# PyGamer MP3 Player with CircuitPython

Created by Jeff Epler

https://learn.adafruit.com/pygamer-mp3-player-with-circuitpython

Last updated on 2024-09-25 01:38:51 PM EDT

# Table of Contents

# Overview


Whether it's the soundtrack for your workout or a mix for your sweetheart, you can turn your PyGamer into a standalone MP3 player with JEplayer.

This guide will take you through the steps of assembling the PyGamer case, installing CircuitPython and JEplayer, adding your music and art, and then using the player itself.

## Installing or upgrading CircuitPython

You should ensure you have CircuitPython 7.3.3 or greater on your board. Plug your board in with a known good data + power cable (not the cheesy USB cable that comes with USB power packs, they are power only). You should see a new flash drive pop up.

If the drive is **CIRCUITPY**, then open the **boot_out.txt** file to ensure the version number is 7.3.3 or greater.

```
Adafruit CircuitPython 7.3.3 on 2022-08-29; Adafruit PyGamer with samd51j19
```

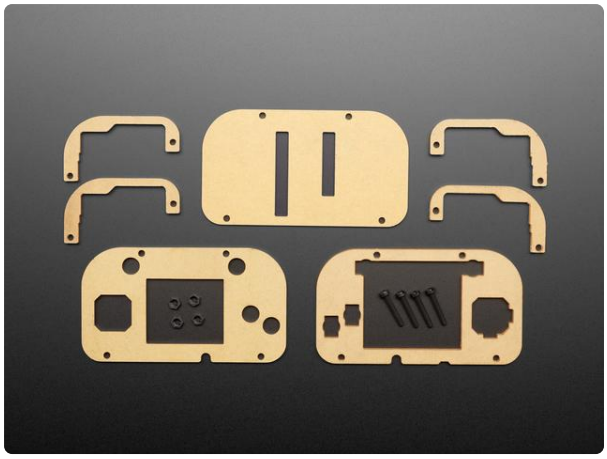
If you need to install or upgrade CircuitPython, see this guide page on the pygamer (https://adafru.it/Led).

The PyGamer board doesn't have a built in speaker. If you buy the kit, it includes a speaker; or use standard 3.5mm headphones for private audio listening.

## Parts

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 PyGamer Acrylic Enclosure Kit
You've got your PyGamer, and you're ready to start jammin' on your favorite arcade games. You gaze adoringly at the charming silkscreen designed by Ada-friend...
https://www.adafruit.com/product/4238


SD/MicroSD Memory Card - 16GB Class 10 - Adapter Included

Add speedy mega-storage in a jiffy using this 16 GB Class 10 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...

https://www.adafruit.com/product/2693

Cell-phone TRRS Headset - Earbud Headphones w/ Microphone

These earbud headphones are the perfect accessory for your FONA - they've been tested to work with our modules - but can be used with any iOS or Android device that uses a TRRS...

