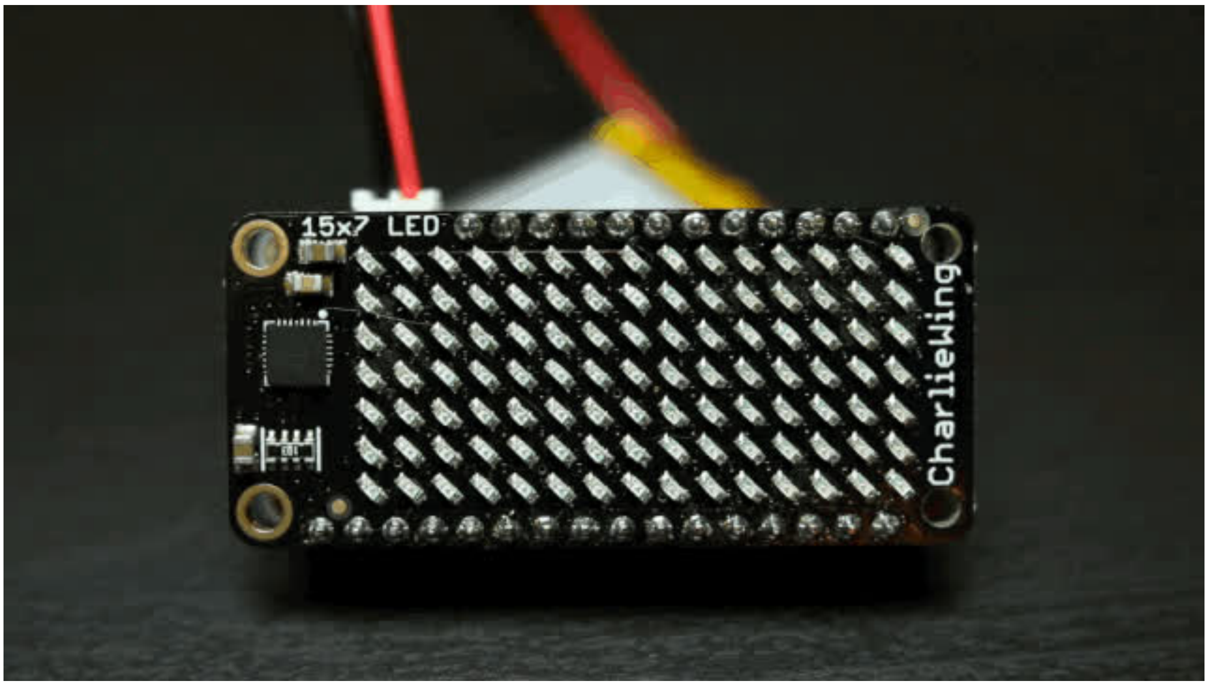
MicroPython Displays: Drawing Text

Created by Tony DiCola



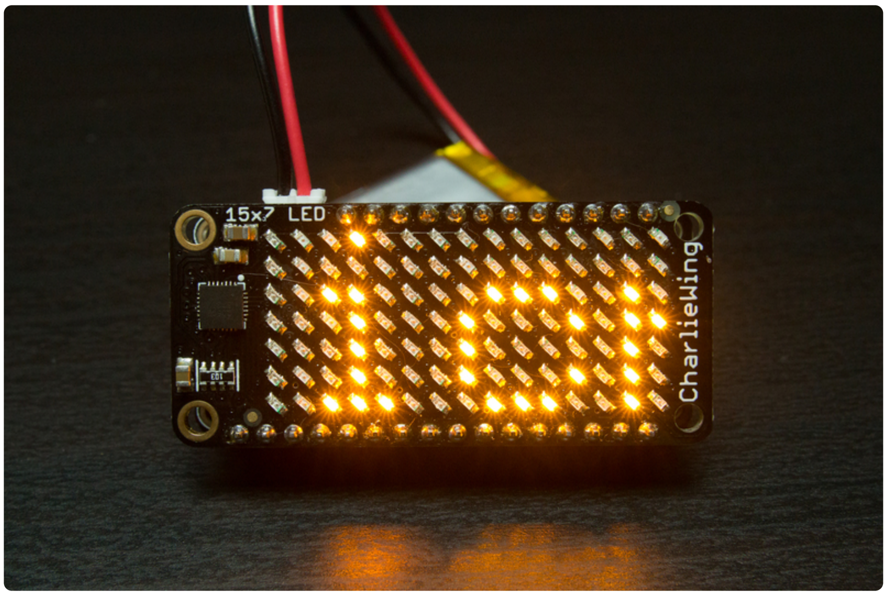
https://learn.adafruit.com/micropython-displays-drawing-text

