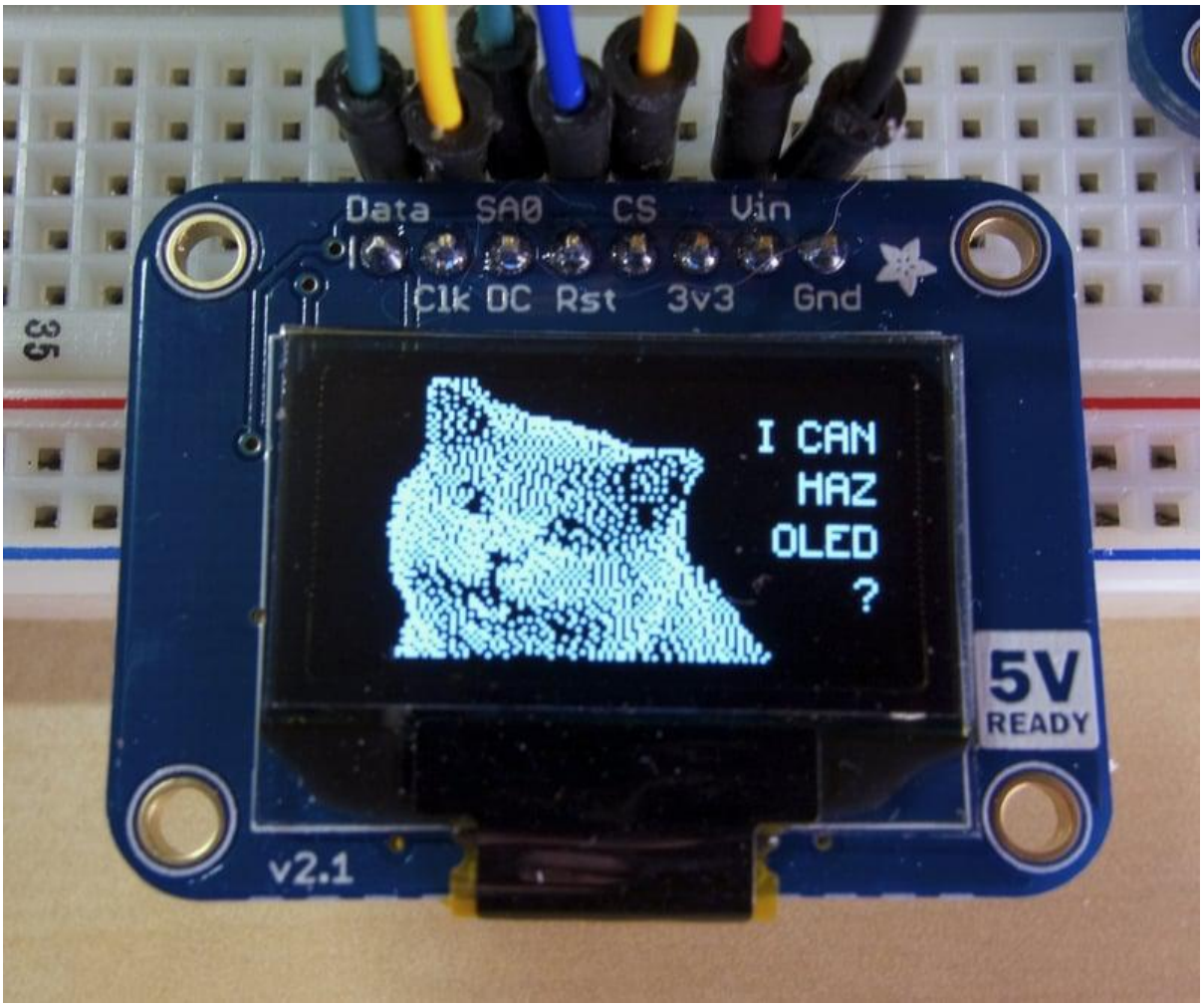
SSD1306 OLED Displays with Raspberry Pi and BeagleBone Black

Created by Tony DiCola



https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black

Last updated on 2022-12-01 02:12:39 PM EST

Table of Contents

Overview

This code is discontinued - Check out our newer tutorial at - https://learn.adafruit.com/monochrome-oled-breakouts/python-wiring

Are you looking for a bright graphical display to use with you Raspberry Pi or BeagleBone Black project? Consider using one of the SSD1306-based OLED displays (), with the SSD1306 Python library ()!

