Saving CircuitPython Bitmaps and Screenshots

Created by Dave Astels

https://learn.adafruit.com/saving-bitmap-screenshots-in-circuitpython

Last updated on 2024-06-03 02:51:16 PM EDT

Table of Contents

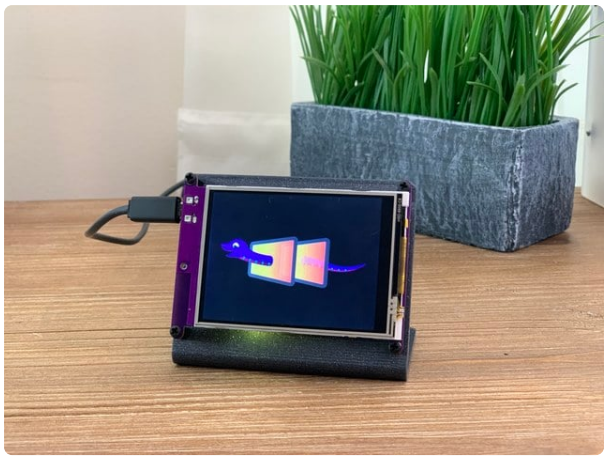
Overview

Working with graphics in CircuitPython is a joy. CircuitPython has support for a plethora of screen types and a rich set of functions to perform graphics operations.

When working on a device that has a screen, it can be useful to programmatically take a snapshot what's on the screen and save it into a file for examination or use later. This could be for various reasons including capturing graphics for use at another time and/or system, debugging the code that is putting pixels on the screen, or use in documentation. It can also be useful to be able to save the contents of a bitmap, for example in PyPaint (https://adafru.it/Frm).

This guide introduces a CircuitPython library which is compatible with CircuitPython 5.0 and later. The library allows the programmer to take the pixels in a bitmap or on a screen and save them into standard 24 bits per pixel BMP files. Files may be on the CircuitPython USB flash filesystem (if it's set to writable mode) or onto a SD card.

Related Products



[Adafruit PyPortal - CircuitPython Powered Internet Display]

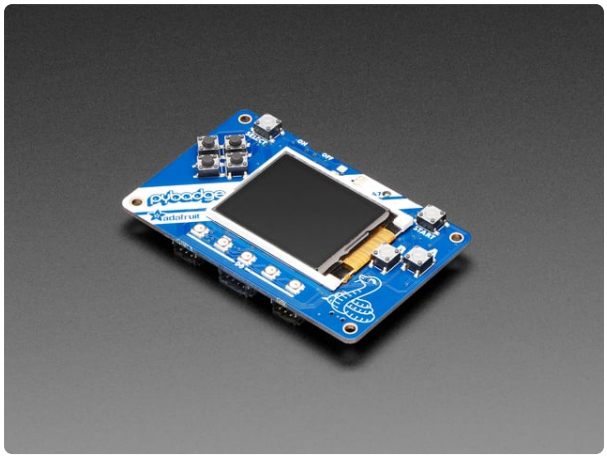
PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...

https://www.adafruit.com/product/4116

[Adafruit PyGamer for MakeCode Arcade, CircuitPython or Arduino]

What fits in your pocket, is fully Open Source, and can run CircuitPython, MakeCode Arcade or Arduino games you write yourself? That's right, it's the Adafruit...

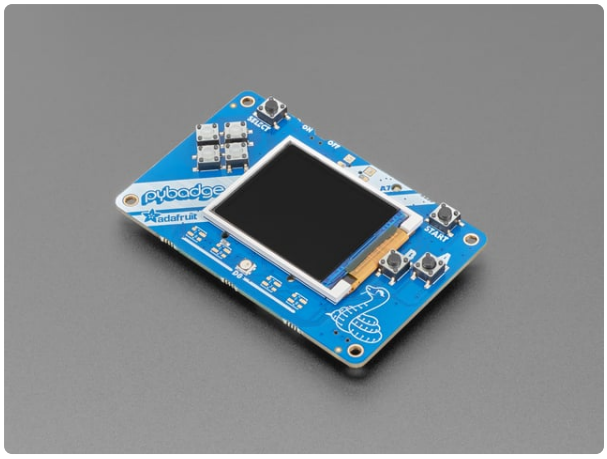
https://www.adafruit.com/product/4242



[Adafruit PyBadge for MakeCode Arcade, CircuitPython, or Arduino]

What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino? That's right, its the Adafruit PyBadge! We wanted to see how much we...

https://www.adafruit.com/product/4200



[Adafruit PyBadge LC - MakeCode Arcade, CircuitPython, or Arduino]
What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino even when you're on a budget? That's right, it's the Adafruit...
https://www.adafruit.com/product/3939

[SD/MicroSD Memory Card (8 GB SDHC)]
Add mega-storage in a jiffy using this 8 GB class 4 micro-SD card. It comes with a SD adapter so you can use it with any of our shields or adapters. Preformatted to FAT so it works out...
https://www.adafruit.com/product/1294

Setup

CircuitPython is a programming language based on Python, one of the fastest growing programming languages in the world. It is specifically designed to simplify

experimenting and learning to code on low-cost microcontroller boards. Here is a guide which covers the basics:

- Welcome to CircuitPython! (https://adafru.it/cpy-welcome)

Be sure you have the latest CircuitPython for your board loaded onto your board, as described here (https://adafru.it/F6N). You will need at least version 5.0 (possibly a beta version of it). While we'll reference the PyPortal in this guide, but this will work on all of the TFT based CircuitPython boards such as PyBadge and PyGamer.

