adafruit learning system

Cleveland Museum of Art PyPortal Frame

Created by Dan Cogliano

Last updated on 2020-01-29 08:14:01 PM UTC

Overview

Introduction

Museums today are expanding public access to their art collections, not just inside the walls of the museum but also outside. Digital initiatives are bringing artwork, once relegated within the confines of the museum, to a 21st century global audience. These modern museums have essentially become the new content providers. Much like the modern companies Netflix and Pandora that provide video and music content, museums are becoming their own content providers with their collection of paintings, photographs, jewelry and other media. Digitizing these collections and making it publicly available brings this material to a global audience.

One such museum that is at the forefront of this endeavor is the Cleveland Museum of Art.

The Cleveland Museum of Art

The Cleveland Museum of Art (CMA) is located in Cleveland, Ohio and houses over 61,000 works of art from around the world. Founded in 1913, it is internationally renowned for its holdings of Asian and Egyptian art. More than 770,000 visited the museum in 2018.

On January 23, 2019, the CMA announced over 31,000 digital images of artwork from its collection will be available in the public domain. There are dozens of art types included in the collection, including paintings, drawings, photographs, coins and jewelry. All of these images are available under the Creative Commons Zero (CC0) license (), which allows people to share, collaborate, remix, and reuse images from the collection for both commercial and non-commercial use. The CMA also created a free API () that allows programs to easily retrieve information about each of these images.

Using the Art PyPortal Frame

This project accesses the CMA API using CircuitPython running on the PyPortal. It uses the API to pick a random item from the collection. It converts and resizes the JPEG image from the collection to a BMP image sized to the PyPortal using Adafruit IO image conversion service. Finally, the converted image is downloaded to the PyPortal for display. A new feature of the PyPortal allows the correct scaling of portrait images, which is a great feature for this project.

When powering up the device, the PyPortal displays a Cleveland Museum of Art banner while it connects to the internet and retrieves its first image. The PyPortal retrieves an image from one of the 31,000+ images available in the collection and displays it along with its title at the bottom of the screen.

By default the display updates automatically every 5 minutes (this can be changed in the code), which will take over 3 months 24x7 to display that many images. You can also manually update the display by touching anywhere on the screen using the PyPortal's touch screen. When the screen is touched, a red circle will appear in the bottom right corner of the display, signaling the screen is currently being updated. The touch screen will be inactive while a screen update is being done, either by an automatic or manual update. Note the Adafruit IO image converter service is rate limited so a one minute interval between images is the minimum time.

Which PyPortal to Choose From?

This project is compatible with all PyPortals. The PyPortal comes in three sizes:

- Pynt PyPortal (): 2.4" display, 320x240 (76,800 pixels), micro USB
- PyPortal (): 3.2" display, 320x240 (76,800 pixels), micro USB
- PyPortal Titano (): 3.5" display, 480x320 (153,600 pixels), USB C

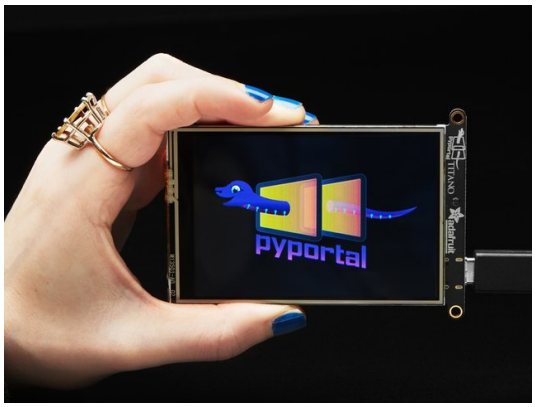
The PyPortal Titano has a larger screen and higher resolution, making it an ideal display for artwork. The Titano actually has twice as many pixels as the standard and Pynt PyPortals, and this project will automatically use those extra pixels for a higher resolution display.

Note that the Titano uses the newer, reversible USB C cable, whereas the Standard and Pynt PyPortals use a micro USB cable.

This project is compatible with all PyPortals, including the Titano with its larger screen and higher resolution.

Parts

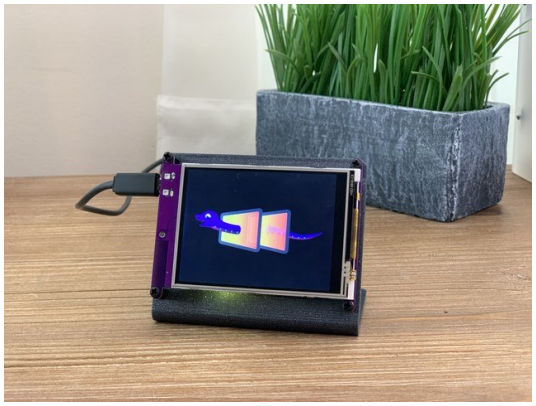
This is an easy project requires no soldering. It will work with any of the PyPortals, including the Titano, which provides more pixels for displaying the work of art. In addition to the PyPortal, this project also requires a MicroSD card, which is used for temporary storage to download the converted image prior to it being displayed.