Although they're small (only an inch or so in size), these displays produce a beautiful and crisp 128x32 or 128x64 pixel image. Connecting the display to a Raspberry Pi or BeagleBone Black is easy too thanks to the display's 3.3 volt support, and I2C or SPI interface.

This guide will walk you through how to connect the display to a Raspberry Pi or BeagleBone Black, and how to install and use the SSD1306 Python library. Before you get started it will help to read the guide on their usage () so you know how the displays are assembled and configured. Make sure your Raspberry Pi is running the latest Raspbian () or Occidentalis () operating system, or your BeagleBone Black is running the official Debian () or a Debian-based operating system like Ubuntu.

Wiring

This code is discontinued - Check out our newer tutorial at - https://learn.adafruit.com/monochrome-oled-breakouts/python-wiring

Raspberry Pi

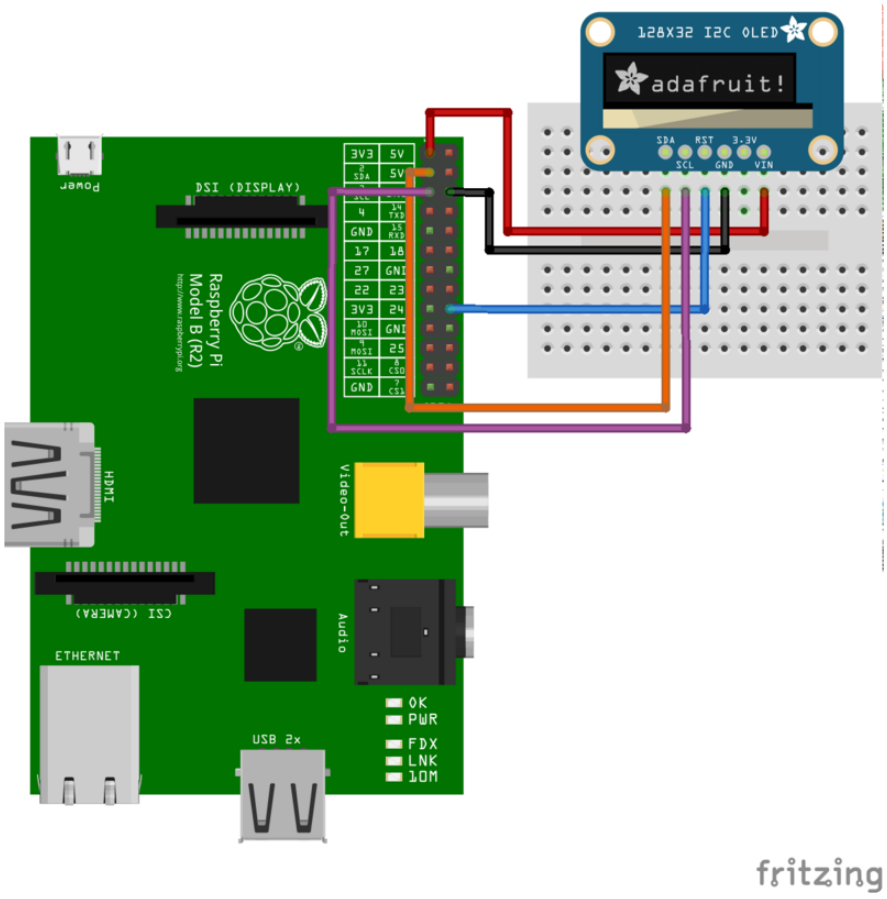
You can hook up an OLED to the Raspberry Pi using either the Pi's I2C or SPI interface. If your OLED supports both I2C and SPI, make sure to check how the solder jumpers are configured to expose the right interface ().

