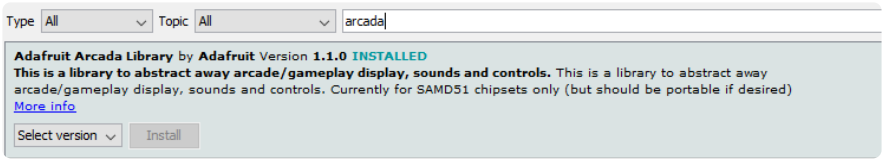
Open up the library manager...




And install the following libraries:

## Adafruit Arcada

This library generalizes the hardware for you so you can read the joystick, draw to the display, read files, etc. without having to worry about the underlying methods

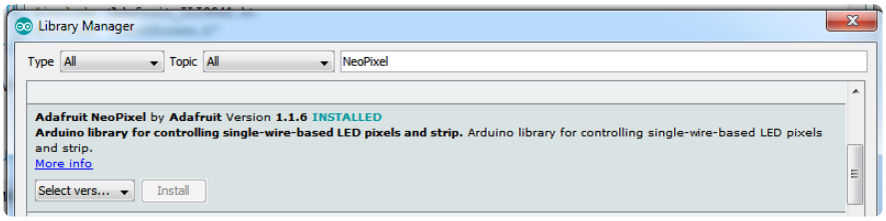



If you use Arduino 1.8.10 or later, the IDE will automagically install all the libraries you need to run all the Arcada demos when you install Arcada. We strongly recommend using the latest IDE so you don't miss one of the libraries!

If you aren't running Arduino IDE 1.8.10 or later, you'll need to install all of the following!

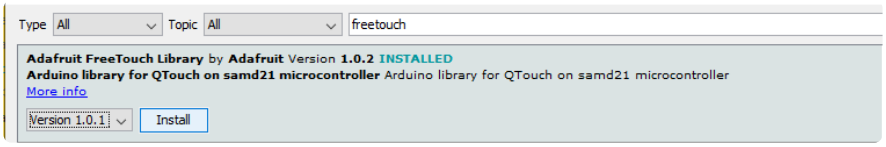
## Adafruit NeoPixel

This will let you light up the status LEDs on the front/back



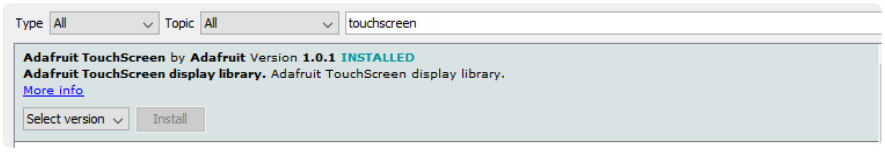

# Adafruit FreeTouch

This is the open source version of QTouch for SAMD21 boards




# Adafruit Touchscreen

Used by Adafruit Arcada for touchscreen input (required even if your Arcada board does not have a touchscreen)

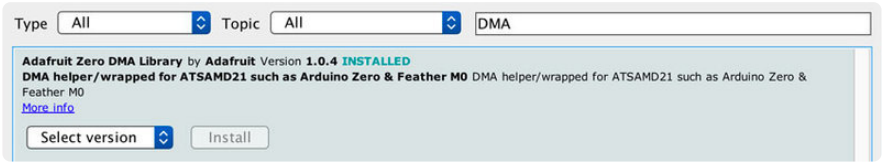



## Adafruit SPIFlash

This will let you read/write to the onboard FLASH memory with super-fast QSPI support

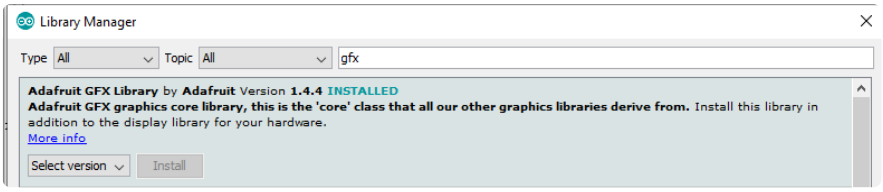

## Adafruit Zero DMA

This is used by the Graphics Library if you choose to use DMA




## Adafruit GFX

This is the graphics library used to draw to the screen




If using an older (pre-1.8.10) Arduino IDE, locate and install **Adafruit_BusIO** (newer versions do this one automatically).