Adafruit PyPortal Titano

$59.95
IN STOCK

Add To Cart

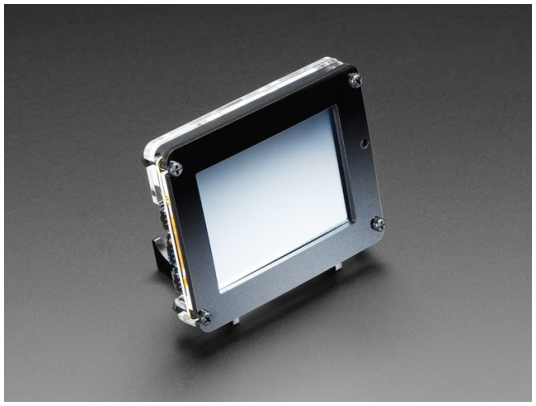


Adafruit PyPortal - CircuitPython Powered Internet Display

$54.95
IN STOCK

Add To Cart

This stand is only compatible with the standard PyPortal.



Adafruit PyPortal Desktop Stand Enclosure Kit

$9.95
IN STOCK

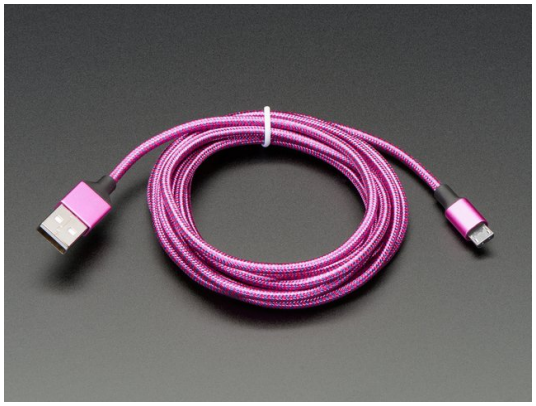
Add To Cart

Pick from one of these USB cables for the standard PyPortal or Pynt PyPortal.

USB A/Micro Cable - 2m

$4.95
IN STOCK

Add To Cart

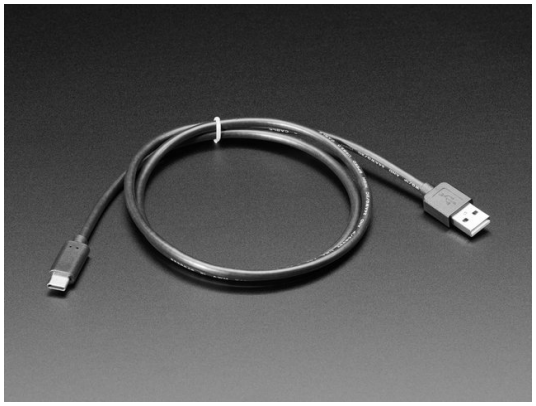


Pink and Purple Braided USB A to Micro B Cable - 2 meter long

$3.95
IN STOCK

Add To Cart

Use this USB type C cable for the Titano.



USB Type A to Type C Cable - approx 1 meter / 3 ft long

$4.95
IN STOCK

Add To Cart

Code

Update CircuitPython

First make sure you are running the latest version of Adafruit CircuitPython for your board.

See the guide page Install CircuitPython () in the Adafruit Learning System guide Adafruit PyPortal () for more information.

Library Files

Next you'll need to install the latest libraries to use the hardware.

Please follow the guide page CircuitPython Libraries () in the Adafruit PyPortal () guide.

Before continuing, please make sure to copy at least these files and folders into your board's **lib** folder.

- **adafruit_bitmap_font**
- **adafruit_bus_device**
- **adafruit_display_shapes**
- **adafruit_display_text**
- **adafruit_esp32spi**
- **adafruit_io**
- **adafruit_logging.mpy**
- **adafruit_pyportal.mpy**
- **adafruit_requests.mpy**
- **adafruit_sdcard.mpy**
- **adafruit_touchscreen.mpy**
- **neopixel.mpy**

Grab the Project Files

Once you have your CircuitPython libraries installed, you can grab the rest of the project files. Use the download "Project Zip" link in the code window below to download the code, fonts, background bitmap and a starter **secrets.py** file.