In general, I2C uses fewer pins but is slower. SPI is way fast, but requires a bunch extra pins. Choose your adventure based on your needs!

I2C

To use the Pi with an I2C display wire it up as follows:

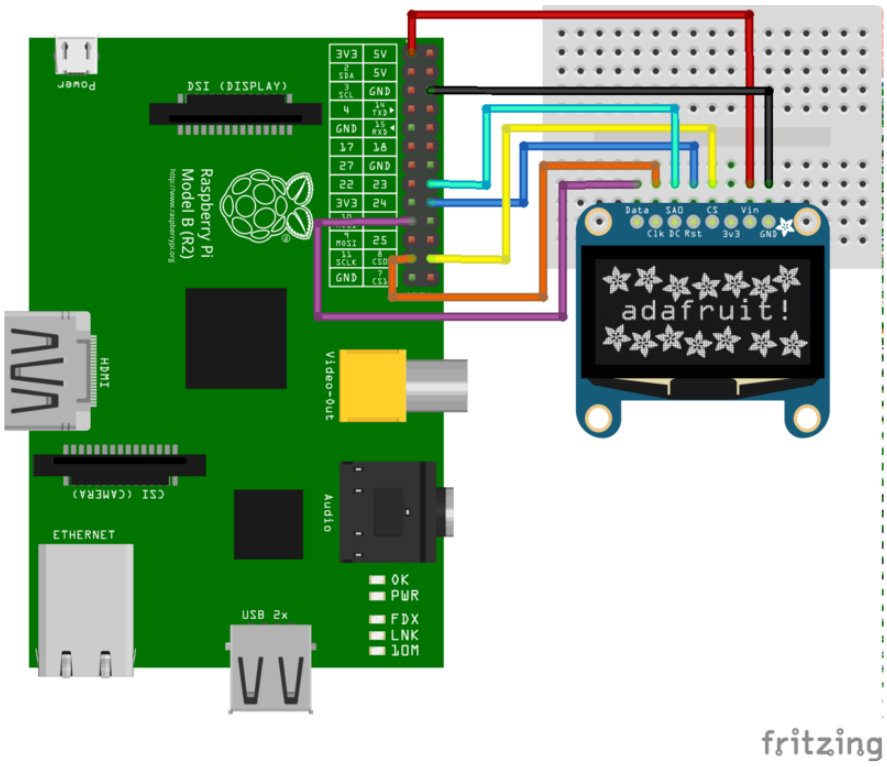
NOTE: Make sure to enable I2C on your Pi () before wiring it up!



- Connect display ground to Raspberry Pi ground (black wire).
- Connect display VIN to Raspberry Pi 3.3 volt (red wire).
- Connect display RST to Raspberry Pi GPIO 24 (blue wire). You can alternatively use any free digital GPIO pin for the reset pin.
- Connect display SCL to Raspberry Pi SCL (purple wire).
- Connect display SDA to Raspberry Pi SDA (orange wire).

SPI

To use the Pi with an SPI display, wire it up as follows:



- Connect display ground to Raspberry Pi ground (black wire).
- Connect display VIN to Raspberry Pi 3.3 volt (red wire).
- Connect display CS to Raspberry Pi CE0 (yellow wire).
- Connect display RST to Raspberry Pi GPIO 24 (blue wire). You can alternatively use any free digital GPIO pin for the reset pin.
- Connect display DC to Raspberry Pi GPIO 23 (cyan wire). You can alternatively use any free digital GPIO pin for the DC pin.
- Connect display CLK to Raspberry Pi SCLK (orange wire).
- Connect display Data to Raspberry Pi MOSI (purple wire).

Note that the wiring above will use the Raspberry Pi's hardware SPI bus to communicate with the display. If you haven't done so already with your Pi, make sure to edit the blacklist.conf file () to comment the line which disables SPI. Reboot your Pi and you should see the files /dev/spidev0.0 and /dev/spidev0.1 are now available.

