# Table of Contents

## Overview
- Supported Boards

## Convert Your GIF
- GIF by Example
  - Step one - Resize GIF
  - Step 2 - Optimize
  - Reduce Frames

## Quickstart
- PyBadge or EdgeBadge using QSPI for storage
- PyGamer using QSPI or SD for storage
- PyPortal using QSPI or SD for storage
- CLUE using QSPI for storage

## Loading GIFs
- Getting GIFs onto the Filesystem
- Format QSPI with CircuitPython
- Loading GIFs onto QSPI / SD over USB

## Arcada Libraries
- Install Libraries
- Adafruit Arcada
- If you aren’t running Arduino IDE 1.8.10 or later, you’ll need to install all of the following!
- Adafruit NeoPixel
- Adafruit FreeTouch
- Adafruit Touchscreen
- Adafruit SPIFlash
- Adafruit Zero DMA
- Adafruit GFX
- Adafruit ST7735
- Adafruit ILI9341
- Adafruit LIS3DH
- Adafruit Sensor
- Adafruit ImageReader
- ArduinoJson
- Adafruit ZeroTimer
- Adafruit TinyUSB
- Adafruit WavePlayer
- SdFat (Adafruit Fork)
- Audio - Adafruit Fork

## Build In Arduino
- Compiling the Code
- Download the Animated GIF Demo code
- Configuring SdFat
- Initial Test

## Customization
- Loop Time & Brightness
- Storage Locations
Example GIFs

- Animation Demos
Overview

Animated GIFs are a staple of the Internet - and thanks to the speedy SAMD51 chip and our Arcada display library, we can show animated GIFs right from the onboard filesystem for keeping your favorite memes with you all the time even when the Internet is down!

In this project we'll show you how to use some nifty Arduino C++ code that can decode GIF files, and display them on a TFT display. Behind the scenes we use DMA for fast display updates so you don't see any flickering.

The SAMD51 microcontroller we use is pretty fast and does the best it can to decode GIFs, but at full-size, complex GIFs may not run at full speed!

We recommend sticking to GIFs that don't have high frame rates, lots of colors, or need to run at full-speed to look good. Plenty of GIFs will look just fine, just don't expect highest quality / high definition!

Supported Boards

The GIF decoder builds on Adafruit Arcada which supports the SAMD51 chipset and color GFX displays. At this time we recommend the following boards which 'just work' and also have touch/joystick controls:

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 Starter Kit
Please note: you may get a royal blue or purple case with your starter kit (they're both lovely colors) What fits in your pocket, is fully Open...
https://www.adafruit.com/product/4277

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, it's 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
Convert Your GIF

The good news is that you do not have to do any particularly hairy transformations with your GIFs to get them to display - you don't have to make them into header files or anything. However, we do have to perform some steps to make them appear nicely on the TFT you're using

- Reduce size to no larger than 320 wide, 240 tall for PyPortal, and 160x128 for PyBadge or PyGamer (check your hardware to see the size you want to target)
- Remove transparent pixels (sometimes called de-optimization) because it slows down our decoding
- Reduce colors so we don't have to work as hard to decode
- Reduce frames if it makes sense to

This GIF viewer isn't good for detailed, high-frame-rate GIFs. Keep it simple and you'll be happy with the results!

GIF by Example

Let's show an example GIF conversion so you can see how the process happens.

We will convert this BadgerBadger GIF from Giphy (https://adafru.it/Ekr)

You can download it via this zip link:

badger.zip
https://adafru.it/Eks

Step one - Resize GIF

We'll use ezgif.com (https://adafru.it/Ekt) for these steps, its free and we find it works really well!

Click on Resize
Upload the original GIF and click Upload!

Now it's uploaded,

- Set the width and height to be 320 and 240 (you can also set just one, as long as the other is no-greater than the max size)
- Use ImageMagick + coalesce as the resize method. Don't use Gifsicle!
- For the aspect ratio, you can stretch, center or force-aspect.

If you're not sure which one you want, keep going to resize and look at the preview, if you don't like it, change and resize again!