## Adafruit ST7735

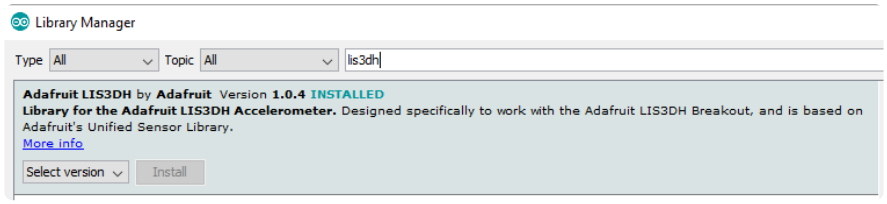
The display on the PyBadge/PyGamer & other Arcada boards


## Adafruit ILI9341

The display on the PyPortal & other Arcada boards

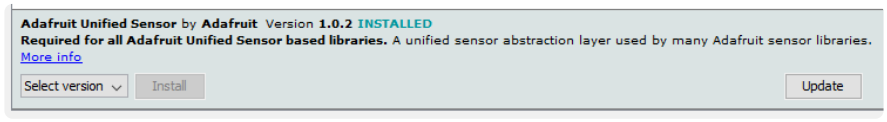

## Adafruit LIS3DH

For reading the accelerometer data, required even if one is not on the board



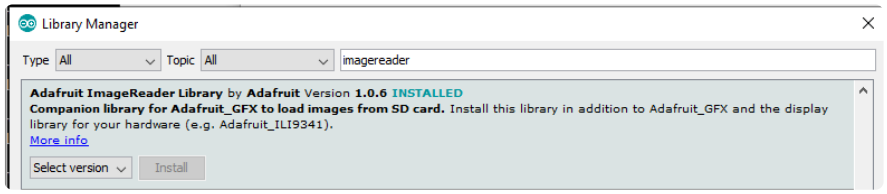

## Adafruit Sensor

Needed by the LIS3DH Library, required even if one is not on the board



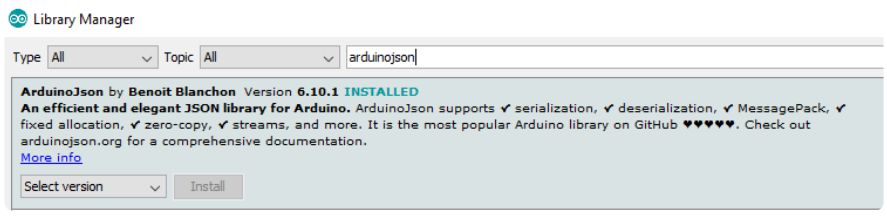

# Adafruit ImageReader

For reading bitmaps from SPI Flash or SD and displaying



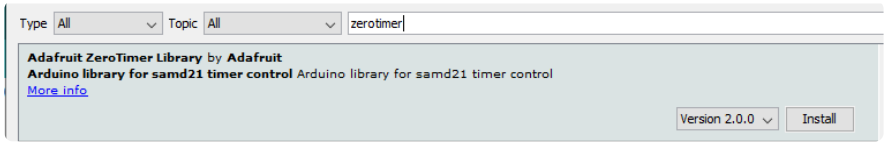

# ArduinoJson

We use this library to read and write configuration files



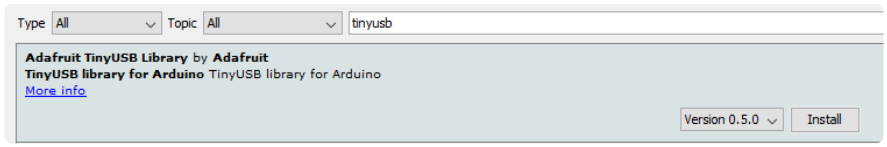
# Adafruit ZeroTimer

We use this library to easily set timers and callbacks on the SAMD processors



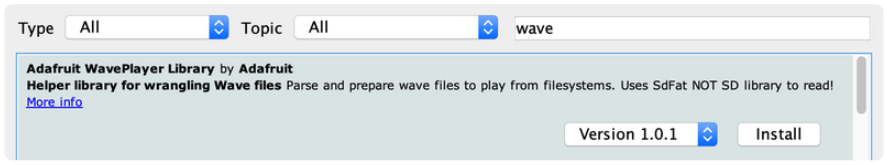

# Adafruit TinyUSB

This lets us do cool stuff with USB like show up as a Keyboard or Disk Drive



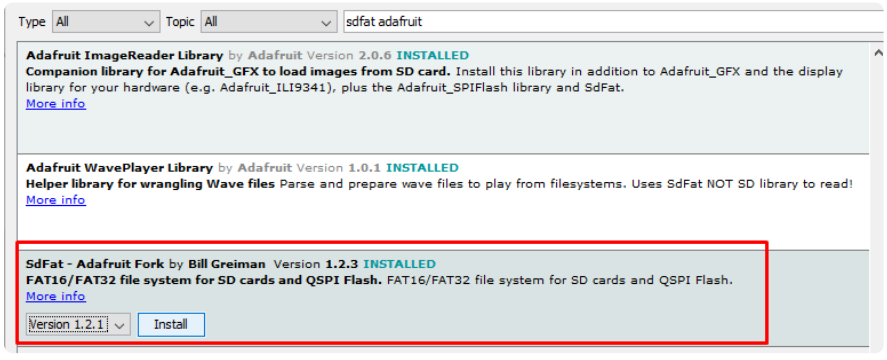

# Adafruit WavePlayer

Helps us play .WAV sound files.