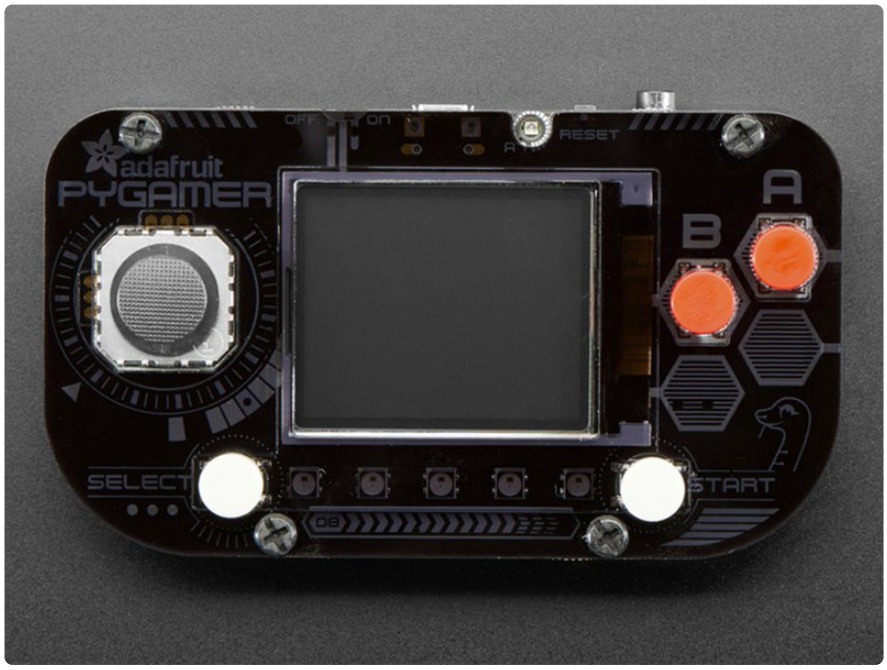
https://www.adafruit.com/product/1966

**1 x** USB MicroSD Card Reader/Writer https://www.adafruit.com/product/939

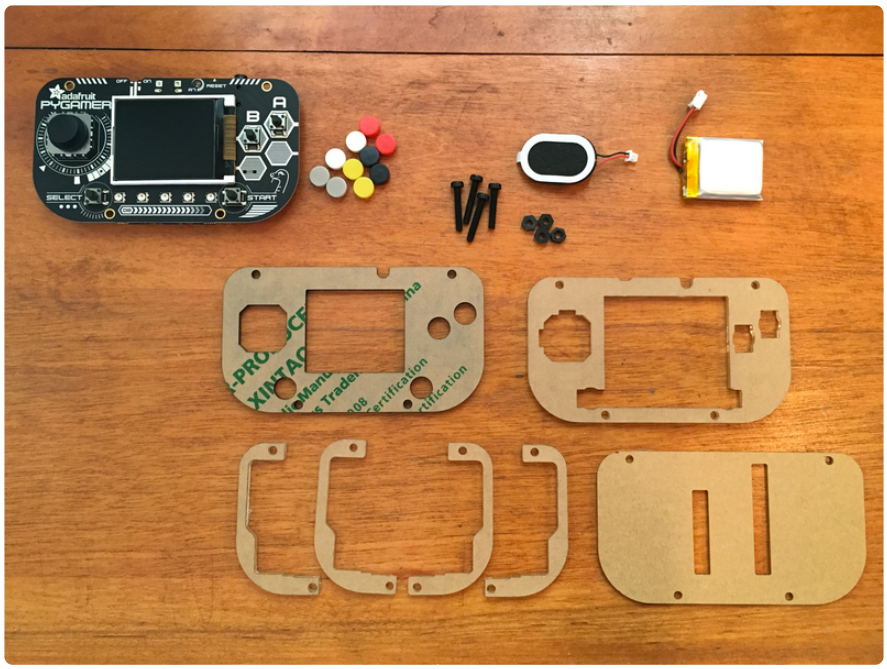
microSD / microSDHC / microSDXC

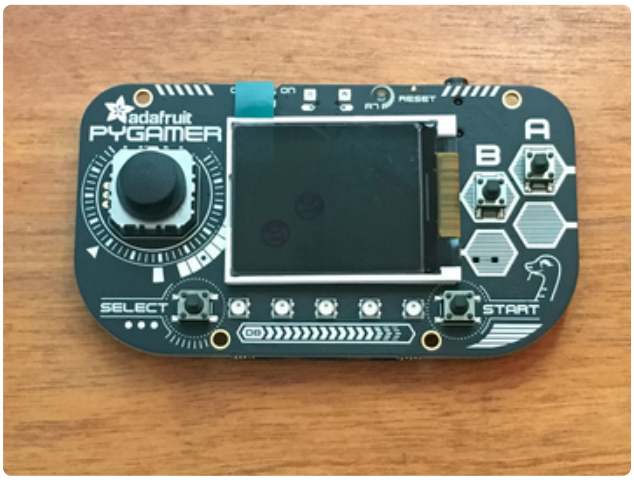
---

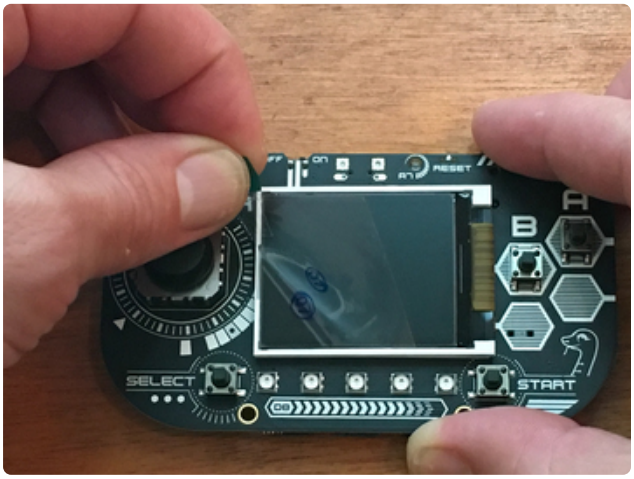
# Build the PyGamer Case



Here's how to assemble the laser cut acrylic case for the PyGamer. The kit comes with seven pieces of acrylic, and four screws and nuts. You've got ten button caps to pick from (you'll pick four), and you'll also want to connect the speaker and battery for the full portable experience.

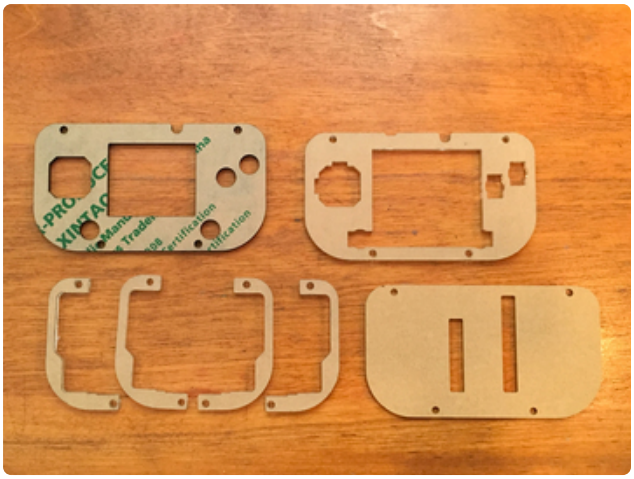


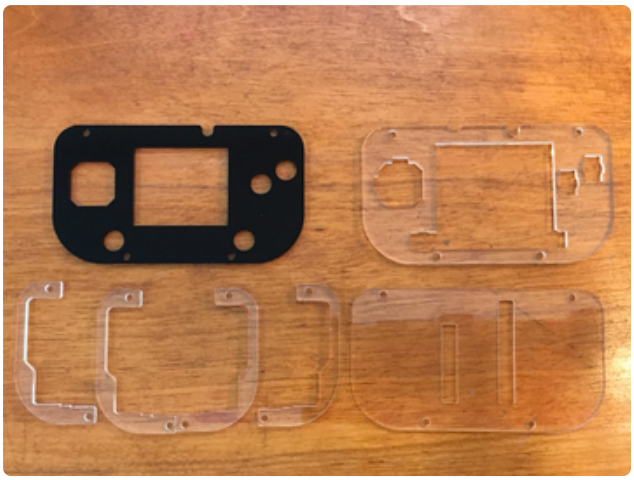



## Prep

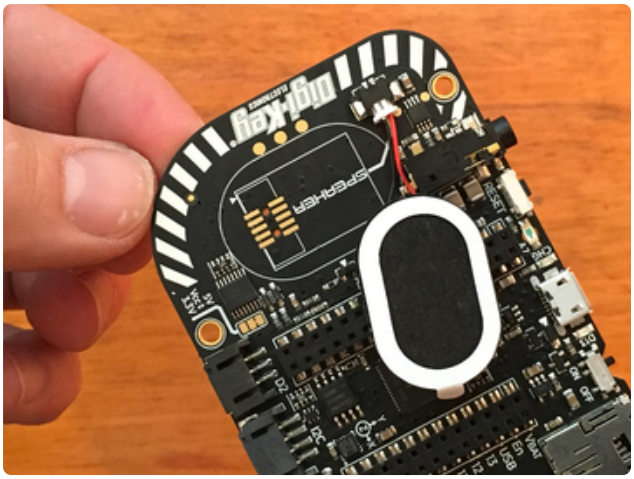
If you haven't already, remove the clear plastic screen protector film from the PyGamer display.