Click resize to get your preview! Check the width is no more than 320, and the height is no more than 240 pixels
You can use the GIF as is, but we found that you can do some optimizations on it to make smaller and faster-running

**Step 2 - Optimize**

You can make a backup of the smaller GIF now. Then click Optimize

The first and easiest is color reduction and dithering. The fewer colors we have, the less decompression we have to do. Especially with a cartoony GIF like this, we can set the colors to be 32 and its not terribly noticable
The new GIF has banded colors on the grass but overall is pretty similar!

Reduce Frames

Another very effective way to reduce the size and increase speed is to remove duplicate frames. Fewer frames means it's easier to decode!

Try Remove duplicate frames and start with a Fuzz factor of around 25%, you can preview the GIF to see if you can fuzz higher (more savings but maybe it starts to look stuttery)

Whatever you do, don't add transparency because it will make the file smaller but the decoding bigger!

OK here's our new final GIF:
Quickstart

You can get started instantly with these UF2's - doubleclick Reset to put your board into BOOT mode, and drag these UF2 files over!

If you are using SD card storage, make a directory called gifs on the SD card and put your GIFs in there

If you are using QSPI storage, and at some point you had CircuitPython loaded, use a USB cable, click reset to have it show up as a disk drive. Make a directory called gifs on the SD card and put your GIFs in there. If you've never loaded CircuitPython, check the Loading GIFs page up ahead.

For boards that have both SD and QSPI storage, the gif player will default to SD if its found, and if not, fall back to QSPI

PyBadge or EdgeBadge using QSPI for storage

PyGamer using QSPI or SD for storage
Loading GIFs

Getting GIFs onto the Filesystem

Alright so now you want your own favorite animations to show up, right? The fun part is getting those files onto your Arcada board. There's two options

- In the previous step, select using SD card storage UF2 then create a folder called /gifs on an SD card that is FAT formatted, and put your GIFs in there. Pop it into your SD card slot and you're good to go!
- Use the default QSPI flash storage, which you can access via the computer USB port if you use our pre-compiled code or compile with TinyUSB support

Format QSPI with CircuitPython

For QSPI, you need to have it 'formatted' as a CircuitPython filesystem. The easiest way to do that is to load CircuitPython. You only have to do this once, then you can load more gifs or remove them over USB

(If you have an SD Card just format it with an SD card writer/reader on your computer)
Start with your folder of GIFs

With your board running the animated GIF Arduino code, plug it into USB and then double-click to launch the bootloader.

Grab CURRENT.UF2 from the xxxxBOOT drive and drag onto your gifs folder on your computer, making a backup of your Arduino firmware.

Visit https://circuitpython.org/downloads (https://adafru.it/Em8) and download the latest firmware for your board as a UF2
Now drag the circuitpython.UF2 onto xxxxBOOT, which will replace the Arduino code with CircuitPython runtime.

Your gifs folder now has two UF2s - CURRENT.UF2 and circuitpython.UF2 (with the version info)

In a few moments you'll get a CIRCUITPY drive, this may contain circuitpython code, files, etc. This is the 8 MB Flash storage on the board, available as a disk drive

Now's a good time to create a folder called gifs on the CIRCUITPY drive if you have not done so
Drag all the GIFs you want to display into the CIRCUITPY/gifs folder.

Once you're done, you can re-load the GIF code by double-clicking the reset button again, to launch the PORTALBOOT bootloader drive, then dragging the backup you made before, CURRENT.UF2 back on.

The board will automatically reboot into GIF playing mode!

Loading GIFs onto QSPI / SD over USB

Arduino doesn't have the ability to access both the SD and QSPI at the same time (because they both have a File type and are incompatible) so we can't copy files from SD to QSPI easily.

However Arduino can be convinced to show the QSPI or SD as USB mass storage, so you can have it show up as a disk drive.

For QSPI, you need to have it 'formatted' as a CircuitPython filesystem. The easiest way to do that is to load CircuitPython. You only have to do this once, as shown above.