## SdFat (Adafruit Fork)

The Adafruit fork of the really excellent SD card library that gives a lot more capability than the default SD library



# Audio - Adafruit Fork

Our fork of the Audio library provides a toolkit for building streaming audio projects.


---

# Drawing Bitmaps

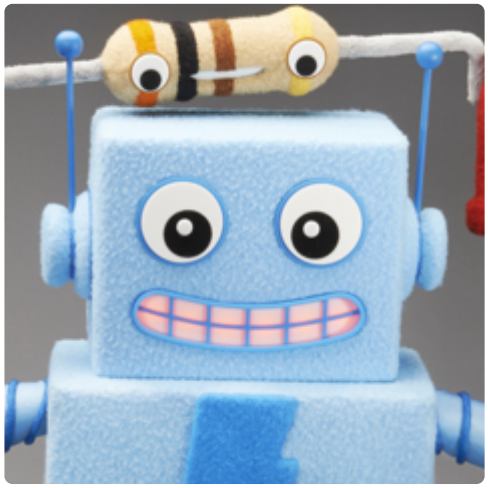
This will not work for the Circuit Playground Classic because it does not have SPI flash.

Although there isn't a built in microSD card slot, we can still use the SPI flash to store and load bitmap images! It's really easy with Adafruit Arcada because it can make the SPI flash show up as a drive on your computer that you can easily copy files over to.

Let's start by downloading this image of Adabot and Blinka.

If you would like to use your own images, they should be a 240x240 sized image in 24-bit Bitmap format.

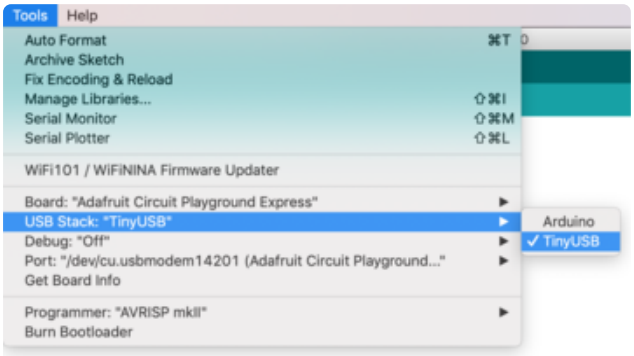
The images are also available in the images folder of the Arcada Library.



## Required Libraries for Bitmaps

To draw Bitmaps, we'll be using the Arcada library, but there are a few other required libraries as well. If you haven't already, go ahead and take a look at the Arcada Libraries page to get all set up.

