CircuitPython Hardware: ILI9341 TFT & FeatherWing
Created by Tony DiCola

Last updated on 2019-09-18 12:36:13 AM UTC
Overview

Small TFT displays are a great way to add graphics to your projects. These are like tiny little LCD monitors that you can drive with a simple SPI serial interface. You can even use these displays in CircuitPython and MicroPython using a module from Adafruit! This module allows you to do basic drawing like putting pixels and filling rectangles on TFT displays like the TFT FeatherWing (https://adafruit.it/szd). You can start to explore a fun world of Python and graphical TFT displays!

This guide explores how to use ILI9341/ILI9340 TFT displays with CircuitPython and MicroPython. Note that right now drawing support for these displays is limited to basic pixel and rectangle drawing commands. You can use another library to draw basic graphics (https://adafruit.it/zdy) or to draw text (https://adafruit.it/sA0). In addition the touchscreens commonly found on these small TFT displays are not currently supported by the Python module. In the future a touchscreen module might be available, but for now these displays are only for viewing graphics.

Note the video above was made showing the MicroPython version of this library. Follow the guide to see both CircuitPython and MicroPython versions of the ILI9341 library.
You'll need the following parts to follow this guide:

**CircuitPython board.** This guide focuses on the ESP8266 (https://adafruit.it/n6A) and Feather M0/SAMD21-based boards (http://adafruit.it/2772), but any CircuitPython board that supports SPI should work.

If your board doesn't come with CircuitPython running on it already then check out your board's guide for how to load CircuitPython firmware. For example the Feather M0 express guide (https://adafruit.it/wbv) is a good reference.

**ILI9341/9340 TFT Display Breakout or FeatherWing.** If you're using a Feather the TFT FeatherWing (https://adafruit.it/szd) is the perfect option that easily connects to the Feather. For other boards you'll need a ILI9341 or ILI9340 display breakout, like this large 2.8" TFT display breakout (https://adafruit.it/sze). ILI9340 displays like the 2.2" TFT breakout (https://adafruit.it/szf) or 2.4" TFT breakout (http://adafruit.it/2478) should work too. Make sure the display you're using has the ILI9341 or ILI9340 driver chip!
Breadboard (http://adafru.it/64) and jumper wires (http://adafru.it/153). If you aren't using a Feather and FeatherWing you'll need a breadboard and jumper wires to connect the components.

Soldering tools (http://adafru.it/136). You'll need to solder headers to the boards. Check out the guide to excellent soldering (https://adafru.it/dxy) if you're new to soldering.

Make sure to follow the board and TFT FeatherWing (https://adafru.it/szA) or 2.8" TFT display breakout guide (https://adafru.it/szB) to assemble and test the hardware before continuing!

Wiring

If you're using a TFT FeatherWing and Feather just slide the wing onto the Feather board and you're all set! The FeatherWing will automatically be connected to the board using its SPI connection. Skip to the next page to learn about the software to control the display.
If you're using a TFT display breakout you'll need to connect its power, ground, and SPI connections to the board. For example the wiring for a 2.8" TFT breakout to Feather HUZZAH ESP8266 might look like:

- Board SCK / SPI clock to Display CLK / SPI clock.
- Board MO / MOSI / data output to Display MOSI / data input.
- Board GPIO 0 (or any other GPIO pin) to Display CS / chip select.
- Board GPIO 15 (or any other GPIO pin) to Display DC / data/command.
- Board 3.3V power to Display VIN / voltage input.
- Board GND / ground to Display GND / ground.
- Display 3.3V output to IM3, IM2, and IM1 (but not IM0!) pins. This configures the breakout to use its SPI interface. See the breakout guide for details on soldering closed these connections to make the SPI interface the default (https://adafruit.it/dLU).
CircuitPython

Adafruit CircuitPython Module Install

To use the TFT display with your Adafruit CircuitPython (https://adafruit.it/tCy) board you'll need to install the Adafruit_CircuitPython_RGB_Display (https://adafruit.it/u1C) module on your board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

**Bundle Install**

For express boards that have extra flash storage, like the Feather/Metro M0 express and Circuit Playground express, you can easily install the necessary libraries with Adafruit's CircuitPython bundle (https://adafruit.it/zdx). This is an all-in-one package that includes the necessary libraries to use the ILI9341 display with CircuitPython. To install the bundle follow the steps in your board's guide, like these steps for the Feather M0 express board (https://adafruit.it/zco).