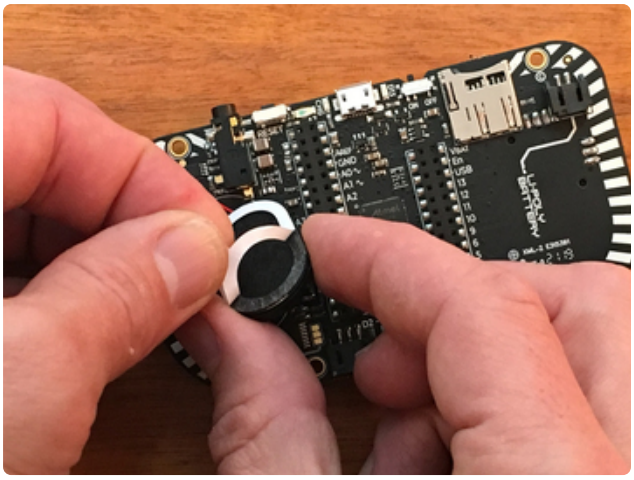


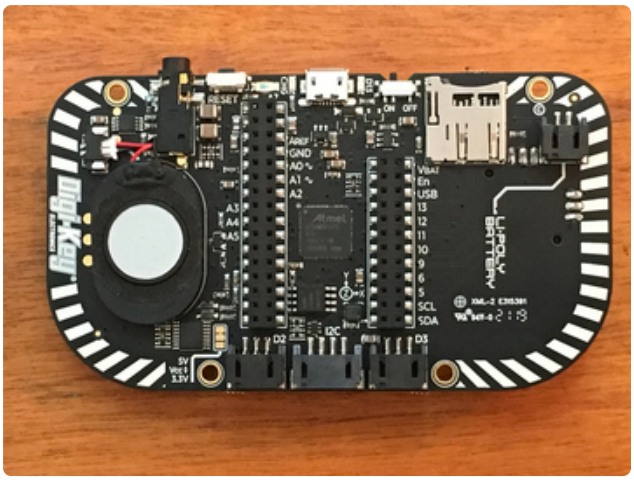


## Paper Protection

Remove the protective paper backing from both sides of all the acrylic pieces.



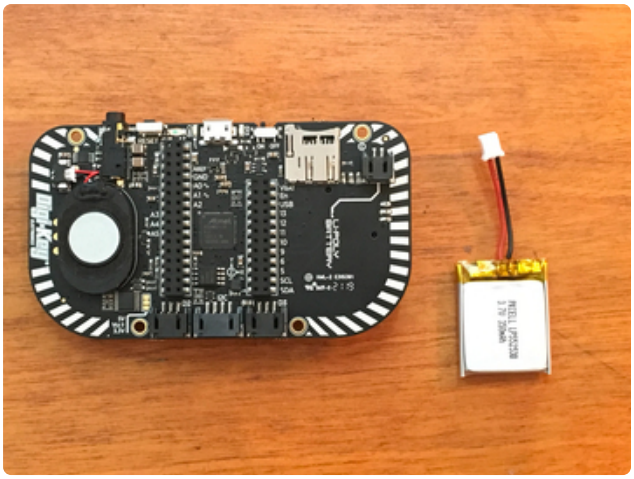


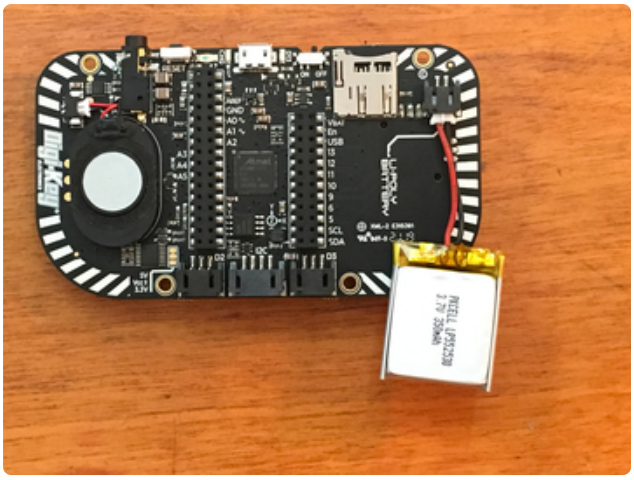


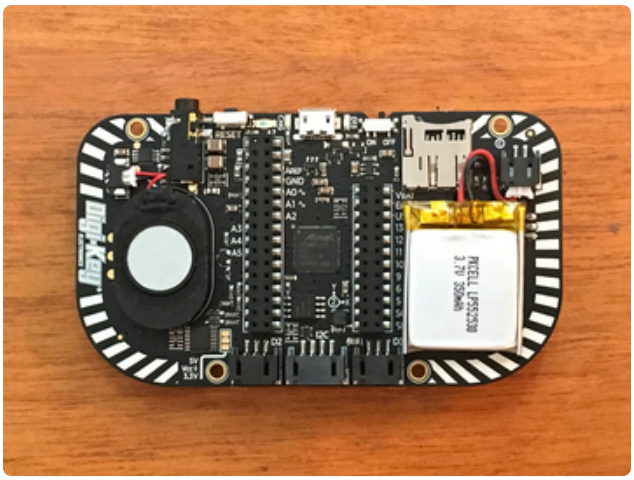
## Speaker

Plug the speaker into the speaker port on the PyGamer.

Then, remove the white oval plastic ring to expose the adhesive and press the speaker to the PyGamer where the silkscreen oval outline indicates.