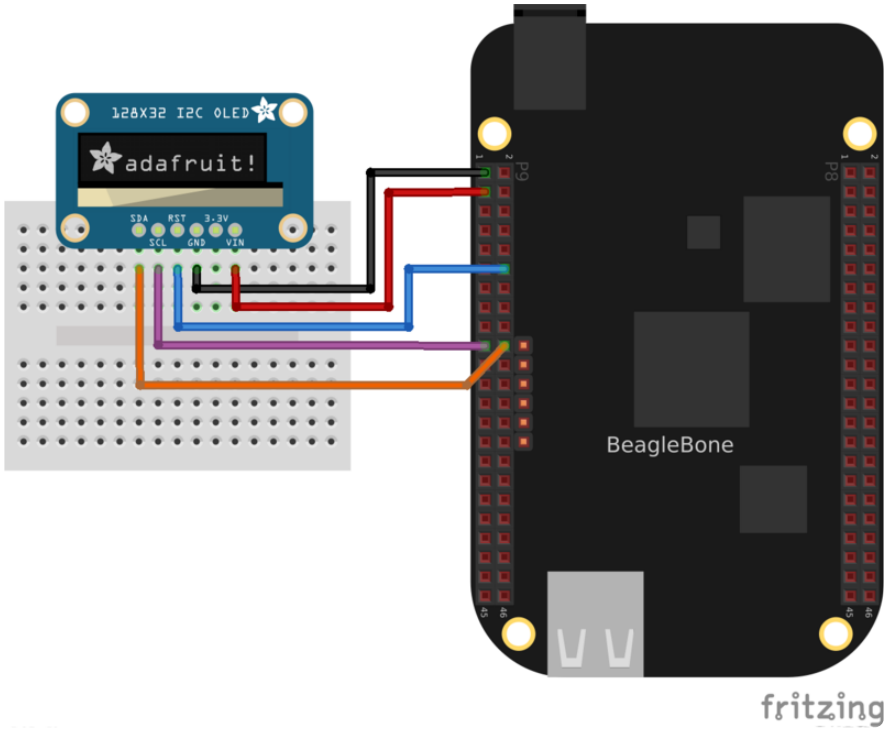
Using hardware SPI is great for getting the fastest response from the display, however if you need more flexibility with pin usage you can use a software-based SPI implementation with any 5 free digital GPIO pins. See the example code usage on the next page for more information about configuring software SPI.

BeagleBone Black

Just like with the Raspberry Pi, you can use an OLED display with the BeagleBone Black over either the SPI or I2C interface.

I2C

To use the BeagleBone Black with an I2C display, wire it up as follows:

