adafruit learning system

MicroPython Hardware: SSD1306 OLED Display

Created by Tony DiCola

Last updated on 2017-05-31 02:06:27 AM UTC

Guide Contents

Overview

Note this guide has been updated to show how to use the SSD1306 with both MicroPython.org and Adafruit CircuitPython firmware. There are small differences between the two so be careful to read and follow the instructions that call out these differences.

Small OLED (organic light emitting diode) displays are an easy way to add text and graphics to your project. These displays have beautifiul high contrast black and white graphics which are perfect for making simple interfaces, displaying sensor readings, creating a retro game, and much more. Even better you can now use OLED displays with MicroPython! This guide explores how to use SSD1306-based monochrome OLED displays with MicroPython.

The FeatherWing OLED (http://adafru.it/sao) in particular is a simple way to add an OLED display to a Feather-based board like the Feather HUZZAH ESP8266 (http://adafru.it/n6A) or Feather M0 (http://adafru.it/s1d). Simply plug in the the FeatherWing to the Feather board and you're all set to control it from MicroPython. No need for messy or confusing wiring!

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

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

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

In addition check out the OLED breakout guide (http://adafru.it/sap) for more details on OLED displays and the available boards in the shop.

Hardware

Parts

You'll need the following parts to follow this guide:

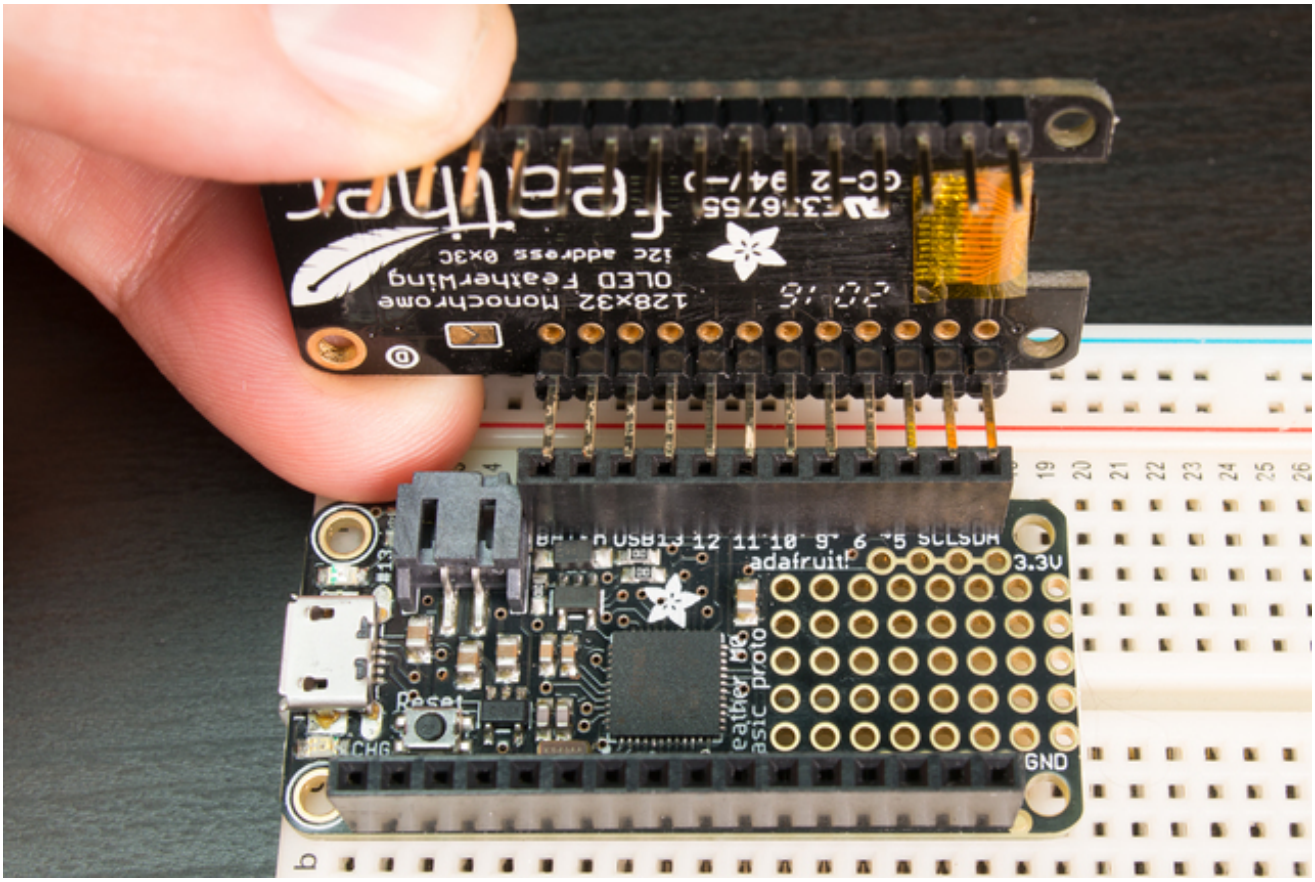
- **MicroPython board.** This guide focuses on the ESP8266 (http://adafru.it/n6A) and Feather M0/SAMD21-based boards (http://adafru.it/2772), but any MicroPython board that supports SPI or I2C should work.
    - If your board doesn't come with MicroPython running on it already then check out this how to load MicroPython on a board guide (http://adafru.it/saq).
    - If you're using a Feather board and FeatherWing OLED you probably want a Feather female header set (http://adafru.it/2886) or Feather stacking female header set (http://adafru.it/2830).
- **SSD1306-based OLED display.** If you're using a Feather board the FeatherWing OLED (http://adafru.it/2900) is a perfect option.
- **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 board and display. Check out the guide to excellent soldering (http://adafru.it/dxy) if you're new to soldering.