For SD, format using any USB SD card formatter dongle.

You must have the board plugged into USB with a Data/Sync Cable then press the reset button. You should see a disk drive appear. It may be called CIRCUITPY or USB Disk (you can rename it if you like)
You can explore the disk, if it doesn't yet contain a gifs folder, please make one now - then put a bunch of gifs in it!

When you're done, be sure to Eject the disk - it wont disappear but it will flush/save all your gif data so it doesn't get corrupted

**Arcada Libraries**

OK now that you have Arduino IDE set up, drivers installed if necessary and you've practiced uploading code, you can start installing all the Libraries we'll be using to program it.

There's a lot of libraries!
Install Libraries

Open up the library manager...

And install the following libraries:

Adafruit Arcada

This library generalizes the hardware for you so you can read the joystick, draw to the display, read files, etc. without having to worry about the underlying methods.

If you use Arduino 1.8.10 or later, the IDE will automagically install all the libraries you need to run all the Arcada demos when you install Arcada. We strongly recommend using the latest IDE so you don't miss one of the libraries!

If you aren't running Arduino IDE 1.8.10 or later, you'll need to install all of the following!

Adafruit NeoPixel

This will let you light up the status LEDs on the front/back.
Adafruit FreeTouch

This is the open source version of QTouch for SAMD21 boards

Adafruit Touchscreen

Used by Adafruit Arcada for touchscreen input (required even if your Arcada board does not have a touchscreen)

Adafruit SPIFlash

This will let you read/write to the onboard FLASH memory with super-fast QSPI support

Adafruit Zero DMA

This is used by the Graphics Library if you choose to use DMA
Adafruit GFX

This is the graphics library used to draw to the screen

If using an older (pre-1.8.10) Arduino IDE, locate and install Adafruit_BusIO (newer versions do this one automatically).

Adafruit ST7735

The display on the PyBadge/PyGamer & other Arcada boards

Adafruit ILI9341

The display on the PyPortal & other Arcada boards

Adafruit LIS3DH

For reading the accelerometer data, required even if one is not on the board
Adafruit Sensor

Needed by the LIS3DH Library, required even if one is not on the board

Adafruit ImageReader

For reading bitmaps from SPI Flash or SD and displaying

ArduinoJson

We use this library to read and write configuration files

Adafruit ZeroTimer

We use this library to easily set timers and callbacks on the SAMD processors
Adafruit TinyUSB

This lets us do cool stuff with USB like show up as a Keyboard or Disk Drive

---

Adafruit WavePlayer

Helps us play .WAV sound files.

---

SdFat (Adafruit Fork)

The Adafruit fork of the really excellent SD card library that gives a lot more capability than the default SD library

---

Audio - Adafruit Fork

Our fork of the Audio library provides a toolkit for building streaming audio projects.
Build In Arduino

Compiling the Code

If you don't want to use the pre-compiled code, you can build it yourself to load onto your Arcada board. This project requires using the Arduino IDE because it needs every bit of computational energy!

Visit the PyPortal Arduino Setup page to install the Arduino IDE (or update it), install PyPortal support (https://adafru.it/Ekv) or ditto for whatever hardware you're using. Also make sure you installed all the Arduino Libraries (https://adafru.it/EU5) (and you need quite a lot!)

Download the Animated GIF Demo code

Open up the Arduino library manager

Search for the Adafruit Arcada GifDecoder library and install it

Verify you have the latest version of all the libraries on this page (https://adafru.it/EU5) via the Arduino Library Manager. The code will not compile with earlier versions.