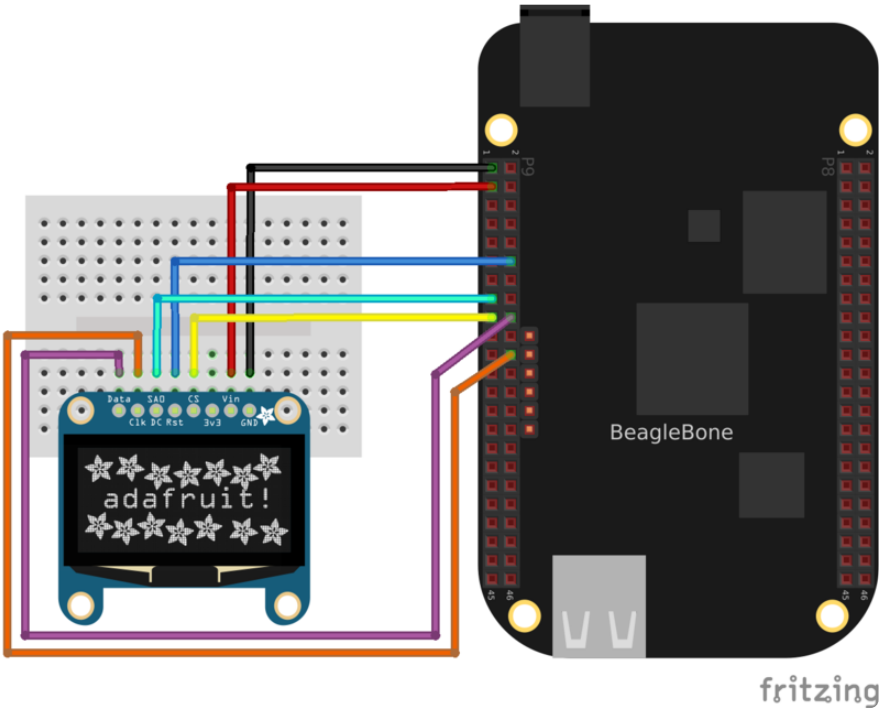


- Connect display ground to BeagleBone Black ground (black wire).
- Connect display VIN to BeagleBone Black 3.3 volt (red wire).
- Connect display RST to BeagleBone Black P9_12 (blue wire). You can alternatively use any free digital GPIO pin for the reset pin.
- Connect display SCL to BeagleBone Black I2C2_SCL pin P9_19 (purple wire).
- Connect display SDA to BeagleBone Black I2C2_SDA pin P9_20 (orange wire).

Note that the BeagleBone Black has two I2C interfaces and this wiring will use the /dev/i2c-1 interface. Make sure there aren't any device tree overlays loaded () which use these I2C pins for other purposes. The default BeagleBone Black device tree configuration with no overlays loaded will expose the necessary I2C interface for the wiring above.

SPI

To use the BeagleBone Black with an SPI display, wire it up as follows:



- Connect display ground to BeagleBone Black ground (black wire).
- Connect display VIN to BeagleBone Black 3.3 volt power (red wire).
- Connect display CS to BeagleBone Black SPI0_CS0 pin P9_17 (yellow wire).
- Connect display RST to BeagleBone Black P9_12 (blue wire). You can alternatively use any free digital GPIO pin for the reset pin.
- Connect display DC to BeagleBone Black P9_15 (cyan wire). You can alternatively use any free digital GPIO pin for the DC pin.
- Connect display CLK to BeagleBone Black SPI0_SCLK pin P9_22 (orange wire).
- Connect display Data to BeagleBone Black SPI0_D1 pin P9_18 (purple wire).

Like with the Raspberry Pi, the wiring above assumes using a hardware SPI interface on the BeagleBone Black, specifically /dev/spidev1.0. Before you can use this SPI interface you must enable a device tree overlay () to turn on the SPI pin functionality. The easiest way to enable this device tree overlay is to configure the BeagleBone Black to load the overlay automatically on boot.

With the BeagleBone Black connected to your computer over USB, open the USB mass storage drive named 'boot' and edit the file uEnv.txt on the drive. Add the following line to the file:

optargs=capemgr.enable_partno=BB-SPIDEV0

NOTE: Be careful editing the uEnv.txt file on Windows, as changing the line endings can cause your BeagleBone Black not to boot and require an OS reinstall! The safest option is to connect to the BeagleBone Black in a command window and follow the steps at the end of this page to mount and edit uEnv.txt on the BeagleBone Black ().

Reboot your device and you should see the files /dev/spidev1.0 and /dev/spidev1.1 no w exist.

Usage

This code is discontinued - Check out our newer tutorial at - https://learn.adafruit.com/monochrome-oled-breakouts/python-wiring

Dependencies

Before using the library you will need to make sure you have a few dependencies installed. Connect to your device using SSH and follow the steps below.