Make sure to follow the board and OLED display product guides to assemble and verify they work before continuing.

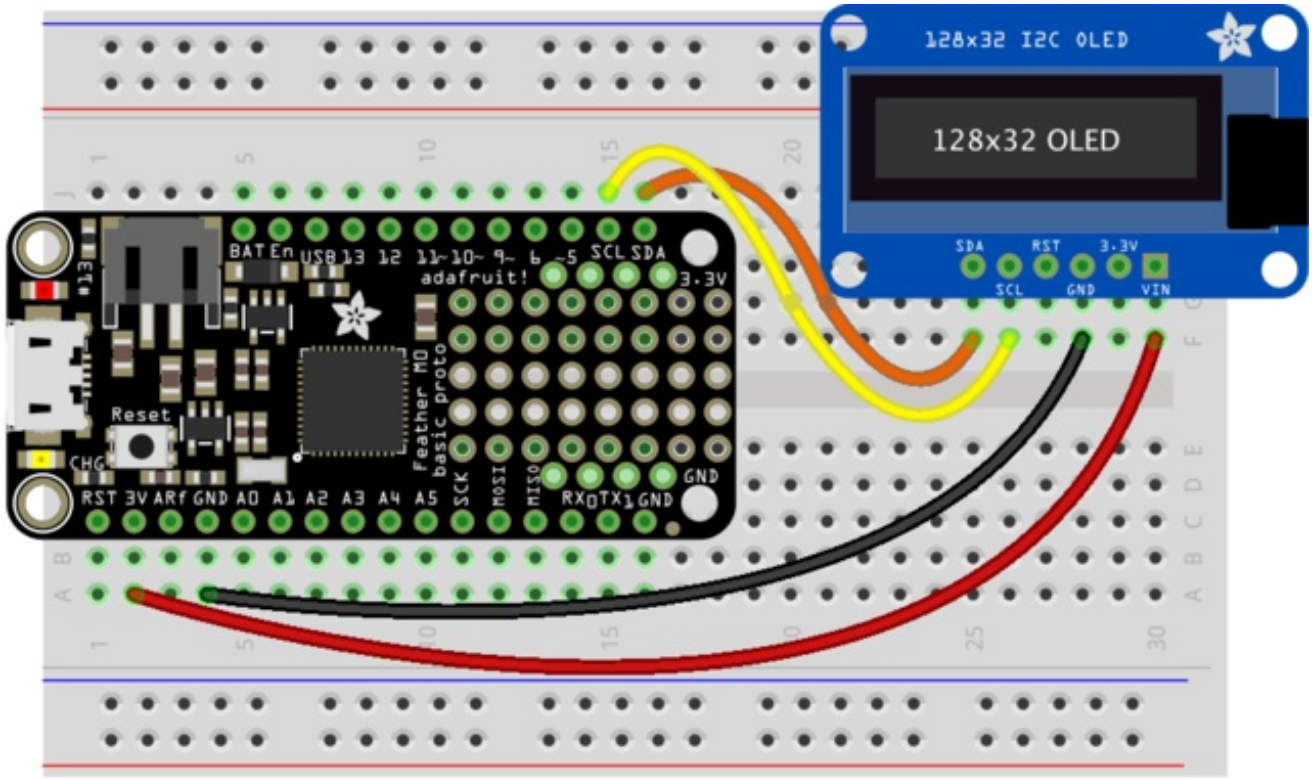
Wiring

There are two ways to wire the SSD1306 OLED to a board, either with an I2C or SPI connection. Some displays only support one connection type so be sure to read the product page and associated guide to understand how your display works.

For a Feather and FeatherWing OLED connecting them together is as easy as sliding the FeatherWing into the headers of the Feather. The wing will use an I2C connection to the board--nothing else is necessary to do. Skip to the next page on setting up the software!

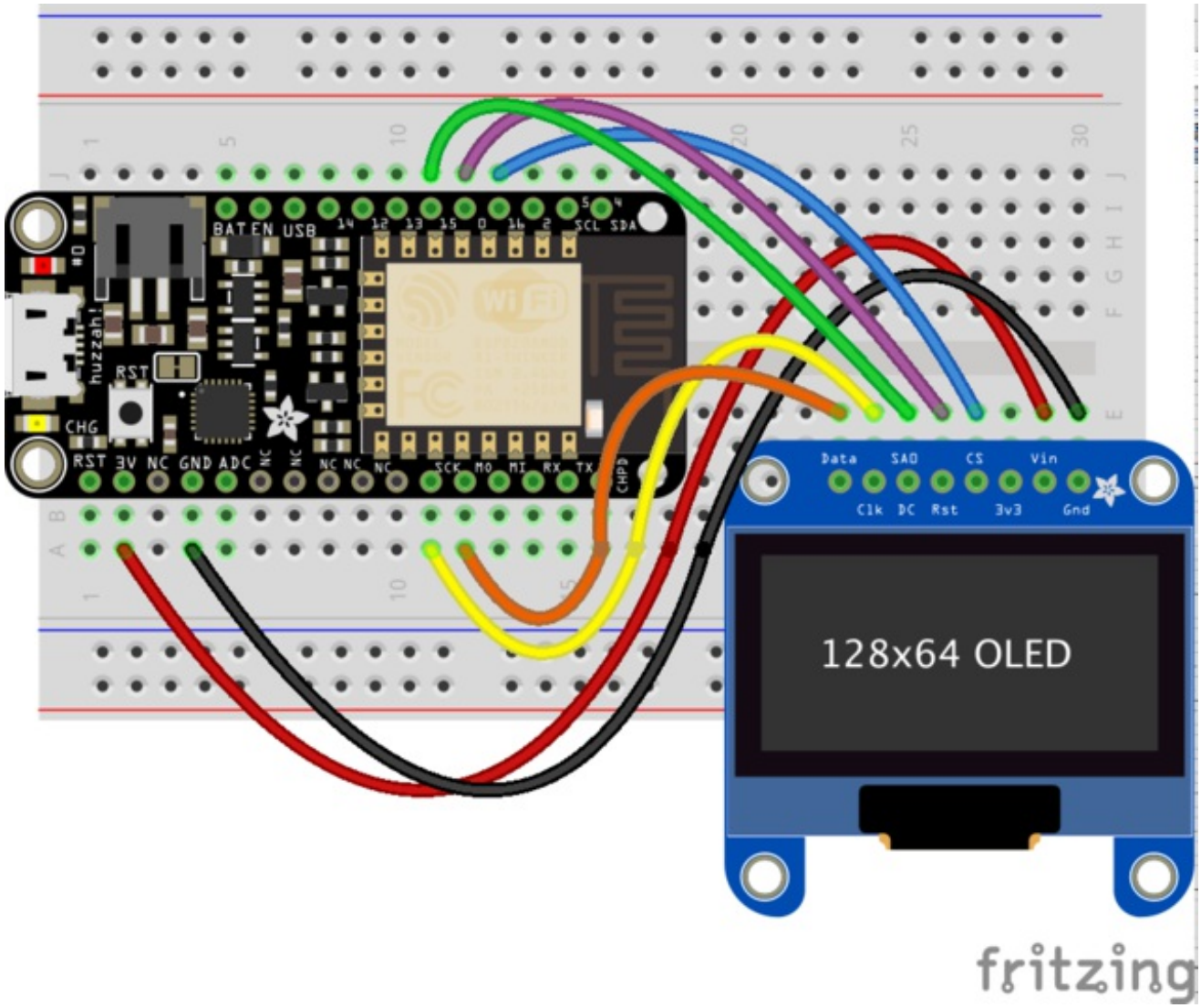


For an OLED with an **I2C interface** you'll want to wire it to the board's I2C and power pins. For example with a Feather M0 the connections might look like:



- **Board SCL** / **I2C clock** to **display SCL**.
- **Board SDA** / **I2C data** to **display SDA**.
- **Board 3.3V power** to **display VIN** / **voltage input**.
- **Board GND** / **ground** to **display GND** / **ground**.

For an OLED with a **SPI interface** you'll need to wire it to a few more pins on the board. For example with a Feather HUZZAH ESP8266 the connections might look like:



- **Board SCK** / **SPI clock** to **display CLK** / **clock**.
- **Board MOSI** / **data out** to **display Data** / **data in**.
- **Board 3.3V power** to **display Vin** / **voltage input**.
- **Board ground** to **display ground**.
- **Board digital input 15** (or any other digital IO pin) to **display DC** / **data/command**.
- **Board digital input 0** (or any other digital IO pin) to **display Rst** / **reset**.
- **Board digital input 16** (or any other digital IO pin) to **display CS** / **chip select**

Once your board is wired to the display continue on to learn how to use a MicroPython module to control the display!