We also have a great tutorial on Arduino library installation at: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)
Configuring SdFat

If you plan to use the SD card to play GIFs you'll need to update `SDFatConfig.h` in the SdFat library you installed, and change the following `#define`'s:

```c
//-------------------------------------------------------------------------------
/**
 * If the symbol ENABLE_EXTENDED_TRANSFER_CLASS is nonzero, the class SdFatEX
 * will be defined. If the symbol ENABLE_SOFTWARE_SPI_CLASS is also nonzero,
 * the class SdFatSoftSpiEX will be defined.
 *
 * These classes used extended multi-block SD I/O for better performance.
 * the SPI bus may not be shared with other devices in this mode.
 */
#define ENABLE_EXTENDED_TRANSFER_CLASS 1
//-------------------------------------------------------------------------------
/**
 * If the symbol USE_STANDARD_SPI_LIBRARY is zero, an optimized custom SPI
 * driver is used if it exists.  If the symbol USE_STANDARD_SPI_LIBRARY is
 * one, the standard Arduino SPI.h library is used with SPI. If the symbol
 * USE_STANDARD_SPI_LIBRARY is two, the SPI port can be selected with the
 * constructors SdFat(SPIClass* spiPort) and SdFatEX(SPIClass* spiPort).
 */
#define USE_STANDARD_SPI_LIBRARY 2
```

Initial Test

Let's make sure you can get the basics going before we customize it

Make sure you have the Adafruit PyPortal type of board selected or whatever board you're looking to use, and pick the right COM port. It may or may not have Adafruit PyPortal next to the serial port name. If you can't find it, double-click the reset button to put it into bootloader mode, then select the serial port when it appears.

Under USB Stack select TinyUSB - this will let the board show the filesystem up as a disk drive
In order to really be able to display GIFs at full speed, we need to overclock a bit...Make sure that before you continue, you have the following selected:

- CPU Speed: 180 MHz (you can also try 200)
- Optimize: fast
- Cache: Enabled

Click Upload to upload the sketch to the PyPortal

You may get a notice that there's not configuration file found, and also likely
Customization

There are a few settings you can change

Loop Time & Brightness

First is how long to display a GIF, second is how bright the backlight is. We can display multiple images, by default waiting 10 seconds until continuing to the next one. You can change this by creating or replacing a text file called arcada_config.json in the root directory of the QSPI flash or SD card. If you have the Arcada board plugged in and press reset, it will appear in the filesystem for you to access.

Here's an example of the contents of the file you can use

```
{"volume":255,"brightness":255,"seconds_per_gif":5}
```

`volume` is not used in this project. `brightness` ranges from 0 (display off) to 255 (all the way max brightness). `seconds_per_gif` is what you expect.

And here's where it resides, remember its not in the gifs folder! Place it in the 'root' directory.
When you're done, be sure to Eject the disk - it won't disappear but it will flush/save your config data so it doesn't get corrupted.

Storage Locations

Of course you want to change the built in GIFs! You can store GIFs in one of two places - either an SD card you insert OR on the internal QSPI Flash.

You cannot pick or choose in one compiled version due to some unavoidable constraints on how Arduino handles Files - so you have to pick one!

By default we use QSPI, because almost all our boards have it built in for free.

However, if you'd like to use SD cards for storage, go to the Adafruit_Arcada library and look in Adafruit_Arcada.h for a section starting with #elif defined(ADAFRUIT_PYPORTAL)
Find the lines within that section that look like:

```c
#define ARCADA_USE_QSPI_FS
//#define ARCADA_USE_SD_FS
```

And change it to

```c
//#define ARCADA_USE_QSPI_FS
#define ARCADA_USE_SD_FS
```

Then save and re-upload

Displaying GIFs from SD cards is easier to move big files on and off the device because you can use an SD card reader, but they will display slower than when stored on the QSPI internal memory! That’s because QSPI is a much faster storage system, we can read the data a few seconds faster!

### Example GIFs

#### Animation Demos

Looking for some cool GIFs to get the creative juices flowing? We put together some neat ones we think you'll like. Download the zip file and drop the files into your GIFs folder.

- blinka-wipe.gif – Blinka transition for project videos
- cpython-logo.gif – The circuit python logo
- mu-editor.gif – Our favorite python editor
- pyportal-blinka.gif – PyPortal logo featuring Blinka
And here's a pack with them resized to 160x128 for 1.8" Displays

160x128_GIFs.zip
https://adafruit.it/Etp