If you're using a Raspberry Pi, install the RPi.GPIO library by executing:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-pip
sudo pip install RPi.GPIO
```

If you're using a BeagleBone Black, install the Adafruit_BBIO library by executing:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-pip
sudo pip install Adafruit_BBIO
```

Finally, on both the Raspberry Pi and Beaglebone Black install the Python Imaging Library () and smbus library by executing:

```
sudo apt-get install python-imaging python-smbus
```

Now to download and install the SSD1306 python library code and examples, execute the following commands:

sudo apt-get install git
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git ()
cd Adafruit_Python_SSD1306
sudo python setup.py install

Usage

Inside the examples subdirectory you'll find python scripts which demonstrate the usage of the library. To help you get started, I'll walk through the shapes.py code below:

```
import time

import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306

import Image
import ImageDraw
import ImageFont
```

First a few modules are imported, including the Adafruit_SSD1306 module which contains the OLED display driver classes. You can also see some of the Python Imaging Library modules like Image, ImageDraw, and ImageFont being imported.

```
# Raspberry Pi pin configuration:
RST = 24
# Note the following are only used with SPI:
DC = 23
SPI_PORT = 0
SPI_DEVICE = 0

# Beaglebone Black pin configuration:
# RST = 'P9_12'
# Note the following are only used with SPI:
# DC = 'P9_15'
# SPI_PORT = 1
# SPI_DEVICE = 0
```

Next some configuration values are set depending on the platform. If you're using the BeagleBone Black you'll need to comment the Raspberry Pi pin configuration lines and uncomment the BeagleBone Black configuration lines.

If you want to change to a different RST or DC pin, you can update the RST and DC pin values respectively. You can also change the SPI port and device, but I recommend sticking with those interfaces above as they've been tested and are known to work.

```
# 128x32 display with hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)

# 128x64 display with hardware I2C:
```

```
# disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)

# Alternatively you can specify an explicit I2C bus number, for example
# with the 128x32 display you would use:
# disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST, i2c_bus=2)

# 128x32 display with hardware SPI:
# disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST, dc=DC, spi=SPI.SpiDev(SPI_PORT,
SPI_DEVICE, max_speed_hz=8000000))

# 128x64 display with hardware SPI:
# disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST, dc=DC, spi=SPI.SpiDev(SPI_PORT,
SPI_DEVICE, max_speed_hz=8000000))

# Alternatively you can specify a software SPI implementation by providing
# digital GPIO pin numbers for all the required display pins.  For example
# on a Raspberry Pi with the 128x32 display you might use:
# disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST, dc=DC, sclk=18, din=25, cs=22)
```

Below the configuration values is the display class setup. There are two classes you can create, SSD1306_128_32 or SSD1306_128_64. The SSD1306_128_32 class represents a 128x32 pixel display, and the SSD1306_128_64 class represents a 128x64 pixel display.

Along with the size of the display you also configure what interface the display uses in these lines. The first couple examples use the I2C interface and only need to specify an RST pin. Internally the SSD1306 library will look up the default I2C bus number for the platform and use it--if you've followed the wiring in this guide you should be all set! However if you need to explicitly control the I2C bus number, the third example shows how to specify it with an i2c_bus parameter.

The last three examples show how to configure the SPI interface. For hardare-based SPI you only need to specify the RST pin, DC pin, and hardware SPI device. Use one of these examples if you've followed the wiring in this guide. However if you want to use a software-based SPI interface, the last example shows how to specify each SPI pin for a Raspberry Pi (for a BeagleBone Black just change the pin values based on the pins you're using).

Uncomment the appropriate display line based on the size of your display and the interface you're using to talk to the display. The code assumes you're using a 128x32 pixel display that's communicating over hardware I2C, however if you're using a different display or protocol uncomment the appropriate line. Make sure to comment all the other lines that aren't being used!

```
# Initialize library.
disp.begin()

# Clear display.
disp.clear()
disp.display()

# Create blank image for drawing.
```

```
# Make sure to create image with mode '1' for 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a black filled box to clear the image.
draw.rectangle((0,0,width,height), outline=0, fill=0)
```