Software

This page explains how to use this library with both MicroPython.org and Adafruit CircuitPython firmware. Pay careful attention to the differences as each firmware requires slightly different install and setup.

MicroPython Module Install

To use the display with your MicroPython board you'll need to install themicropython-adafruit-ssd1306 MicroPython module (http://adafru.it/sar) on your board. **Remember this module is for MicroPython.org firmware and not Adafruit CircuitPython!** Skip down to the CircuitPython Module Install section if you're using 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 (http://adafru.it/sas)** as earlier versions do not support using .mpy modules as shown in this guide.

Next download the latest **ssd1306.mpy** file from the releases page (http://adafru.it/sat) of the micropython-adafruit-ssd1306 GitHub repository (http://adafru.it/sar). Once the ssd1306.mpy file is on your computer you'll need to copy it to your MicroPython board's file system and can use a tool like ampy to copy the files to the board(http://adafru.it/r2B).

Adafruit CircuitPython Module Install

To use the SSD1306 with your Adafruit CircuitPython (http://adafru.it/tCy) board you'll need to install the Adafruit_CircuitPython_SSD1306 (http://adafru.it/u1f) module on your board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!** Jump back up to the MicroPython Module Install section if you're using MicroPython firmware.

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

Next download the latest **adafruit_ssd1306.zip** file from the releases page of the Adafruit_CircuitPython_SSD1306 GitHub repository (http://adafru.it/u1A). You'll need to unzip this file and copy the entire **adafruit_ssd1306** directory to the board's root filesystem.

If your board supports USB mass storage, like the SAMD21 CircuitPython port, 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 (http://adafru.it/u1d).**

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

In addition you'll need both the Adafruit CircuitPython Bus Device (http://adafru.it/u0b) and Adafruit CircuitPython Register (http://adafru.it/u0c) modules installed on your board. Just like installing the module as mentioned above, download the latest release .zip file and copy the folder inside it to the board's root filesystem for each:

- Adafruit CircuitPython Bus Device Module releases (http://adafru.it/u0d)
- Adafruit CircuitPython Register Module releases (http://adafru.it/u0e)

Furthermore, CircuitPython builds after 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 (http://adafru.it/vcB) and copy it to the board as well.

Before continuing make sure your board's root filesystem has the **adafruit_ssd1306**, **adafruit_bus_device**, **adafruit_register** and **framebuf** folders/modules copied over.

Usage

The following section will show how to control the display from the board's Python prompt / REPL. You'll learn how to interactively control the display by typing in the code below, then later you can use the same code in your own MicroPython scripts.

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

Import SSD1306 Module

First you need to import the SSD1306 module to use it in your own code. This differs slightly between the MicroPython.org library and Adafruit CircuitPython library so pay

careful attention to the command to run.

For MicroPython.org firmware and the micropython-adafruit-ssd1306 module run this command:

```
import ssd1306
```

For Adafruit CircuitPython firmware and the Adafruit_CircuitPython_SSD1306 module run this command:

```
import adafruit_ssd1306 as ssd1306
```

I2C Initialization

If your display is connected to the board using I2C (like if using a Feather and the FeatherWing OLED) you'll first need to initialize the I2C bus. Note the I2C initialization differs slightly for MicroPython vs. Adafruit CircuitPython so pay careful attention to the right section below for your firmware.

MicroPython I2C Initialization

On MicroPython.org firmware which uses the machine API you can initialize I2C like the MicroPython I2C guide mentions (http://adafru.it/sau). For example on a board like the ESP8266 you can run (assuming you're using the default SDA gpio #4 and SCL gpio #5 pins like on a Feather & SSD1306 FeatherWing):

```
import machine
i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
```

Adafruit CircuitPython I2C Initialization

On CircuitPython the I2C initialization is slightly different as it uses the board module to more easily address board pins, and busio or bitbangio modules for I2C access.

First import the necessary modules:

```
from board import *
import busio
```

Now for either board run this command to create the I2C instance using the default SCL and SDA pins (which will be marked on the boards pins if using a Feather or similar Adafruit board):

```
i2c = busio.I2C(SCL, SDA)
```

You can also use a context manager for I2C as mentioned in the SAMD21 MicroPython guide (http://adafru.it/s1a). If you aren't using a context manager be sure to call **deinit** when finished with the I2C bus.