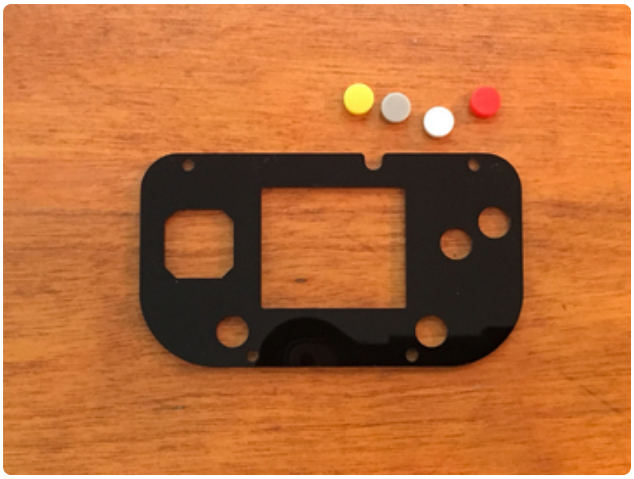


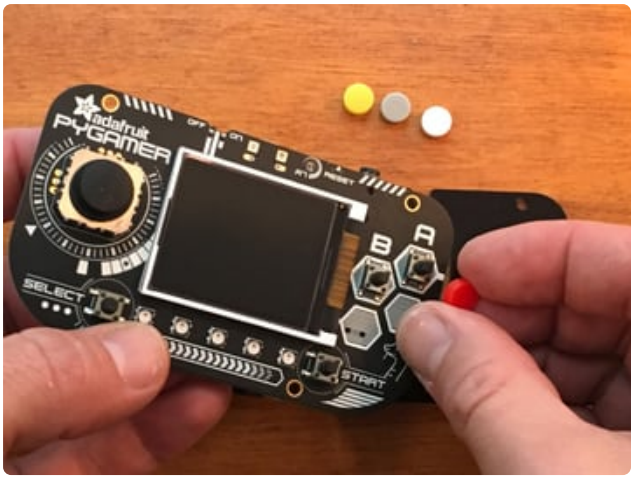


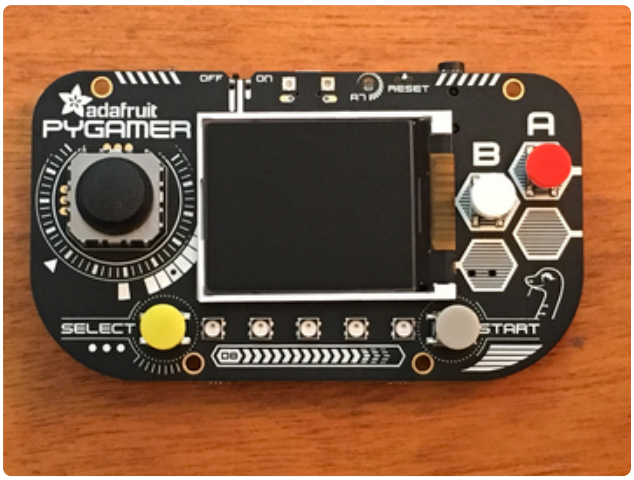


## Battery

Plug the battery into the on-board connector. Very carefully, bend the wires so that the battery fits the spot shown.

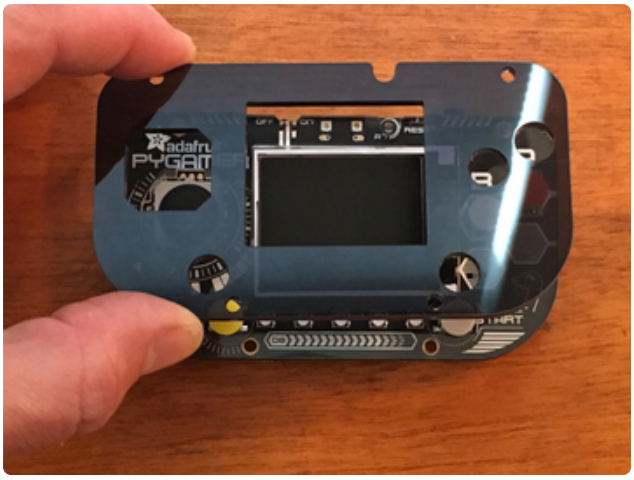





## Button Caps

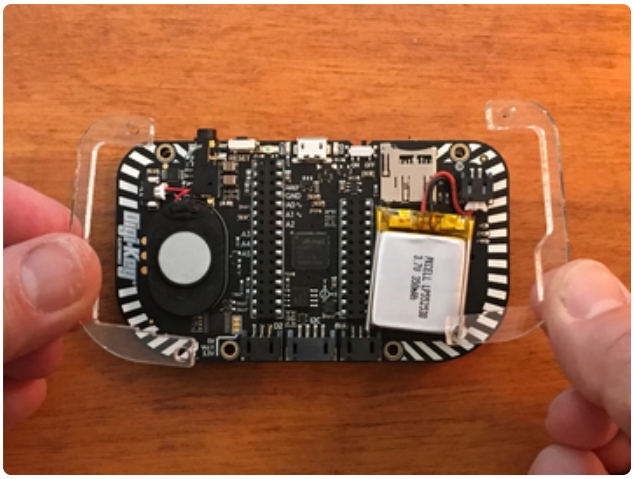
Pick four of the button caps and click them into place on the square shafts of the buttons. Which color combo will you choose?!


## Case Layers

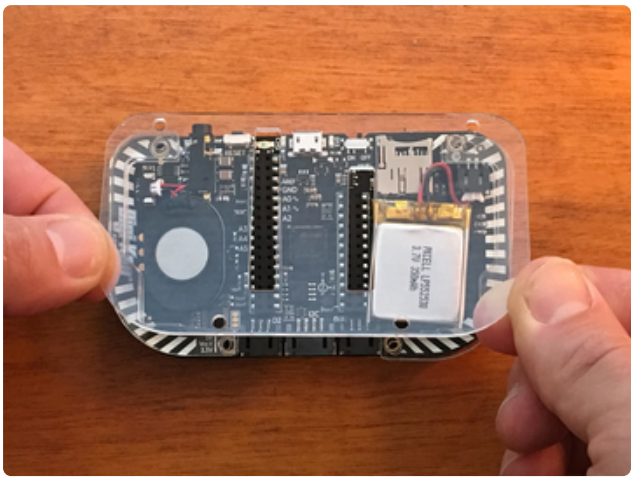
The case assembly is pretty simple. Place the clear top side piece on as shown.

Next, place the smoked gray piece on.