The next bit of code will initialize the display library, clear the display, and configure a PIL drawing class to prepare for drawing graphics. Notice that the image buffer is created in 1-bit mode with the '1' parameter, this is important because the display only supports black and white colors.

```
# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 2
shape_width = 20
top = padding
bottom = height-padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top , x+shape_width, bottom), outline=255, fill=0)
x += shape_width+padding
# Draw a rectangle.
draw.rectangle((x, top, x+shape_width, bottom), outline=255, fill=0)
x += shape_width+padding
# Draw a triangle.
draw.polygon([(x, bottom), (x+shape_width/2, top), (x+shape_width, bottom)],
outline=255, fill=0)
x += shape_width+padding
# Draw an X.
draw.line((x, bottom, x+shape_width, top), fill=255)
draw.line((x, top, x+shape_width, bottom), fill=255)
x += shape_width+padding
```

Once the display is initialized and a drawing object is prepared, you can draw shapes and graphics using PIL's drawing commands (). The code first does a little work to scale drawings based on the display size, then it moves left to right drawing each shape.

```
# Load default font.
font = ImageFont.load_default()

# Alternatively load a TTF font.
# Some other nice fonts to try: http://www.dafont.com/bitmap.php
#font = ImageFont.truetype('Minecraftia.ttf', 8)

# Write two lines of text.
draw.text((x, top),    'Hello',  font=font, fill=255)
draw.text((x, top+20), 'World!', font=font, fill=255)
```

Next the code loads a built-in default font and draws a few lines of text. You can also load your own TrueType font and use it to render text!

```
# Display image.
disp.image(image)
disp.display()
```

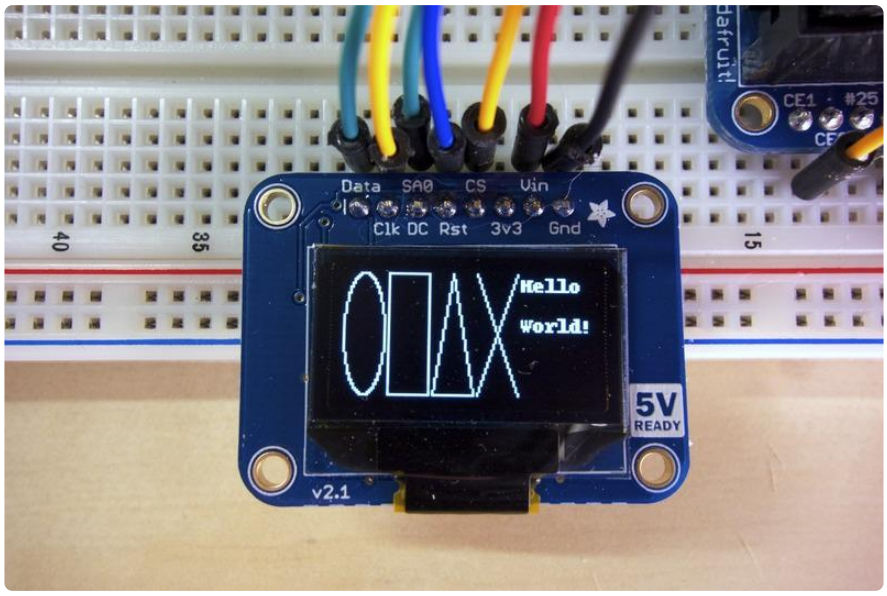
Finally the image that was created is written to the display buffer, and the display is told to show its buffer. Remember, every time you make a change to the image that you want it to be visible on the display you need to call the image and display functions!

That's all there is to the shapes.py code! You can run the code by executing this command in the examples directory:

```
sudo python shapes.py
```

Make sure to run as root with the sudo command so the program has access to the hardware.

If you run the shapes.py example you should see something like this (on a 128x64 display):



Check out the other examples included in the library, such as animate.py which displays an animated text scroller and image.py which displays an image loaded from a file. Modify the configuration at the top of each example just like you did for the sha pes.py example above, then run them just like running shapes.py but substituting the appropriate file name.

Enjoy using your OLED display with a Raspberry Pi or BeagleBone Black! If you have issues or want to contribute to the library, feel free to do so on the library's GitHub home ().