I2C Initialization Continued

After initializing the I2C interface for your firmware as described above you can create an instance of the SSD1306 I2C driver by running:

```
oled = ssd1306.SSD1306_I2C(128, 32, i2c)
```

Note that the first two parameters to the SSD1306_I2C class initializer are the **width** and **height** of the display in pixels. Be sure to use the right values for the display you're using!

At this point your I2C bus and display are initialized, skip down to the drawing section.

SPI Initialization

If your display is connected to the board using SPI you'll first need to initialize the SPI bus. Note the SPI bus initialization differs slightly for MicroPython vs. Adafruit CircuitPython so pay careful attention to the right section below for your firmware.

MicroPython SPI SSD1306 Initialization

On MicroPython.org firmware which uses the machine API you can initialize SPI like the MicroPython SPI guide mentions (http://adafru.it/sav).

```
import machine
spi = machine.SPI(1, baudrate=8000000, polarity=0, phase=0)
oled = ssd1306.SSD1306_SPI(128, 32, spi, machine.Pin(15), machine.Pin(0), machine.Pin(16))
```

Note the first two parameters to the SSD1306_SPI class initializer are the **width** and **height** of the display in pixels. Be sure to use the right values for the display you're using!

The next parameters to the initializer are the pins connected to the display's **DC**, **reset**, and **CS** lines in that order.

Adafruit CircuitPython SPI SSD1306 Initialization

On CircuitPython the SPI initialization is slightly different as it uses the board module to more easily address board pins, and busio module for SPI access. Run the following

commands:

```
from board import *
import busio
spi = busio.SPI(clock=CLK, MOSI=MOSI, MISO=MISO)
oled = ssd1306.SSD1306_SPI(128, 32, spi, GPIO15, GPIO0, GPIO16)
```

Note the first two parameters to the SSD1306_SPI class initializer are the **width** and **height** of the display in pixels. Be sure to use the right values for the display you're using!

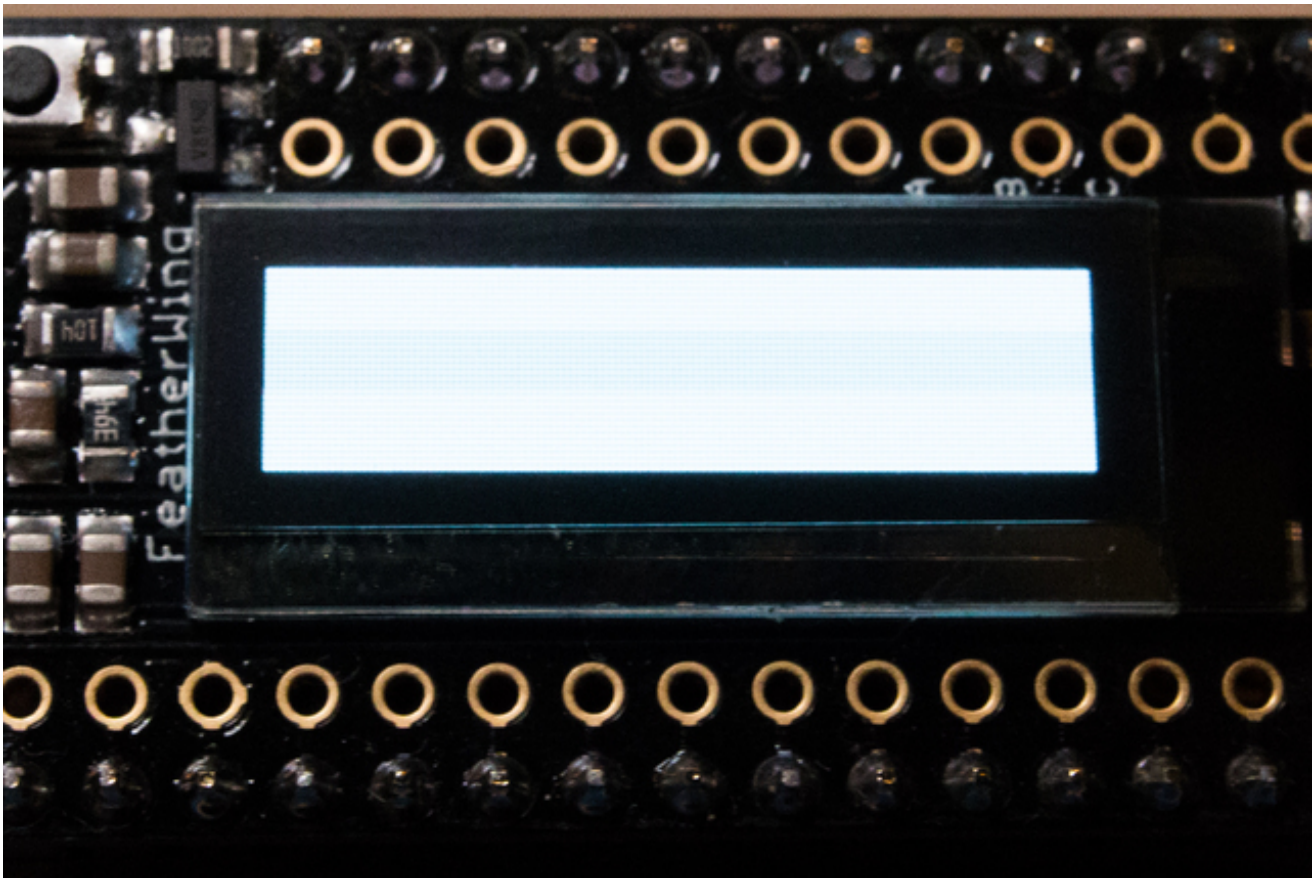
The next parameters to the initializer are the pins connected to the display's **DC**, **reset**, and **CS** lines in that order.