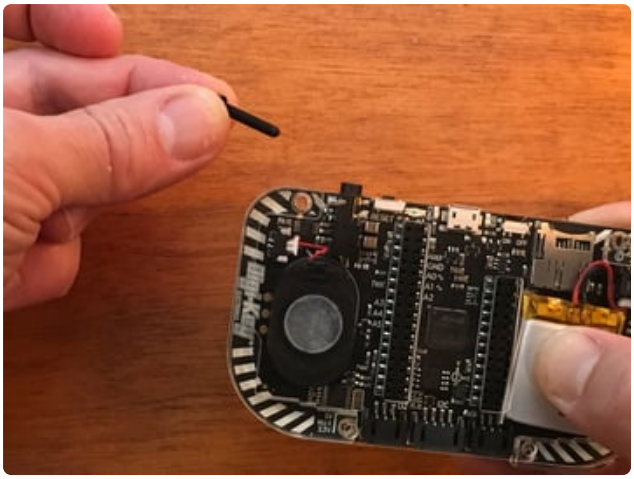
## Spacers

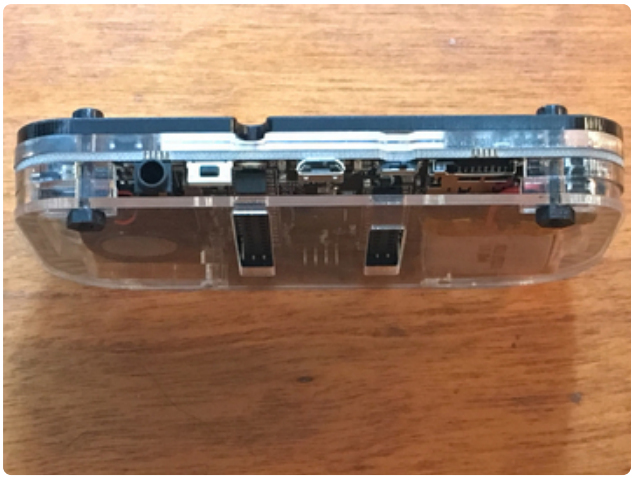
Flip the board over, then place the four spacer pieces onto the back as shown.



## Backing

The last piece to go on is the thin bottom layer with the Feather header cutouts.

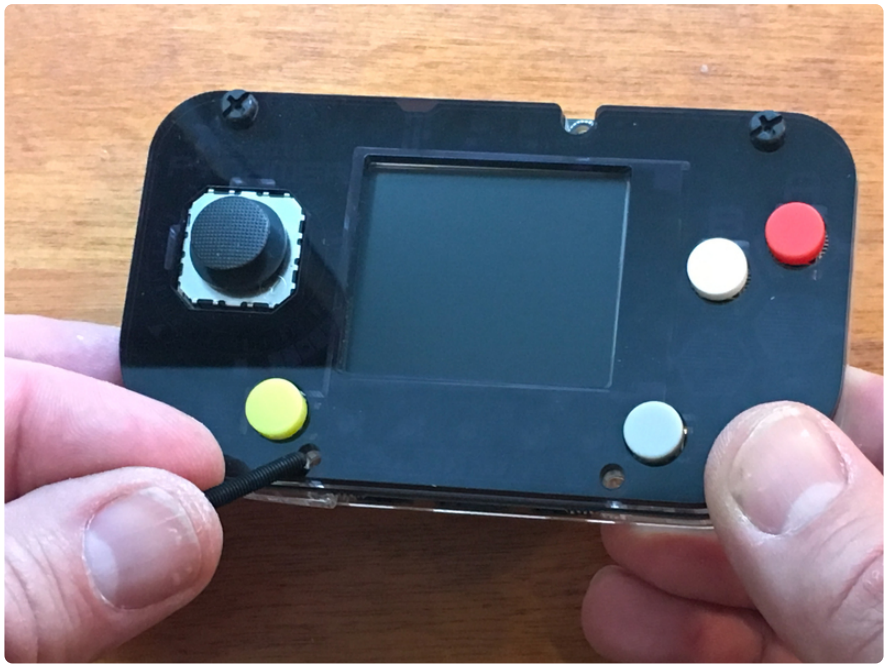


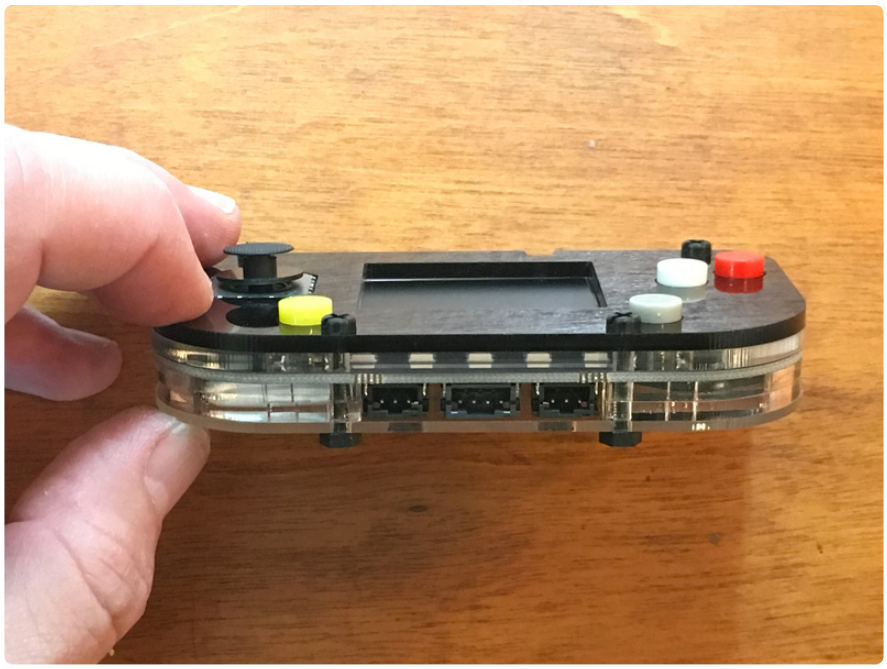


## Fasteners

Push the four screws through from front to back, being sure they go through the holes in all layers and the PyGamer.

Screw on the nuts to secure things. Hand tight is fine -- you don't want to crack anything by using excessive force.





That's all there is to it -- you're ready to play with your PyGamer in its excellent, stylish case!