## TinyUSB

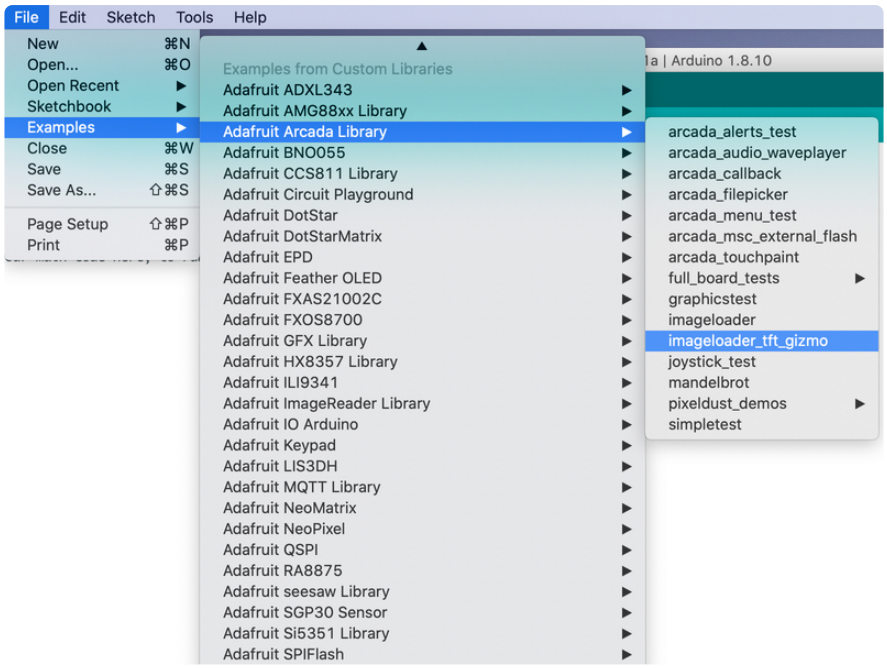



Now that you have the required libraries installed, there is one last thing to do if you are using the Circuit Playground Express. You will need to change your USB Stack over to TinyUSB.

## Arcada Library

If you get any messages about missing libraries, make sure you installed all of the required Arcada libraries.

Start by opening the **File→examples→Adafruit Arcada Library→imageloader_tft_gizmo** example:




## Uploading the Images

Now upload the example sketch to the Circuit Playground.

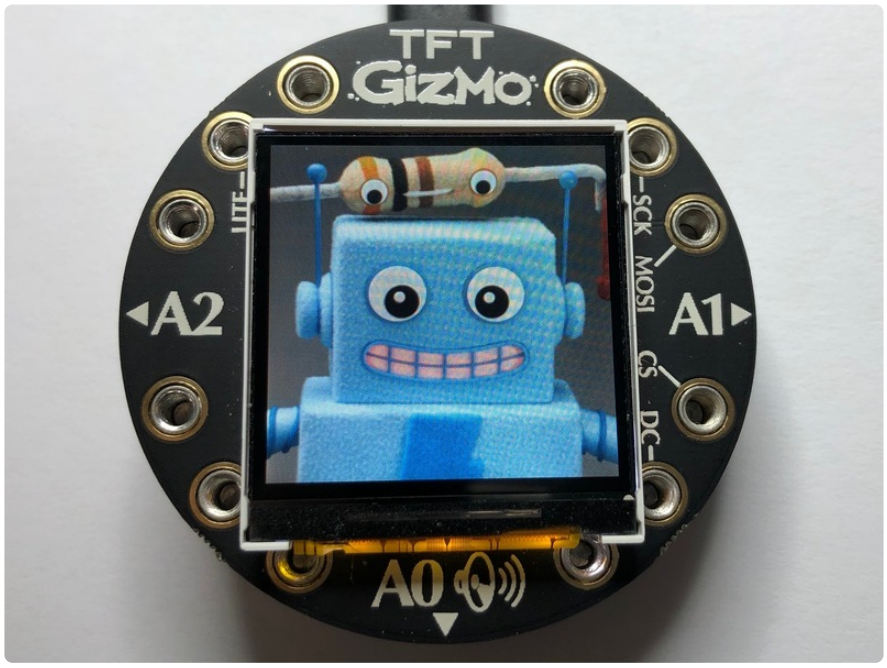

If you have not previously initialized the filesystem, you may receive a messages stating so. Go ahead and load CircuitPython on the device to initialize the files. If you are not sure how, you can check out our Welcome to CircuitPython guide (https://adafru.it/Amd).

After it is initialized, go ahead and upload the sketch again.


After it finishes uploading, and with the Circuit Playground board connected over USB, it should appear on your computer as a flash drive called CIRCUITPY.