Using your computer, unzip the files and copy the contents of the zip file to your PyPortal **CIRCUITPY** drive.

```
import time
import random
import board
from adafruit_pyportal import PyPortal
from adafruit_display_shapes.circle import Circle

WIDTH = board.DISPLAY.width
HEIGHT = board.DISPLAY.height

#pylint: disable=line-too-long

# these lines show the entire collection
APIURL = "https://openaccess-api.clevelandart.org/api/artworks?
cc0=1&has_image=1&indent=2&limit=1&skip="
IMAGECOUNT = 31954
```

```
# uncomment these lines to show just paintings
# APIURL = "https://openaccess-api.clevelandart.org/api/artworks?
cc0=1&has_image=1&indent=2&limit=1&type=Painting&skip="
# IMAGECOUNT = 3223

BACKGROUND_FILE = "/background.bmp"
if WIDTH > 320:
    BACKGROUND_FILE = "/background_480.bmp"

pyportal = PyPortal(default_bg=BACKGROUND_FILE,
                    image_json_path=["data", 0, "images", "web", "url"],
                    image_dim_json_path=(["data", 0, "images", "web", "width"],
                                         ["data", 0, "images", "web", "height"]),
                    image_resize=(WIDTH, HEIGHT - 15),
                    image_position=(0, 0),
                    text_font="/fonts/OpenSans-9.bdf",
                    json_path=["data", 0, "title"],
                    text_position=(4, HEIGHT - 9),
                    text_color=0xFFFFFF)

circle = Circle(WIDTH - 8, HEIGHT - 7, 5, fill=0)
pyportal.splash.append(circle)
loopcount = 0
errorcount = 0
while True:
    response = None
    try:
        circle.fill = 0xFF0000
        itemid = random.randint(1, IMAGECOUNT)
        # itemid = 20 # portrait mode example
        # itemid = 21 # landscape mode example
        print("retrieving url:", APIURL + str(itemid))
        response = pyportal.fetch(APIURL + str(itemid))
        circle.fill = 0
        print("Response is", response)
        loopcount = loopcount + 1

    except (RuntimeError, KeyError) as e:
        print("An error occured, retrying! -", e)
        print("loop counter:", loopcount)
        assert errorcount < 20, "Too many errors, stopping"
        errorcount = errorcount + 1
        time.sleep(60)
        continue

    errorcount = 0
    stamp = time.monotonic()
    # wait 5 minutes before getting again
    while (time.monotonic() - stamp) < (5*60):
        # or, if they touch the screen, fetch immediately!
        if pyportal.touchscreen.touch_point:
            break
```

Your **CIRCUITPY** drive should now contain at least these files and folders:

- **background_480.bmp**
- **background.bmp**

- code.py
- fonts (folder):
    - OpenSans-9.bdf
- lib (folder):
    - adafruit_bitmap_font (folder)
    - adafruit_bus_device (folder)
    - adafruit_display_shapes (folder)
    - adafruit_display_text (folder)
    - adafruit_esp32spi (folder)
    - adafruit_io (folder)
    - adafruit_logging.mpy
    - adafruit_pyportal.mpy
    - adafruit_requests.mpy
    - adafruit_sdcard.mpy
    - adafruit_touchscreen.mpy
    - neopixel.mpy
- secrets.py
- unsafe_boot.py

Secrets File Setup

Next is to get the PyPortal connected to the internet. To do this, we need to create a *secrets* file. If you have not yet set up a **secrets.py** file in your **CIRCUITPY** drive and connected to the internet using it, follow this guide and come back when you've successfully connected to the internet ().

Using Mu () or any text editor, you should add your Adafruit IO Username and Adafruit IO Key to the **secrets.py** file. This project connects to Adafruit IO to convert JPEG images to BMP images sized to the PyPortal screen. Adafruit IO is free to use, but you'll need to log in with your Adafruit account to use it. If you don't already have an Adafruit login, create one here ().

Once you have logged into your account, there are two pieces of information you'll need to place in your **secrets.py** file: Adafruit IO username, and Adafruit IO key. Head to io.adafruit.com () and simply click the **View AIO Key** link on the left hand side of the Adafruit IO page to get this information.

Then, update the `aoi_username` and `aio_key` values in the **secrets.py** file. Your **secrets.py** file should look like this:

```
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    'ssid' : 'CHANGE ME',
    'password' : 'CHANGE ME',
    'aio_username' : 'CHANGE ME',
    'aio_key' : 'CHANGE ME',
}
```

Fonts

A font is used to display the title of the artwork on the PyPortal. You will need to create a **fonts** folder on your CircuitPython **CIRCUITPY** drive and download the font file from the project's GitHub repository ().

The Background Image

The background image is used when the art frame initially starts. It is an image of the Cleveland Museum of Art name. The **background.bmp** and background480.bmp files are placed in the main (root) directory of the **CIRCUITPY** drive. The background480.bmp is used with the Titano's larger screen.

The unsafe_boot.py File

This project contains the **unsafe_boot.py** file to switch the CircuitPython filesystem to read-write mode, allowing the artwork to be downloaded locally prior to displaying it. This is a safeguard mechanism, since having two different processors writing to the same filesystem can be a bit risky. It is not a problem here unless you heavily customize this project, which you may want to backup the files. Once the project is installed onto the PyPortal, rename this file to **boot.py** and reboot the PyPortal. Your PyPortal art frame is now ready to display 31,000+ images from the Cleveland Museum of Art.

```
import time
import storage

print("***************** WARNING ******************")
print("Using the filesystem as a write-able cache!")
print("This is risky behavior, backup your files!")
print("***************** WARNING ******************")

storage.remount("/", disable_concurrent_write_protection=True)
time.sleep(5)
```

Resources

Here are some useful links about the Cleveland Museum of Art and the API project:

- The Cleveland Museum of Art ()
- The Cleveland Museum of Art Open Access API ()
- Article: The Cleveland Museum of Art Digitized 30,000 Artworks in the Public Domain ()
- A Raspberry Pi Picture Frame Project Using the Open Access API (and inspired this one) ()
  This project was written for the Raspberry Pi by Michael Weinberg and was the inspiration for this project.