Last updated on 2024-06-03 02:01:54 PM EDT

Table of Contents

Overview

Note this guide was written for MicroPython.org firmware and not Adafruit CircuitPython firmware, though parts are applicable to CircuitPython.



So you have a nifty display like a bright Charlieplex LED or NeoPixel matrix powered by MicroPython, but what if you want to write text on it? Most MicroPython display modules only give you basic pixel drawing commands so it's very cumbersome to draw text yourself pixel by pixel. Luckily there's a handy new MicroPython bitmap font module (https://adafru.it/spa) you can use to draw text on any pixel-based display! This module shows some of the power of MicroPython--a single module can work with **any** pixel display because of MicroPython's dynamic language features. In this guide you'll learn how to use a bitmap font rendering module with pixel-based displays like the Charlieplex LED matrix (https://adafru.it/spb) or simple LED backpack matrix (https://adafru.it/spc).

To follow this guide you'll want to be familiar with MicroPython by reading these guides:

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

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

In addition be sure to follow the guide for your pixel-based display first:

- MicroPython Hardware: LED Backpacks & FeatherWings (https://adafru.it/spc)
- MicroPython Hardware: Charlieplex LED Matrix (https://adafru.it/spb)
- MicroPython Hardware: SSD1306 OLED Display (https://adafru.it/spd) (note the OLED display already has a font rendering function, but you can use this guide and its font rendering too!)
- MicroPython Hardware: ILI9341 TFT & FeatherWing (https://adafru.it/sB4)

Hardware

For this guide there's no special hardware you need to draw text on a display. However you do need to have a pixel-based display of some sort (LED, TFT, NeoPixel--anything!) connected to your MicroPython board. Check out the following guides for details on how to use a few pixel displays with MicroPython:

- MicroPython Hardware: LED Backpacks & FeatherWings (https://adafru.it/spc)
- MicroPython Hardware: Charlieplex LED Matrix (https://adafru.it/spb)
- MicroPython Hardware: SSD1306 OLED Display (https://adafru.it/spd) (note the OLED display already has a font rendering function, but you can use this guide and its font rendering too!)
- MicroPython Hardware: ILI9341 TFT & FeatherWing (https://adafru.it/sB4)

Follow the appropriate guide to setup the hardware and make sure you can draw pixels on the display with MicroPython before continuing.

Software

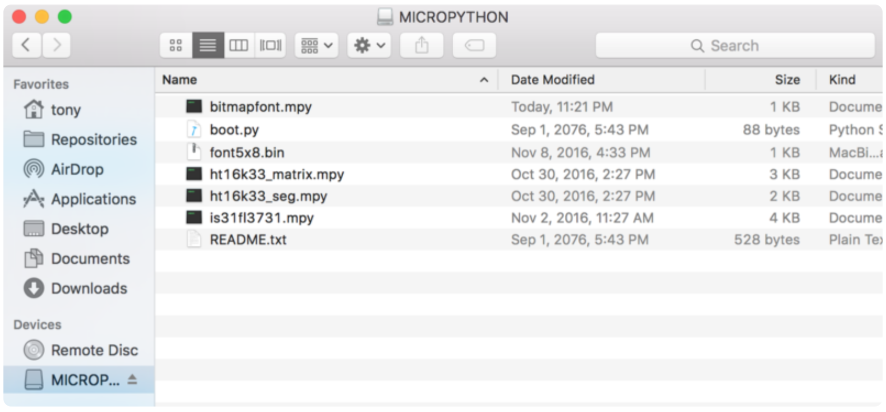
Install Module

To use the bitmap font module with your MicroPython board you'll need to install the micropython-adafruit-bitmap-font MicroPython module (https://adafru.it/spa) on your board.