Drawing

The SSD1306 MicroPython module currently supports a basic set of commands to draw on the display. You can set individual pixels, fill the screen, and write lines of text. There aren't yet more advanced functions like line drawing, etc. but check the module GitHub repository (http://adafru.it/sar) for future releases which might include more functions.

To fill or clear the entire screen use the fill function. This function takes a parameter which specifies the color to fill with, either 0 for black or 1 for white. For example to fill the screen white:

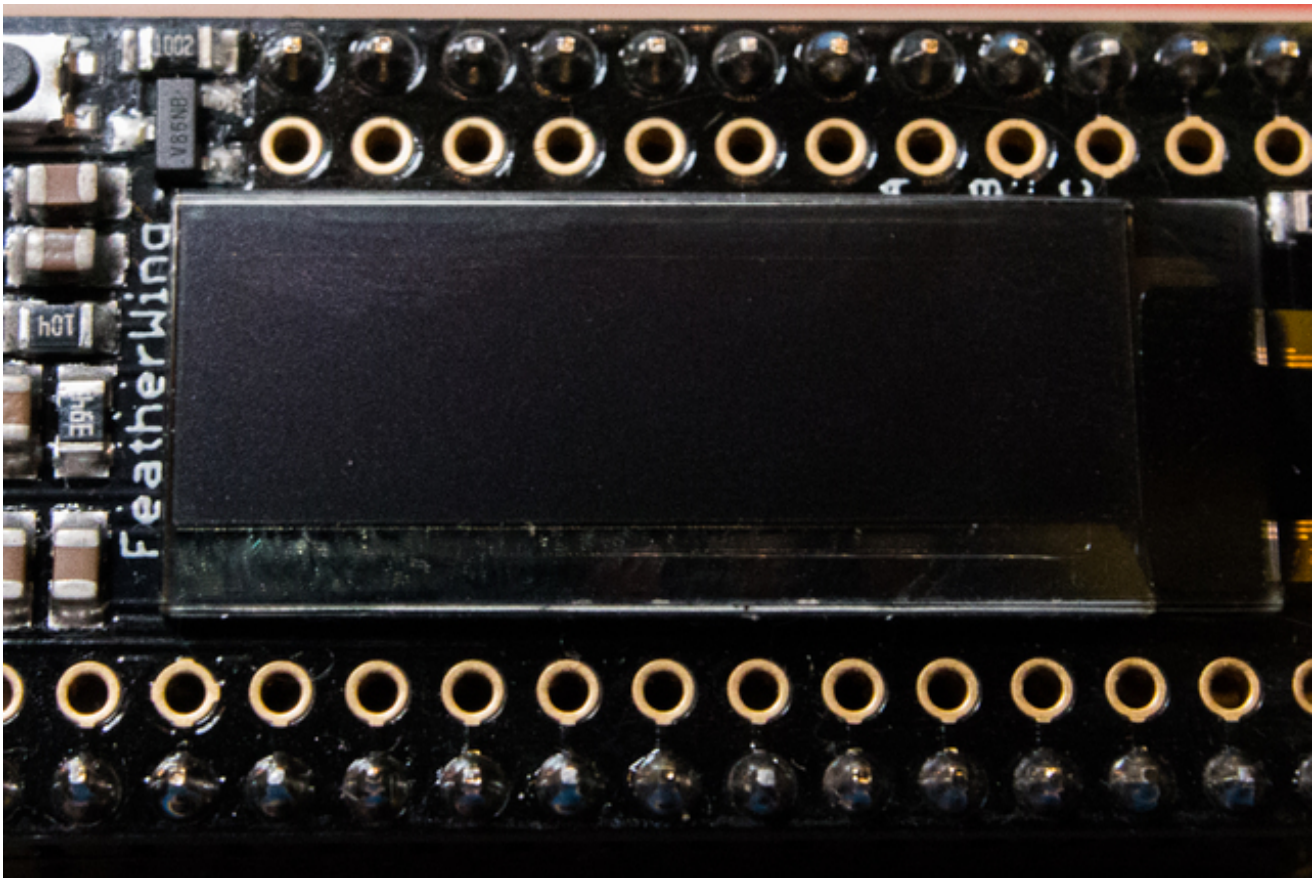
```
oled.fill(1)
oled.show()
```



Notice the **fill** function doesn't actually change the display. You must call **show** after making drawing commands to send the updated pixel data to the display!