Now go ahead and copy **adabot.bmp** and **blinka.bmp** onto the root of the **CIRCUITPY** drive.

Press the buttons on the back of your Circuit Playground board. Pressing one should make ADABOT appear! Pressing the other will make BLINKA appear. If you have any problems, check the serial console for any messages such as not being able to initialize the flash or not finding the image.




## Full ImageLoader Example

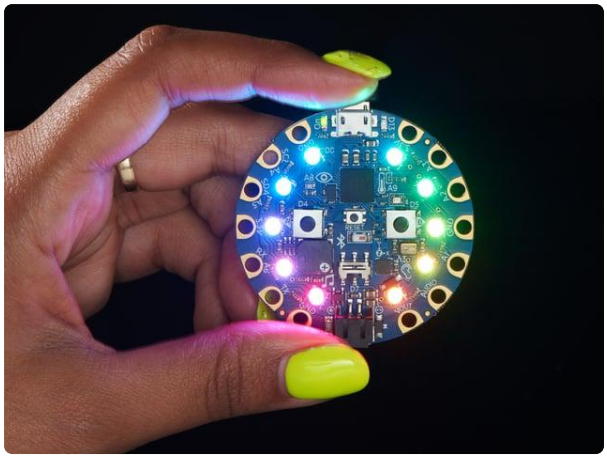
```
#include "Adafruit_Arcada.h"
Adafruit_Arcada arcada;

#define IMAGE_A "adabot.bmp"
#define IMAGE_B "blinka.bmp"

void setup(void) {
  if (!arcada.arcadaBegin()) {
    while (1);
  }
  // If we are using TinyUSB we will have the filesystem show up!
  arcada.filesysBeginMSD();

  Serial.begin(115200);
  //while(!Serial) delay(10);       // Wait for Serial Monitor before continuing

  arcada.enableSpeaker(false);

  // Start TFT and fill blue
  arcada.displayBegin();
  arcada.display->fillScreen(ARCADA_BLUE);
  arcada.setBacklight(255);

  if (arcada.filesysBegin()) {
    Serial.println("Found filesystem!");
  } else {
    arcada.haltBox("No filesystem found! For QSPI flash, load CircuitPython. For SD
```

```
cards, format with FAT");
  }

  // Turn on backlight
  arcada.setBacklight(255);
}


void loop() {
  const char *imagefile = 0;
  arcada.readButtons();
  uint8_t buttons = arcada.justPressedButtons();
  Serial.print("Pressed: ");
  if (buttons & ARCADA_BUTTONMASK_LEFT) {
    imagefile = IMAGE_A;
    Serial.print("< ");
  }
  if (buttons & ARCADA_BUTTONMASK_RIGHT) {
    imagefile = IMAGE_B;
    Serial.print("> ");
  }
  Serial.println();

  delay(25);
  if (! imagefile) return;

  for (int i=255; i>=0; i--) {
    arcada.setBacklight(i);
    delay(1);
  }

  // Load full-screen BMP file at position (0,0) (top left).
  Serial.printf("Loading %s to screen...", imagefile);
  ImageReturnCode stat = arcada.drawBMP((char*) imagefile, 0, 0);
  if(stat == IMAGE_ERR_FILE_NOT_FOUND) {
    arcada.haltBox("File not found");
  } else if(stat == IMAGE_ERR_FORMAT) {
    arcada.haltBox("Not a supported BMP variant.");
  } else if(stat == IMAGE_ERR_MALLOC) {
    arcada.haltBox("Malloc failed (insufficient RAM).");
  }
  for (int i=0; i<=255; i++) {
    arcada.setBacklight(i);
    delay(1);
  }
}
```

# CircuitPython Displayio Quickstart

You will need a board capable of running CircuitPython such as the Circuit Playground Express or Circuit Playground Bluefruit. The Circuit Playground Classic will only run Arduino sketches.