---

# CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** flash drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

## Set up CircuitPython Quick Start!

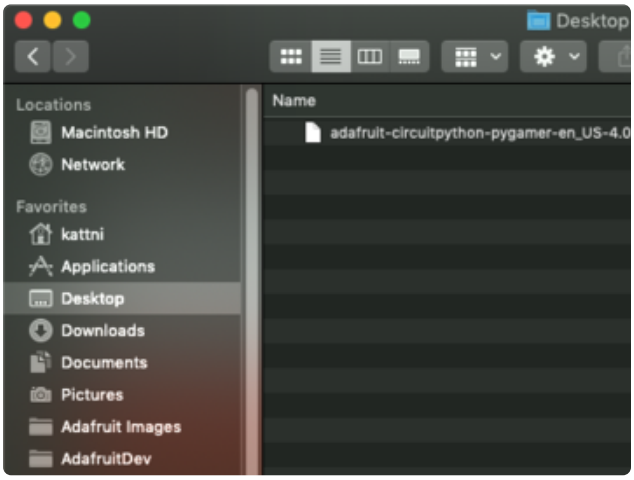
Follow this quick step-by-step for super-fast Python power :)

**Download the latest version of CircuitPython for PyGamer via circuitpython.org**

https://adafru.it/FxM

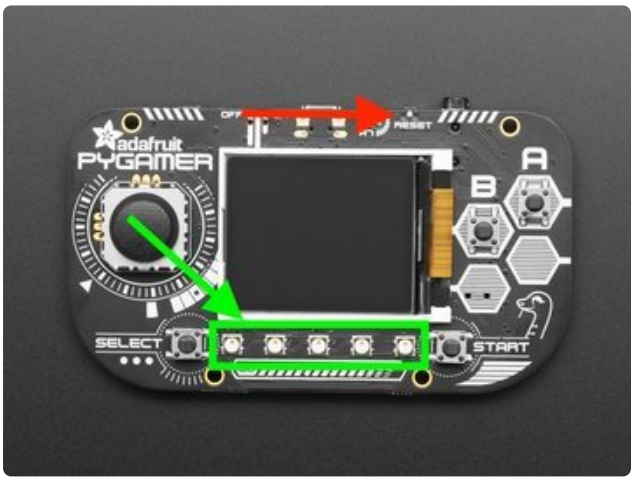
## Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).




**Click the link above and download the latest UF2 file.**

Download and save it to your desktop (or wherever is handy).

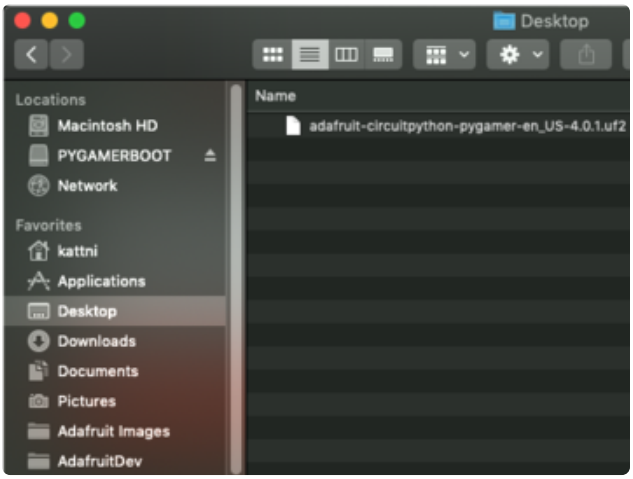



Plug your PyGamer into your computer using a known-good USB cable.

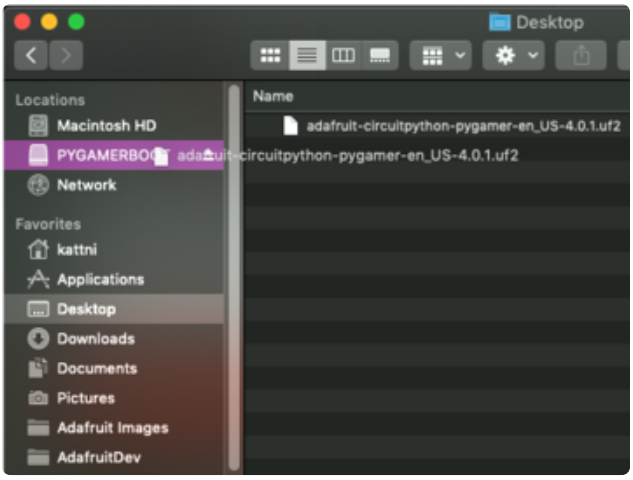
**A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.**

Double-click the **Reset button on the top** of your board (indicated by the red arrow in the first image). You will see an image on the display instructing you to drag a UF2 file to your board, and **the row of NeoPixel RGB LEDs on the front will turn green** (indicated by the green arrow and square in the image). If they turn red, check the USB cable, try another USB port, etc.

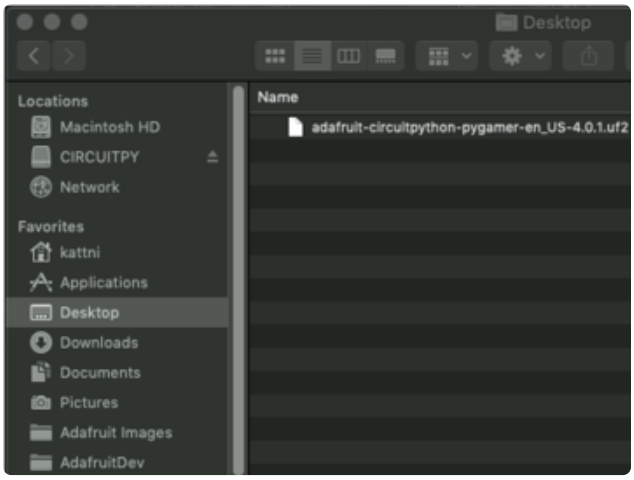
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!




You will see a new disk drive appear called **PYGAMERBOOT**.




Drag the **adafruit_circuitpython_etc.uf2** file to **PYGAMERBOOT.**




The LEDs will flash. Then, the **PYGAMERBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