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- adafruit_rgb_display
- adafruit_bus_device
- adafruit_register

If your board supports USB mass storage, like the M0-based boards, then simply drag the files to the board's file system. **Note on boards without external SPI flash, like a Feather M0 or Trinket/Gemma M0, you might run into issues on Mac OSX with hidden files taking up too much space when drag and drop copying, see this page for a workaround (https://adafruit.it/u1d).**

If your board doesn't support USB mass storage, like the ESP8266, then use a tool like ampy to copy the file to the board (https://adafruit.it/s1f). You can use the latest version of ampy and its new directory copy command (https://adafruit.it/q2A) to easily move module directories to the board.

Furthermore, CircuitPython for M0 boards after version 0.8.1 do not have the framebuf module built in to save flash space. So, please download and install the pure Python implementation of framebuf (https://adafruit.it/vcB) and copy it to lib folder of the board as well.

Before continuing make sure your board's root filesystem has the adafruit_rgb_display, adafruit_bus_device, and adafruit_register folders/modules copied over.
Usage

The following section will show how to control the LED backpack from the board's Python prompt / REPL. You'll walk through how to control the TFT display and learn how to use the CircuitPython module built for the display. As a reference be sure to see the micropython-adafruit-rgb-display module documentation (https://adafru.it/szE) too.

First connect to the board's serial REPL (https://adafru.it/pMf) so you are at the CircuitPython >>> prompt.

SPI Initialization

On CircuitPython the SPI bus must be initialized before the display can be used by your code. Run the following code to import the necessary modules and initialize the SPI bus:

```python
import board
import busio
import digitalio

spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# For the ESP8266
cs = digitalio.DigitalInOut(board.GPIO0)
dc = digitalio.DigitalInOut(board.GPIO15)

# For the Feather M0s
#cs = digitalio.DigitalInOut(board.D9)
#dc = digitalio.DigitalInOut(board.D10)
```

These lines create the SPI bus interface and two digital inputs/outputs for the chip select and data/command lines connected to the display. Notice that the pin numbers might be different depending on your board and how it's wired. Read the comments above and pick the correct two cs and dc lines to run for your setup.
Display Initialization

Next to import the display library and initialize it run the following code:

```python
from adafruit_rgb_display import ili9341, color565
display = ili9341.ILI9341(spi, cs=cs, dc=dc)
```

When creating the display instance of the ILI9341 class you'll need to know which pins are connected to the display's CS, DC, and optionally RST or reset line. For the TFT FeatherWing see its guide for details on these pin connections (https://adafru.it/szF).

The CS and DC parameters to the ILI9341 class initializer are required and should be a pin from the board module. In CircuitPython they are DigitalInOut objects instead of pins directly (like GPIO0) so that other types of GPIO can be used such as GPIO expanders.

There are a few optional keyword arguments you can specify too:

- `rst` - This is a GPIO pin connected to the RST or reset line on the display. The default for this is to not be specified and reset is not used.
- `width` - The width of the display in pixels, the default is 240.
- `height` - The height of the display in pixels, the default is 320.

Drawing

Once the display is initialized you're ready to perform basic fill and pixel drawing. First to fill the display with a solid color use the `fill` function:

```python
display.fill(color565(255, 0, 0))
```
You should see the display fill entirely with a solid red color after running the command above.

Notice how the `color565` function is called to get a color that's passed to the fill function. This `color565` function takes in the red, green, and blue color component values which should range from 0 (lowest intensity) to 255 (highest intensity). Try filling the display with different color values!

To clear the display back to black, fill it with a zero color value:

```
display.fill(0)
```

You can draw individual pixels with the `pixel` function. For example to draw a white pixel at the origin position 0, 0:
Or to draw a pixel at the opposite corner at position 239, 319:

display.pixel(239, 319, color565(255, 255, 255))

The **pixel** function takes the following parameters:

- X position of the pixel to draw.
- Y position of the pixel to draw.
- **Color of the pixel.** Use the `ili9341.color565` function to generate this color value from red, green, blue component values.