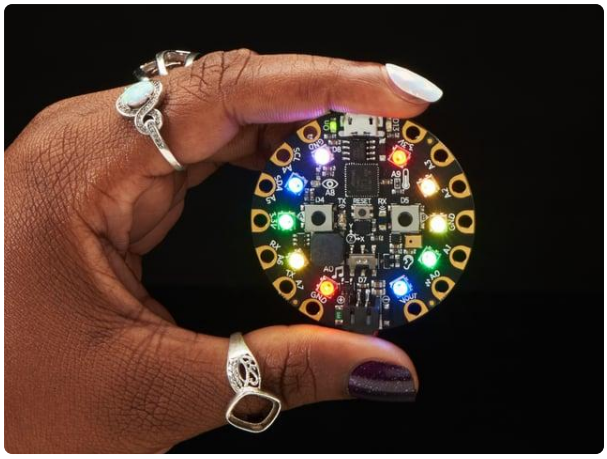
The Circuit Playground Express, which has a SAMD21 chip, is not able to run the full example due to memory constraints. We recommend using the Circuit Playground Bluefruit.



Circuit Playground Bluefruit - Bluetooth Low Energy

Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...

https://www.adafruit.com/product/4333



Circuit Playground Express

Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...

https://www.adafruit.com/product/3333

## Circuit Playground Express with Displayio

If you have a Circuit Playground Express board, you will need a special build that includes displayio to use the TFT Gizmo. Be sure to download the latest one. You can find it here:

**CircuitPython for Circuit Playground Express with displayio**

https://adafru.it/G0A

## Required CircuitPython Libraries

To use this display with `displayio`, there are only two required libraries.

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 the library 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.

Due to the number of libraries in the bundle, it is recommended that you manually install the necessary library from the bundle:

- **adafruit_st7789**
- **adafruit_gizmo**

Before continuing make sure your board's **lib** folder has the **adafruit_st7789.mpy** and **adafruit_gizmo** file and folder 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. You'll also need to copy over the following library from the bundle:

- **adafruit_display_text**

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.
"""
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_gizmo import tft_gizmo

# Create the TFT Gizmo display
display = tft_gizmo.TFT_Gizmo()

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

color_bitmap = displayio.Bitmap(240, 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(200, 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=2, x=50, 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
```

Let's take a look at the sections of code one by one. We start by importing `displayio`, `terminalio` for the font, a `label`, and the `tft_gizmo` helper.

```
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_gizmo import tft_gizmo
```

Next, we initialize the helper, which takes care of all the TFT Gizmo ST7789 Driver initialization for us. 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 = tft_gizmo.TFT_Gizmo()
```

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.

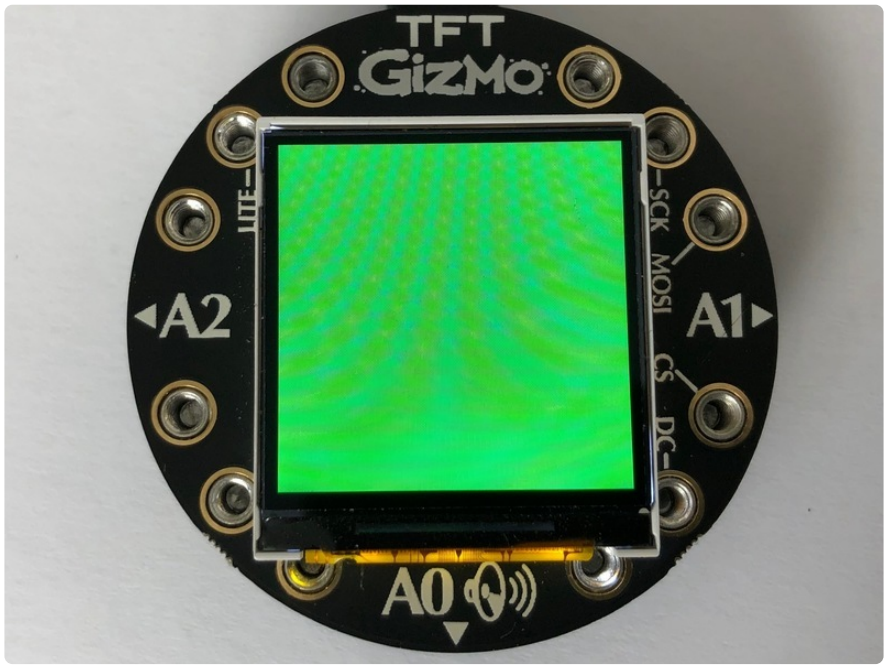
```
splash = displayio.Group(max_size=10)
display.root_group = splash
```