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 **(https://adafru.it/sas)** as earlier versions do not support using .mpy modules as shown in this guide.

Download the latest **bitmapfont.mpy** and **font5x8.bin** file from the releases page (https://adafru.it/spe) of the micropython-adafruit-bitmap-font GitHub repository (https://adafru.it/spf).

If your board supports USB mass storage, like the SAMD21 MicroPython port, then simply drag the .mpy and other files to the board's file system (eject the drive and reset the board to make sure it is picked up by MicroPython).



Name	Date Modified	Size	Kind
bitmapfont.mpy	Today, 11:21 PM	1 KB	Docume
boot.py	Sep 1, 2076, 5:43 PM	88 bytes	Python S
font5x8.bin	Nov 8, 2016, 4:33 PM	1 KB	MacBi...
ht16k33_matrix.mpy	Oct 30, 2016, 2:27 PM	3 KB	Docume
ht16k33_seg.mpy	Oct 30, 2016, 2:27 PM	2 KB	Docume
is31fl3731.mpy	Nov 2, 2016, 11:27 AM	4 KB	Docume
README.txt	Sep 1, 2076, 5:43 PM	528 bytes	Plain Te

If your board doesn't support USB mass storage, like ESP8266 MicroPython boards, then use a tool like ampy to copy the file to the board (https://adafru.it/r2B).

Usage

The following section will show how to draw text on a matrix display like the Charlieplex FeatherWing/breakout or LED backpack.

Note that currently the bitmap font rendering library only supports a simple **5 pixel wide by 8 pixel tall character** font. This is great for large pixel displays like LED, NeoPixel, and Charlieplex matrices.

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

Display Initialization

Next you'll need to initialize your display so that you can draw pixels on it. Consult the MicroPython display guides (https://adafru.it/spA) for details on initializing displays.

For example the Charlieplex FeatherWing display initialization on ESP8266 MicroPython might look like:

```
import machine
import is31fl3731

i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
matrix = is31fl3731.CharlieWing(i2c)
```

Once you have a matrix object which has a function to draw pixels you're ready to start drawing text.

Bitmap Font Initialization

To use the bitmap font rendering module you'll need to import its module and create an instance of the **BitmapFont** class inside it. For example to create a font renderer for the Charlieplex FeatherWing above you would run:

```
import bitmapfont
bf = bitmapfont.BitmapFont(15, 7, matrix.pixel)
bf.init()
```

The BitmapFont class initializer takes three parameters:

1. **The maximum width of the display in pixels**, in this case 15 for the Charlieplex FeatherWing.
2. **The maximum height of the display in pixels**, in this case 7 for the Charieplex FeatherWing.
3. **A pixel drawing function to call when the font class needs to write a pixel.** For this example the Charlieplex matrix class **pixel** function is specified. It's important to note that to use the bitmap font class you **must** have a function it can call to draw pixels! This function can live anywhere, like as a global function or on a class instance. The function needs to take at least a pixel x position and pixel y position parameter (in that order), and any number of other positional and keyword parameters after them (like color, intensity, etc.).

Next the **init()** function is called to setup the bitmap font drawing class. Don't forget to call this function or else the font rendering won't work!

Note too when you're finished using the bitmap font drawing class you shouild call the **deinit()** function to free the font resources it used.