To clear the screen to black just call **fill** again but with the color 0:

```
oled.fill(0)
oled.show()
```

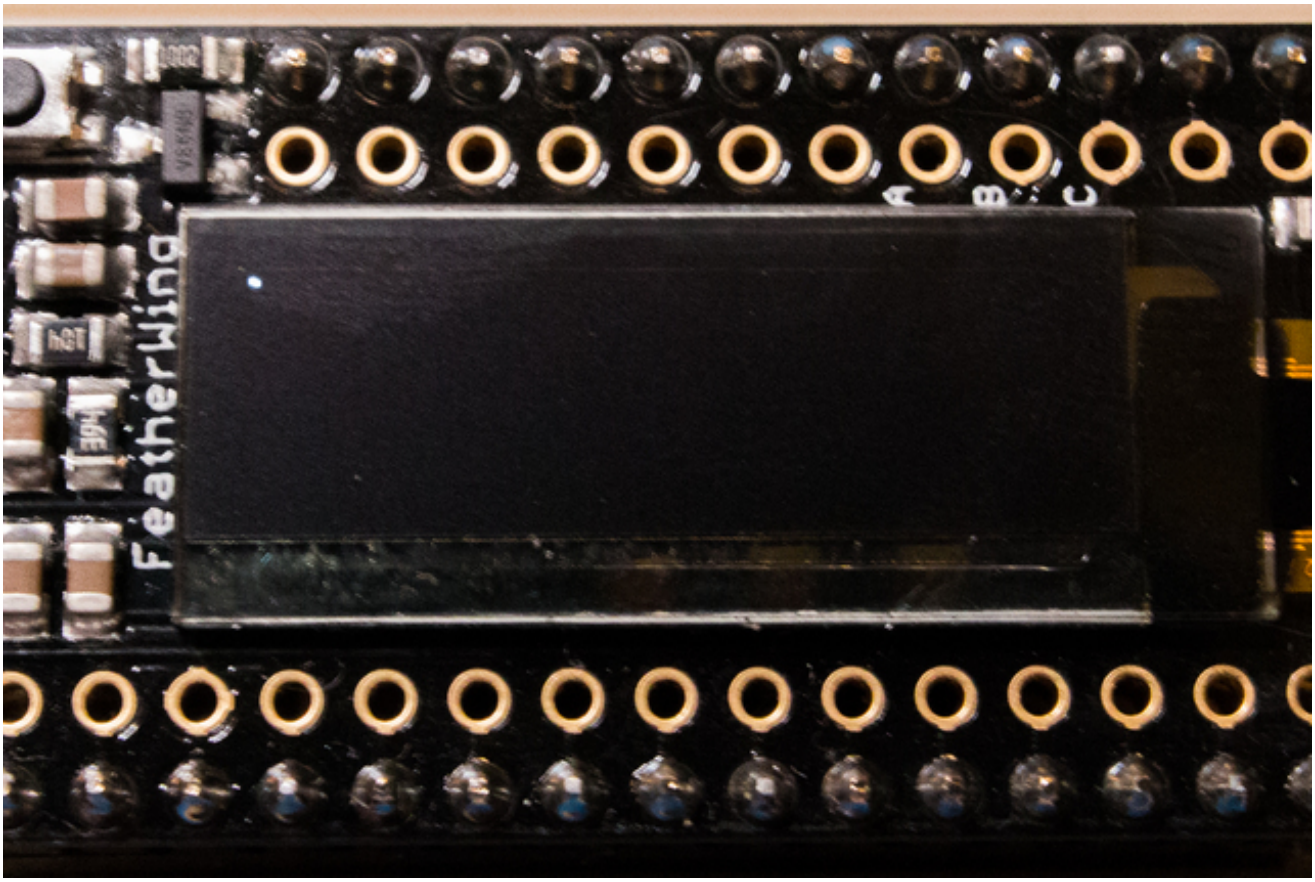


To set a pixel use the **pixel** function. This function takes the following parameters:

- **Pixel X position**
- **Pixel Y position**
- **Pixel color** (0 = black, 1 = white)

For example to set the first pixel white:

```
oled.pixel(0, 0, 1)
oled.show()
```