# Install JEplayer

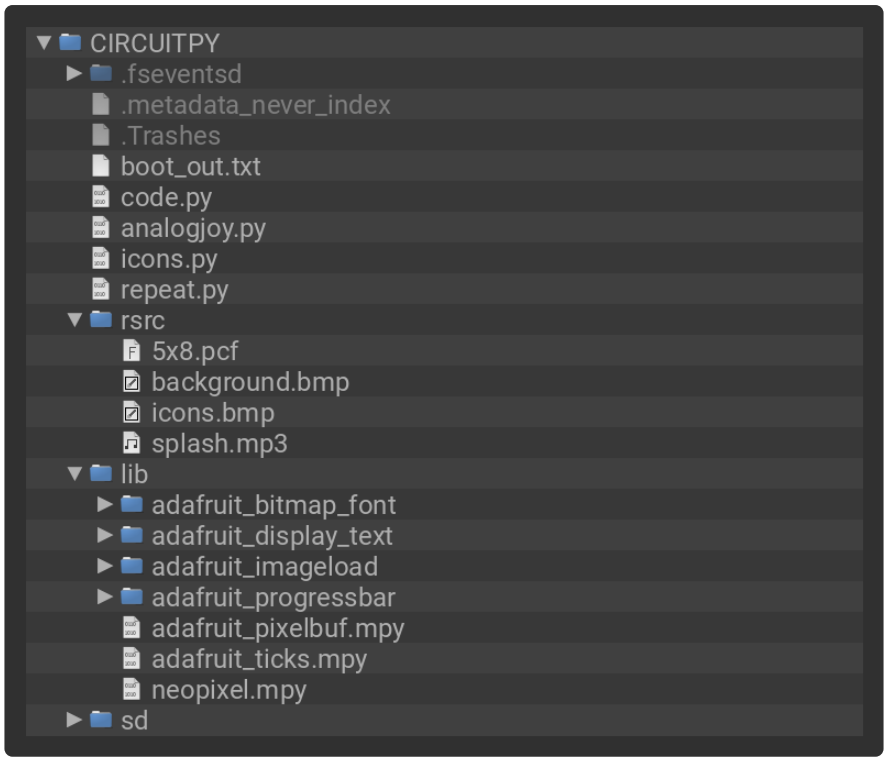
As of CircuitPython 9, a mount point (folder) named /sd is required on the CIRCUITPY drive. Make sure to create that directory after upgrading CircuitPython.

**Follow these steps to create the /sd directory**

https://adafru.it/19ei

## Download JEplayer

Use the **Download Project Bundle link below,** and then drag and drop everything inside the zip (including the folder called **rsrc**) onto the **CIRCUITPY** drive. Once you're done, it should look something like this in your file browser:




The PyGamer will automatically restart and run JEplayer, but until you've loaded a Micro SD card with your tracks, it won't have anything to play.