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(240, 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)
```



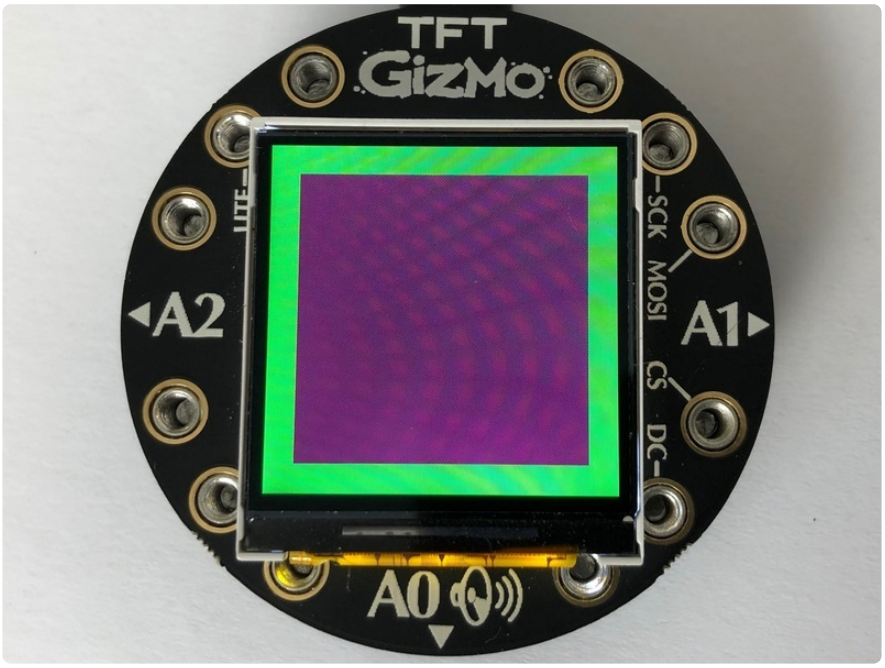

Next we will create a smaller purple square. The easiest way to do this is the create a new bitmap that is a little smaller than the full screen with a single color and place it in

a specific location. In this case, we will create a bitmap that is 20 pixels smaller on each side. The screen is **240x240**, 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(200, 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 square, it's automatically drawn on top. Here's what it looks like now.



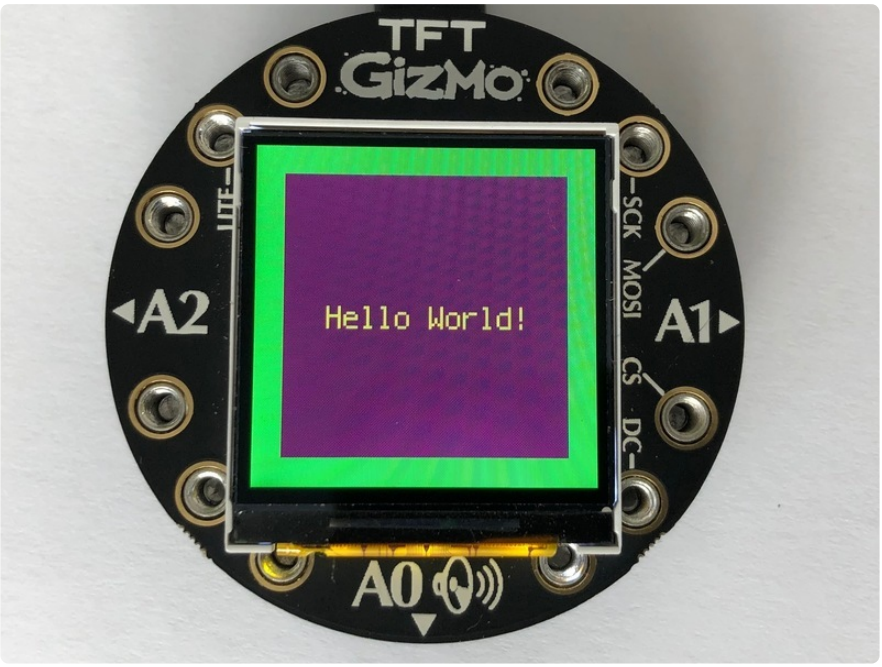

Next let's add a label that says "Hello World!" on top of that. We're going to use the built-in Terminal Font and scale it up by a factor of two. To scale the label only, we will make use of a subgroup, which we will then add to the main group.

Labels are centered vertically, so we'll place it at 120 for the Y coordinate, and around 50 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(max_size=10, scale=2, x=50, 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
```