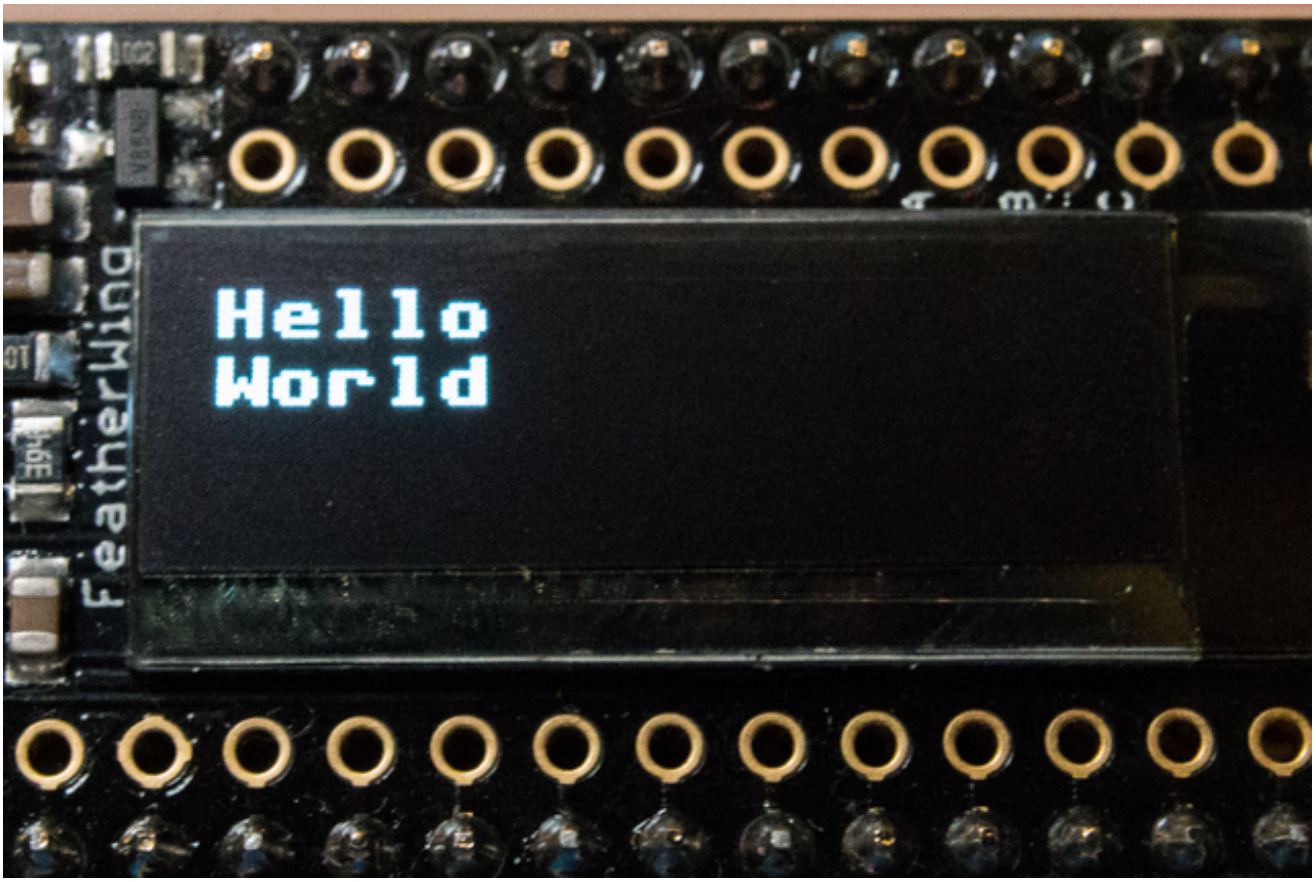
Try setting other pixels white by changing the X and Y position. Remember you have to call **show** after setting pixels to see them appear!

You can write a line of text with the**text** function. This function takes the following parameters:

- **String of text**
- **Text X position**
- **Text Y position**
- **Optional text color** (0 = black, 1 = white, the default)

For example to clear the screen and then write two lines of text:

```
oled.fill(0)
oled.text('Hello', 0, 0)
oled.text('World', 0, 10)
oled.show()
```



Notice the second line of text starts at Y position 10, this moves it down the display 10 pixels so it's below the first line of text. The font used by the text function is 8 pixels tall so a size of 10 gives a bit of room between the lines.

Note you can't currently change the font without digging into the MicroPython source port (check the extmod/modframebuf module source if you are curious).

Finally you can invert the display colors with the**invert** function:

```
oled.invert(True)
```

Note that the invert function doesn't need to have show called after it to see the change.

To go back to a non-inverted display run:

```
oled.invert(False)
```

That's all there is to drawing on the SSD1306 OLED display with MicroPython! The drawing functions are basic but provide building blocks for more advanced usage. For example you can display text with sensor readings or other state, or even program a simple game like pong!