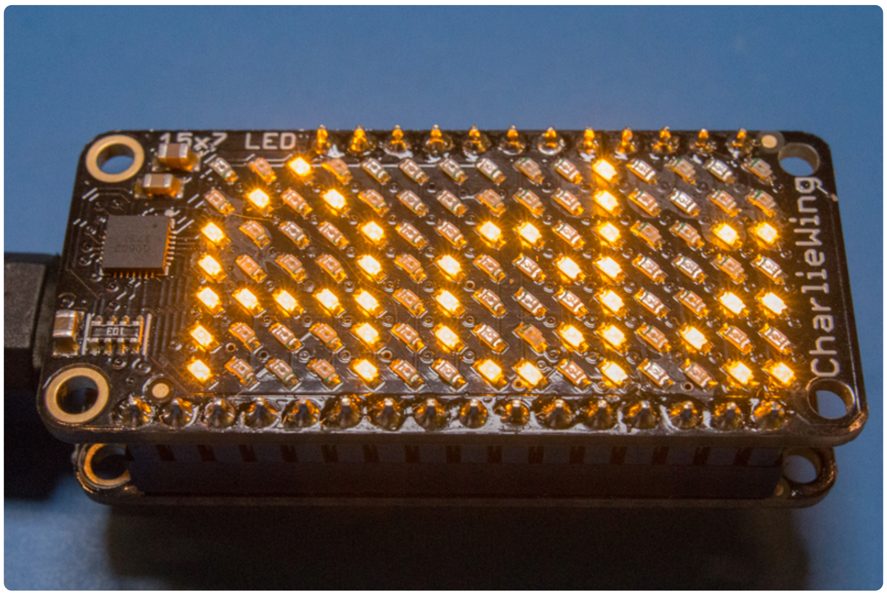
As an alternative to calling **init** and **deinit** you can use the **BitmapFont** class as a context manager (https://adafru.it/spB) that automatically initializes and deinitializes internal resources. The syntax might look like:

```
with bitmapfont.BitmapFont(15, 7, matrix.pixel) as bf:
    # Bitmap font rendering code goes here!
    # No need to call init &amp; deinit.
```

Drawing Characters and Text

Once the bitmap font drawing class is initialized you're ready to write text to the display. You can use the **text** function to draw a line of text anywhere on the display. For the Charlieplex matrix you can call:

```
bf.text('Ada', 0, 0, 100)
```

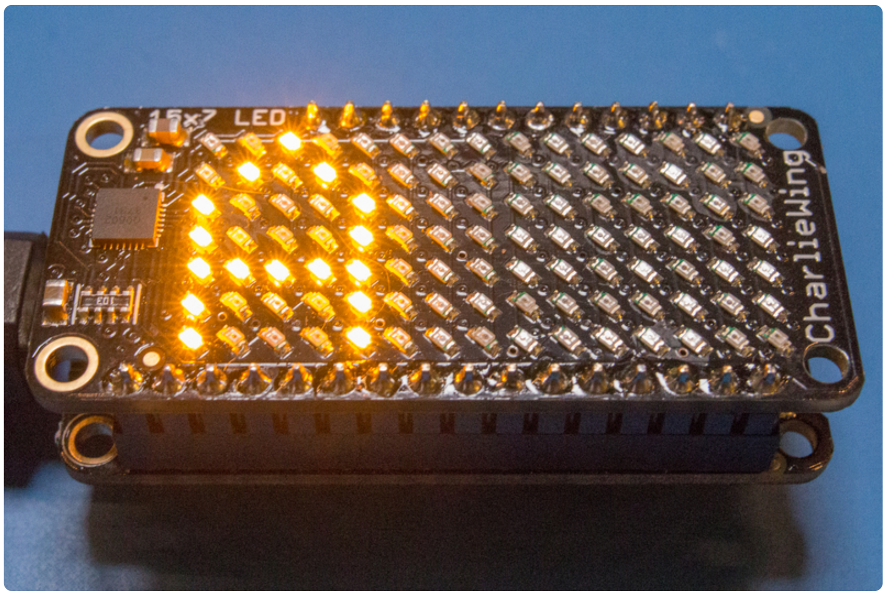


The parameters to the text function are:

- **The string of text to display.**
- **The x position on the display to start drawing text.**
- **The y position on the display to start drawing text.**
- **All the remaining parameters (both positional and keyword) are passed through to the pixel drawing function.** In this case the value 100 is passed to the Charlieplex matrix pixel function which will interpret it as the intensity of the LEDs. For other displays you can pass in color or other data to use when printing the text.

If you need you can draw a single character with the **draw_char** function, it's similar to text but only draws a single character. For example with the Charlieplex matrix:

```
matrix.fill(0) # Clear the display
bf.draw_char('A', 0, 0, 100)
```

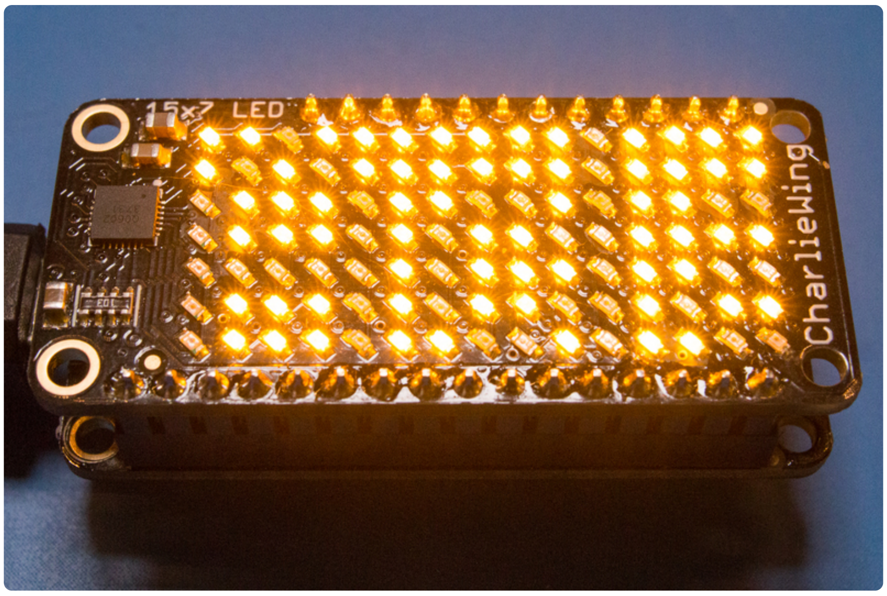


The parameters here are:

- **The character to print on the display.**
- **The x position on the display to start drawing the character.**
- **The y position on the display to start drawing the character.**
- **All the remaining parameters are passed through to the pixel drawing function** (see the text function description above).

Don't forget you can draw text in any color. For example to draw dark text on a bright background you might try:

```
matrix.fill(100)
bf.text('Ada', 0, 0, 0)
```



Notice how the last parameter to the text function changed from a bright value like 100 to 0. This means the display will turn off all the pixels where the text is drawn.

There's one more function on the bitmap font class that can be handy in some cases when animating or moving text, the **width** function to determine the pixel width of a string of text. For example to find the number of pixels wide the string 'Hello' will occupy you can call:

```
width = bf.width('Hello')
print('Hello is {} pixels wide.'.format(width))
```

Just pass a string of text to **width** and it will return the number of pixels in the X dimension that the string will occupy on the display.

Adapting To Other Displays

The power of the bitmap font class and MicroPython is that it's written in a way to work with any pixel-based display. As long as you can call a pixel drawing function which takes at least a pixel X and Y position you can draw text on it with this module. Just pass in the pixel drawing function to the **BitmapFont** class initializer as shown above.

For example to draw text on a 8x8 NeoPixel matrix you could first initialize the NeoPixels:

```
import neopixel
import machine
matrix = neopixel.NeoPixel(machine.Pin(6, machine.Pin.OUT), 64)
```

Then make a little pixel drawing helper function that takes an X, Y position and color as parameters and sets the appropriate pixel in the matrix:

```
def matrix_pixel(x, y, color):
    matrix[y*8 + x] = color
```

Notice how the pixel function has to convert a 2-dimensional X, Y location into a 1-dimensional location in the NeoPixel array. It does this by multiplying the Y position, or the current row, by the stride, or pixel length of each row in the matrix, and adding the X position within that row.

Finally create the font class and pass it the pixel function created above:

```
import bitmapfont
bf = bitmapfont.BitmapFont(8, 8, matrix_pixel)
bf.init()
```

Then draw some text! Remember the extra arguments to the text function will be passed along to the **matrix_pixel** function so you can use it to set the text color. To draw 'Ada' in pink you could run:

```
bf.text('Ada', 0, 0, (255, 0, 255))
```

That's all there is to using the bitmap font module to draw text on pixel displays!

Check out the library examples (https://adafru.it/uEn) to see how to scroll a text message across the display.