In addition to pixel drawing there's a filled rectangle drawing command called `fill_rectangle`. For example to draw a blue box in one quadrant of the screen run:

```python
display.fill_rectangle(0, 0, 120, 170, color565(0, 0, 255))
```

![Image of blue box on screen]

The `fill_rectangle` function takes the following parameters:

- X position of the rectangle upper left corner.
- Y position of the rectangle upper left corner.
- Width of the rectangle in pixels.
- Height of the rectangle in pixels.
- **Color of the rectangle.** Again use the `ili9341.color565` function to generate this value.

That's all there is to drawing on the ILI9341 display with CircuitPython! Right now only basic fill, pixel, and filled rectangle drawing commands are supported. However since this is a pixel-based display you can also draw text with the bitmap font library (https://adafrui.it/sA0). There's even a basic graphics library to draw lines and other shapes (https://adafrui.it/zdy)!
2.4" TFT FeatherWing

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://adafruit.it/vvE).

Parts

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

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.

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). 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.

https://adafruit.it/EGe

https://adafruit.it/EGe
First, make sure you are running the latest version of Adafruit CircuitPython (https://adafruit.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://adafruit.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafruit.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
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
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 = displayio.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(max_size=10)
display.show(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(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)

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.

```python
import board
import displayio
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.

```python
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.

```python
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.

```python
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 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.

```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 **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 on the shield with CircuitPython in this guide, but the library required for enabling resistive touch is 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)
In addition to CircuitPython there's an older MicroPython version of the TFT library that you can use with some MicroPython boards. Before you get started it will help to be familiar with these guides for working with MicroPython:

- MicroPython Basics: What is MicroPython? [link](https://adafru.it/pXa)
- MicroPython Basics: How to Load MicroPython on a Board [link](https://adafru.it/pNB)
- MicroPython Basics: Load Files & Run Code [link](https://adafru.it/s1f)

See all the MicroPython guides in the learning system [link](https://adafru.it/qzD) for more information.

MicroPython Module Install

To use the TFT display with your MicroPython board you'll need to install the micropython-adafruit-rgb-display MicroPython module [link](https://adafru.it/szC) on your board. Remember this module is for MicroPython.org firmware and not Adafruit CircuitPython!

First make sure you are running the latest version of MicroPython for your board. If you're using the ESP8266 MicroPython port you must be running version 1.8.5 or higher [link](https://adafru.it/sas) as earlier versions do not support using .mpy modules as shown in this guide.

Next download the latest ili9341.mpy and rgb.mpy file from the releases page [link](https://adafru.it/szD) of the micropython-adafruit-rgb-display GitHub repository [link](https://adafru.it/szC) and use a tool like ampy to copy the files to the board [link](https://adafru.it/r2B).

Usage

The following section will show how to control the ILI9341 display from the board’s Python prompt / REPL. First connect to the board’s serial REPL [link](https://adafru.it/pMf) so you are at the MicroPython `>>>` prompt.

SPI Initialization

On MicroPython.org firmware which uses the machine API you can initialize SPI like the MicroPython SPI guide mentions [link](https://adafru.it/sav). For example on the ESP8266 with TFT FeatherWing you can run:

```python
import machine
spi = machine.SPI(1, baudrate=32000000)
```

Notice how the baudrate is specifying the SPI bus clock speed at 32mhz. This means pixel data will be sent very quickly to the display--much quicker than a software SPI interface. These displays can actually run up to about 64mhz but the ESP8266 SPI hardware can't support that fast of a speed!

Display Initialization

Next to import the display library and initialize it run the following code:
import ili9341
display = ili9341.ILI9341(spi, cs=machine.Pin(0), dc=machine.Pin(15))

When creating the display instance of the ILI9341 class you'll need to know which pins are connected to the display's CS, DC, and optionally RST or reset line. For the TFT FeatherWing see its guide for details on these pin connections (https://adafruit.it/szF).

The CS and DC parameters to the ILI9341 class initializer are required and should be a pin instance. There are a few optional keyword arguments you can specify too:

- `rst` - This is a GPIO pin connected to the RST or reset line on the display. The default for this is to not be specified and reset is not used.
- `width` - The width of the display in pixels, the default is 240.
- `height` - The height of the display in pixels, the default is 320.

Drawing

After initializing SPI and the display you're ready to start drawing on it. The usage of the drawing library is exactly the same as with CircuitPython so check out the CircuitPython drawing information (https://adafruit.it/Cgx)–the same code will work on MicroPython too!