This capability requires CircuitPython 5.0 or higher. It will not work for versions of CircuitPython less than 5.0.

CircuitPython is easiest to use within the Mu Editor. If you haven't previously used Mu, this guide will get you started (https://adafru.it/ANO).

Libraries

Plug your board into your computer via a USB cable. Please be sure the cable is a good power+data cable so the computer can talk to the board.

A new disk should appear in your computer's file explorer/finder called **CIRCUITPY**. This is the place we'll copy the code and code library. If you can only get a drive named **PORTALBOOT** or **PYGAMERBOOT**, load CircuitPython per the guide mentioned above (https://adafru.it/F6N).

Create a new directory on the **CIRCUITPY** drive named **lib**.

Download the latest CircuitPython driver package to your computer using the green button below. **Match the library you get to the version of CircuitPython you are using**. Save to your computer's hard drive where you can find it.

Go to GitHub to get the latest CircuitPython library bundle

https://adafru.it/zB-

To save bitmaps and screenshots you will need to add **adafruit_bitmapsaver.mpy** to the **CIRCUITPY/lib** directory.

This is in addition to whatever libraries are needed for your main code.

If your device has a microSD card slot or adapter and you wish to save to it you will also need **adafruit_sdcard.mpy** (if your board doesn't support **sdcardio**) and **adafruit_bus_device** in **CIRCUITPY/lib**.

Use

Writable USB Flash Filesystem

When CircuitPython starts up, the internal flash filesystem is read only. If you want to save bitmaps or screenshots to it, you must remount it as writable by creating a **boot.py** file with the following in it:

This code goes in the boot.py file. You must press the reset button on your board after saving the boot.py file, or power cycle the device.

```
import storage

storage.remount("/", False)
```

The catch is that it can only be writable by the MCU or USB. The common solution to this is to check an input to see which should be able to write it. Something like:

```
import board
import digitalio
import storage

switch = digitalio.DigitalInOut(board.D0)
switch.direction = digitalio.Direction.INPUT
switch.pull = digitalio.Pull.UP

# If the D0 is connected to ground with a wire
# CircuitPython can write to the drive
storage.remount("/", switch.value)
```

SD Card Support

Setting up the SD card for use is shown below. From then on files can be saved to files in the **/sd** directory.

```
import board
import busio
import storage

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

try:
    import sdcardio

    sdcard = sdcardio.SDCard(spi, board.SD_CS)
except ImportError:
    import adafruit_sdcard
    import digitalio

    cs = digitalio.DigitalInOut(board.SD_CS)
    sd_card = adafruit_sdcard.SDCard(spi, cs)

vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")
```

The following code is for the code.py file.

Taking a Screenshot

This can be as simple as importing the library and calling the `save_pixels()` function with an argument of either a file object or an absolute filename. The example below configures the SD card support and saves a screenshot onto the card.

If a file object is passed, it is written to. If a filename is passed, the named file is opened for writing and written to. In either case, the file is closed once the screenshot has been written.

It can take 20-30 seconds for the screenshot to write, so don't jump the gun and reset in USB mode (or yank the SD card early) in order to check out the screenshot bmp!

```
import board
import busio
import storage
from adafruit_bitmapsaver import save_pixels

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

try:
    import sdcardio
```

```
    sdcard = sdcardio.SDCard(spi, board.SD_CS)
except ImportError:
    import adafruit_sdcard
    import digitalio

    cs = digitalio.DigitalInOut(board.SD_CS)
    sd_card = adafruit_sdcard.SDCard(spi, cs)

vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")

print('Taking Screenshot...')
save_pixels('/sd/screenshot.bmp')
print('Screenshot taken')
```

Handling rotation

A wrinkle with taking screenshots is that some boards use a rotated buffer and some don't. The library takes this into account. On some boards, e.g. the PyPortal, the screenshot will be oriented as expected (shown at the top of this page).

However, on some otherboards, e.g. the PyGamer, it will be rotated 90 degrees. This can be correctly with any image editing software.

Saving a Bitmap

Saving a bitmap is only slightly more involved than saving a screenshot.

A file or filename is passed and used in the same way.

Since this saves the contents of a `Bitmap` object and not the contents of the screen, that `Bitmap` must be passed as well. Furthermore, `Bitmaps` don't contain the actual 24-bit color that ends up on the screen. Instead, they contain indecies into a `Palette`. The result of this is a dramatic memory savings, but at the cost of an additional step between the bitmap and the screen. The practical result of this is that the associated `displayio.Palette` object must be passed along with the bitmap.

Consider the PyPaint project found in this guide (https://adafru.it/Frm). The ability to save the painting can be added with a few lines, essentially:

```
from adafruit_bitmapsaver import save_pixels
.
.
.
save_pixels('/sd/pypaint.bmp', self._fg_bitmap, self._fg_palette)
```

Because saving a PyPortal sized bitmap blocks other code from running and takes a noticeable amount of time, it can be useful to display a message to the user while the saving is taking place.

This results in a BMP file as shown below: