CircuitPython Hardware: SSD1306 OLED Display

Created by Tony DiCola

https://learn.adafruit.com/micropython-hardware-ssd1306-oled-display

Last updated on 2021-11-15 06:48:23 PM EST
# Table of Contents

## Overview

### Hardware
- Parts 4
- Wiring 5

### CircuitPython
- Adafruit CircuitPython Module Install 8
- Bundle Install 8
- Usage 9
- I2C Initialization 9
- SPI Initialization 10
- Drawing 11

### MicroPython
- MicroPython Module Install 16
- Usage 16
- I2C Initialization 16
- SPI Initialization 17
- Drawing 17
Small OLED (organic light emitting diode) displays are an easy way to add text and graphics to your project. These displays have beautiful 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 CircuitPython and MicroPython! This guide explores how to use SSD1306-based monochrome OLED displays with CircuitPython and MicroPython.

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

Be sure to check out the OLED breakout guide (https://adafruit.it/sap) for more details on OLED displays and the available boards in the shop.
Hardware

The examples in this guide are no longer supported. Check out this OLED guide for CircuitPython usage: https://learn.adafruit.com/adafruit-oled-featherwing/python-circuitpython

Parts

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://adafru.it/2772), but any CircuitPython board that supports I2C 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.

If you're using a Feather board and FeatherWing 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://adafruit.it/2900) is a perfect option.

Breadboard (http://adafruit.it/64) and jumper wires (http://adafruit.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://adafruit.it/136). You’ll need to solder headers to the boards. Check out the guide to excellent soldering (https://adafruit.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, and the reset pin, if it's available. 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 digital #5 to display RST (reset). This can be any available digital pin.
- 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 CircuitPython module to control the display!

**CircuitPython**

The examples in this guide are no longer supported. Check out this OLED guide for CircuitPython usage: https://learn.adafruit.com/adafruit-oled-featherwing/python-circuitpython
Adafruit CircuitPython Module Install

To use the SSD1306 with your Adafruit CircuitPython board you'll need to install the Adafruit_CircuitPython_SSD1306 (https://adafruit.it/u1f) module on your board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafruit.it/Amd) for your board.

Bundle Install

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 like the, you'll need to manually install the necessary libraries from the bundle:

- adafruit_ssd1306
- 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.

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 smaller M0 boards like the Trinket and Gemma 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. You do not need to do this for Express CircuitPython boards (they have enough space for framebuf!)

Before continuing make sure your board's lib folder or root filesystem has at least the adafruit_ssd1306, adafruit_bus_device, adafruit_register, and if necessary framebuf 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 LED display and learn how to use the CircuitPython module built for the display.

Next connect to the board's serial REPL (https://adafruit.it/Awz) so you are at the CircuitPython >>> prompt.

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. First import the necessary modules:

```python
import board
import busio as io
```

Note if you're using the ESP8266 or other boards which do not support hardware I2C you need to import from the bitbangio module instead of busio:

```python
import board
import bitbangio as io
```
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):

```python
i2c = io.I2C(board.SCL, board.SDA)
```

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

```python
import adafruit_ssd1306
oled = adafruit_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!

128 x 64 size OLEDs (or changing the I2C address)

If you are using a 128x64 display, the I2C address is probably different (0x3d), unless you've changed it by soldering some jumpers:

```python
oled = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3d)
```

Adding hardware reset pin

If you have a reset pin (which may be required if your OLED does not have an auto-reset chip like the FeatherWing) also pass in a reset pin like so:

```python
from digitalio import DigitalInOut
reset_pin = DigitalInOut(board.D5) # any pin!
od = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c, reset=reset_pin)
```

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. Run the following commands:
import adafruit_ssd1306
import board
import busio
from digitalio import DigitalInOut

spi = busio.SPI(board.CLK, MOSI=board.MOSI, MISO=board.MISO)
dc_pin = DigitalInOut(board.D5)    # any pin!
reset_pin = DigitalInOut(board.D6) # any pin!
cs_pin = DigitalInOut(board.D9)    # any pin!

oled = adafruit_ssd1306.SSD1306_SPI(128, 32, spi, dc_pin, reset_pin, cs_pin)

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. Again make sure to use the right pin names as you have wired up to your board!

Drawing

The SSD1306 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](https://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:

```python
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:

```python
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:

```plaintext
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:

```plaintext
oled.fill(0)
oled.text('Hello', 0, 0)
oled.text('World', 0, 10)
oled.show()
```
Note that not all boards support built-in text rendering. In particular the non-express boards like Gemma and Trinket M0 don't have the memory to build in text drawing. You can however on those boards load a separate Python text drawing library to draw text. See the [Drawing Text](https://adafruit.it/sA0) guide!

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.

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

```python
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:

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

That's all there is to drawing on the SSD1306 OLED display with CircuitPython! 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!

You can find more drawing functions for basic shapes and text by following these guides (which work with both CircuitPython and MicroPython):

- [Drawing Shapes](https://adafruit.it/zdy)
- [Drawing Text](https://adafruit.it/sA0)

---

**MicroPython**

The examples in this guide are no longer supported. Check out this OLED guide for CircuitPython usage: https://learn.adafruit.com/adafruit-oled-featherwing/python-circuitpython

Note this page describes how to use a MicroPython.org version of this library with MicroPython boards. Skip back to the previous page if you're using a CircuitPython board like the Feather M0 express!
In addition to CircuitPython there's an older MicroPython version of the SSD1306 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: 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.

**MicroPython Module Install**

To use the display with your MicroPython board you'll need to install the micropython-adafruit-ssd1306 MicroPython module (https://adafru.it/sar) 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 (https://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 (https://adafru.it/sat) of the micropython-adafruit-ssd1306 GitHub repository (https://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 (https://adafru.it/r2B).

**Usage**

The following section will show how to control the SSD1306 from the board's Python prompt / REPL. First connect to the board's serial REPL (https://adafru.it/pMf) so you are at the MicroPython >>> prompt.

**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. On MicroPython.org
firmware which uses the machine API you can initialize I2C like the MicroPython I2C guide mentions (https://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):

```python
import machine
import ssd1306
i2c = machine.I2C(-1, machine.Pin(5), machine.Pin(4))
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!

**SPI Initialization**

If your display is connected to the board using SPI you'll first need to initialize the SPI bus. On MicroPython.org firmware which uses the machine API you can initialize SPI like the MicroPython SPI guide mentions (https://adafru.it/sav):

```python
import machine
import ssd1306
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.

**Drawing**

Once the display is initialized using the code above you're ready to start drawing on it. Follow the drawing section on the CircuitPython page (https://adafru.it/zzd) for all the details on pixel, fill, and other drawing functions. The usage of the library between CircuitPython and MicroPython is exactly the same!