## Where to go from here

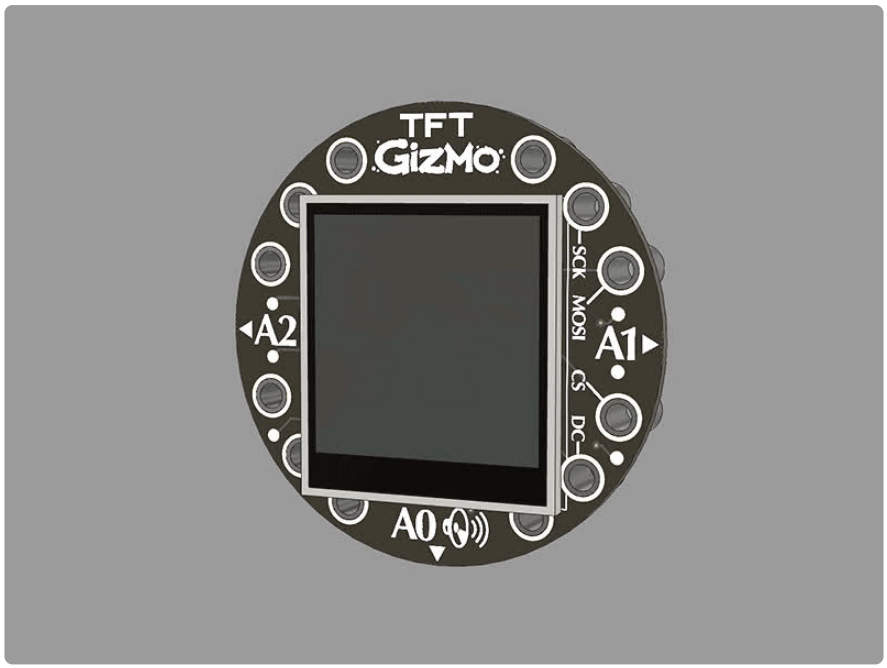
Be sure to check out this excellent guide to CircuitPython Display Support Using displayio (https://adafru.it/EGh)

---

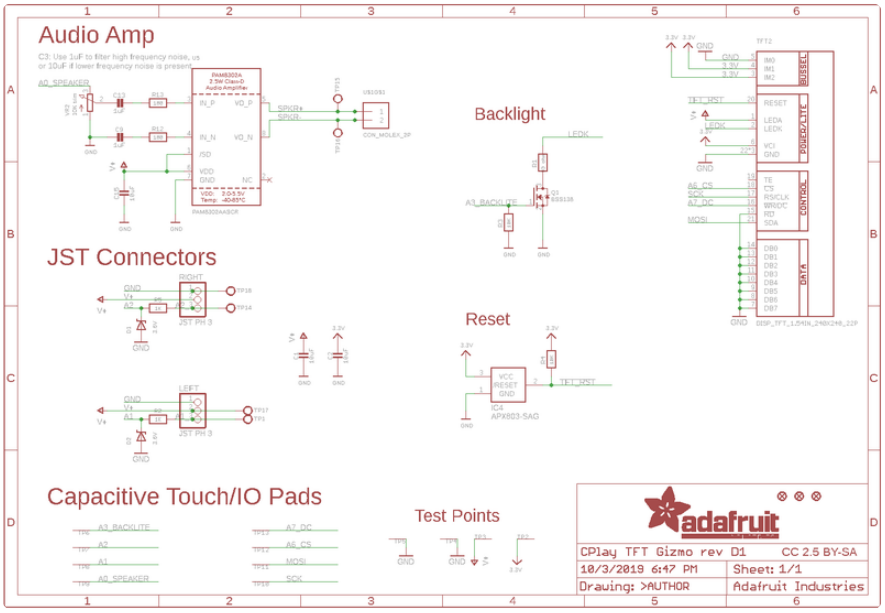
# Downloads

## Files

- EagleCAD files for Circuit Playground TFT Gizmo on GitHub (https://adafru.it/FYA)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/FYB)
- 3D Model on Github (https://adafru.it/Gcl)




## Schematic for Circuit Playground TFT Gizmo



# Fab Print