For more info about working with the CIRCUITPY drive, we have a dedicated guide page. (https://adafru.it/EL3)

```
# SPDX-FileCopyrightText: 2020 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# The MIT License (MIT)
#
# Copyright (c) 2020 Jeff Epler for Adafruit Industries LLC
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
```

```
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
"""
jeplayer - main file

This is an MP3 player for the PyGamer with CircuitPython.

See README.md for more information.
"""

import gc
import os
import random
import time

import adafruit_bitmap_font.bitmap_font
import adafruit_display_text.label
from adafruit_progressbar import ProgressBar
import sdcardio
import analogjoy
import audioio
import audiomp3
import board
import busio
import digitalio
import displayio
import terminalio
from keypad import ShiftRegisterKeys
import icons
import neopixel
import repeat
import storage
from micropython import const

def clear_display():
    """Display nothing"""
    board.DISPLAY.root_group = displayio.Group()

clear_display()

# pylint: disable=invalid-name
def px(x, y):
    """Convert a raw value (x/y) to a pixel value, clamping negative values"""
    return 0 if x <= 0 else round(x / y)
# pylint: enable=invalid-name

(ICON_PLAY, ICON_PAUSE, ICON_STOP, ICON_PREV, ICON_NEXT, ICON_REPEAT,
 ICON_SHUFFLE, ICON_FOLDERNEXT) = range(8)

class PlaybackDisplay:
    """Manage display during playback"""
    def __init__(self):
        self.group = displayio.Group()
        self.glyph_width, self.glyph_height = font.get_bounding_box()[:2]
        self.pbar = ProgressBar(0, 0, board.DISPLAY.width,
                                self.glyph_height*2, bar_color=0x0000ff,
                                outline_color=0x333333, stroke=1)
        self.iconbar = icons.IconBar()
        self.iconbar.group.y = 1000
        for i in range(5, 8):
            self.iconbar.icons[i].x += 32
```

```
        self.label = adafruit_display_text.label.Label(font, line_spacing=1.0)
        self.label.y = 4
        self._bitmap_filename = None
        self._fallback_bitmap = ["/rsrc/background.bmp"]
        self._rms = 0.
        self._text = ""
        self.set_bitmap([]) # Must be first!
        self.group.append(self.pbar)
        self.group.append(self.label)
        self.group.append(self.iconbar.group)
        self.pixels = neopixel.NeoPixel(board.NEOPIXEL, 5)
        self.pixels.auto_write = False
        self.pixels.fill(0)
        self.pixels.show()
        self.paused = False
        self.next_choice = 0
        self.tile_grid = None

    @property
    def text(self):
        """The text shown at the top of the display.  Usually 2 lines."""
        return self._text

    @text.setter
    def text(self, text):
        if len(text) > 256:
            text = text[:256]
        self._text = text
        self.label.text = text

    @property
    def progress(self):
        """The fraction of progress through the current track"""
        return self.pbar.progress

    @progress.setter
    def progress(self, frac):
        self.pbar.progress = frac

    def set_bitmap(self, candidates):
        """Find and use a background from among candidates, or else the fallback
bitmap"""
        for i in candidates + self._fallback_bitmap:
            if i == self._bitmap_filename:
                return # Already loaded

            # CircuitPython 6 & 7 compatible
            try:
                bitmap_file = open(i, 'rb')
            except OSError:
                continue
            bitmap = displayio.OnDiskBitmap(bitmap_file)
            self._bitmap_filename = i
            # Create a TileGrid to hold the bitmap
            self.tile_grid = displayio.TileGrid(
                bitmap,
                pixel_shader=getattr(
                    bitmap, "pixel_shader", displayio.ColorConverter()
                ),
            )

            # # CircuitPython 7+ compatible
            # try:
            #     bitmap = displayio.OnDiskBitmap(i)
            # except OSError:
            #     continue
            # self._bitmap_filename = i
            # # Create a TileGrid to hold the bitmap
            # self.tile_grid = displayio.TileGrid(
```

```
            #     bitmap, pixel_shader=bitmap.pixel_shader
            # )

            # Add the TileGrid to the Group
            if len(self.group) == 0:
                self.group.append(self.tile_grid)
            else:
                self.group[0] = self.tile_grid
            self.tile_grid.x = (160 - bitmap.width) // 2
            self.tile_grid.y = self.glyph_height*2 + max(0, (96 - bitmap.height) //
2)
            break

    @property
    def rms(self):
        """The RMS audio level, used to control the neopixel vu meter"""
        return self._rms

    @rms.setter
    def rms(self, value):
        self._rms = value
        self.pixels[0] = (20, 0, 0) if value > 20 else (px(value, 1), 0, 0)
        self.pixels[1] = (20, 0, 0) if value > 40 else (px(value - 20, 1), 0, 0)
        self.pixels[2] = (20, 0, 0) if value > 80 else (px(value - 40, 2), 0, 0)
        self.pixels[3] = (20, 0, 0) if value > 160 else (px(value - 80, 4), 0, 0)
        self.pixels[4] = (20, 0, 0) if value > 320 else (px(value - 160, 8), 0, 0)
        self.pixels.show()

    # pylint: disable=too-many-branches
    def press(self, idx):
        """Do the action for the current icon"""
        selected = self.iconbar.selected
        if selected in (ICON_PLAY, ICON_PAUSE):  # Play/Pause
            if self.paused:
                self.resume()
            else:
                self.pause()
            self.iconbar.select(not self.paused)
        elif selected == ICON_STOP:
            self.iconbar.deactivate(ICON_FOLDERNEXT)
            return (-1,)
        elif selected == ICON_PREV:
            if self.shuffle:
                return (None,)
            return (idx-1,)
        elif selected == ICON_NEXT:
            if self.shuffle:
                return (None,)
            return (idx+1,)
        elif selected == ICON_SHUFFLE:
            self.iconbar.toggle(selected)
            if self.iconbar.active[ICON_SHUFFLE]:
                self.iconbar.deactivate(ICON_REPEAT)
                self.iconbar.deactivate(ICON_FOLDERNEXT)
        elif selected == ICON_REPEAT:
            self.iconbar.toggle(selected)
            if self.iconbar.active[ICON_REPEAT]:
                self.iconbar.deactivate(ICON_SHUFFLE)
                self.iconbar.deactivate(ICON_FOLDERNEXT)
        elif selected == ICON_FOLDERNEXT:
            self.iconbar.toggle(selected)
            if self.iconbar.active[ICON_FOLDERNEXT]:
                self.iconbar.deactivate(ICON_REPEAT)
                self.iconbar.deactivate(ICON_SHUFFLE)
        return None

    def move(self, direction):
        """Switch the current icon in the given direction"""
        self.iconbar.select((self.iconbar.selected + direction) % 8)
```

```
    def play(self, stream):
        """Starting playing a stream on the speaker"""
        speaker.play(stream)
        self.paused = False
        self.iconbar.set_active(0, not self.paused)
        self.iconbar.set_active(1, self.paused)

    def pause(self):
        """Pause the stream"""
        speaker.pause()
        self.paused = True
        self.iconbar.set_active(0, not self.paused)
        self.iconbar.set_active(1, self.paused)

    def resume(self):
        """Resume the stream"""
        speaker.resume()
        self.paused = False
        self.iconbar.set_active(0, not self.paused)
        self.iconbar.set_active(1, self.paused)

    @property
    def shuffle(self):
        """Whether to shuffle the playlist"""
        return self.iconbar.active[ICON_SHUFFLE]
    @property
    def repeat(self):
        """Whether to repeat the playlist"""
        return self.iconbar.active[ICON_REPEAT]
    @property
    def auto_next(self):
        """Whether to play all folders"""
        return self.iconbar.active[ICON_FOLDERNEXT]

    @staticmethod
    def has_any_mp3s(folder):
        """True if the folder contains at least one item ending in .mp3"""
        return any(not fn.startswith(".") and fn.lower().endswith(".mp3")
                for fn in os.listdir(folder))

    def choose_folder(self, base='/sd'):
        """Let the user choose a folder within a base directory"""
        all_folders = (m for m in os.listdir(base)
                       if not m.startswith('.') and isdir(join(base, m)))
        all_folders = sorted(f for f in all_folders if self.has_any_mp3s(join(base,
f)))
        choices = ['Surprise Me'] + all_folders

        if playback_display.auto_next:
            idx = self.next_choice
        else:
            idx = menu_choice(choices,
                              sel_idx=self.next_choice,
                              text_font=terminalio.FONT)
        clear_display()
        self.next_choice = idx
        if idx >= 1:
            result = all_folders[idx-1]
            self.next_choice = idx+1
            if self.next_choice == len(choices):
                self.next_choice = 1   # Go to first folder, not "surprise me"
        else:
            result = random.choice(all_folders)
        return join(base, result)

# pylint: disable=invalid-name
enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
enable.direction = digitalio.Direction.OUTPUT
```

```
enable.value = True
speaker = audioio.AudioOut(board.SPEAKER, right_channel=board.A1)
mp3stream = audiomp3.MP3Decoder(open("/rsrc/splash.mp3", "rb"))
speaker.play(mp3stream)

font = adafruit_bitmap_font.bitmap_font.load_font("rsrc/5x8.pcf")
playback_display = PlaybackDisplay()
board.DISPLAY.root_group = playback_display.group
font.load_glyphs(range(32, 128))

joystick = analogjoy.AnalogJoystick()

up_key = repeat.KeyRepeat(lambda: joystick.up, rate=0.2)
down_key = repeat.KeyRepeat(lambda: joystick.down, rate=0.2)
left_key = repeat.KeyRepeat(lambda: joystick.left, rate=0.2)
right_key = repeat.KeyRepeat(lambda: joystick.right, rate=0.2)

buttons = ShiftRegisterKeys(clock=board.BUTTON_CLOCK,
                                    data=board.BUTTON_OUT,
                                    latch=board.BUTTON_LATCH, key_count=4,
value_when_pressed=True)
# pylint: enable=invalid-name

def mount_sd():
    """Mount the SD card"""
    spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
    sdcard = sdcardio.SDCard(spi, board.SD_CS)
    vfs = storage.VfsFat(sdcard)
    storage.mount(vfs, "/sd")

def join(*args):
    """Like posixpath.join"""
    return "/".join(args)

def shuffle(seq):
    """Shuffle a sequence using the Fisher-Yates shuffle algorithm (like
random.shuffle)"""
    for i in range(len(seq)-2):
        j = random.randint(i, len(seq)-1)
        seq[i], seq[j] = seq[j], seq[i]

# pylint: disable=too-many-locals,too-many-statements
def menu_choice(seq, *, sel_idx=0, text_font=font):
    """Display a menu and allow a choice from it"""
    gc.collect()
    board.DISPLAY.auto_refresh = True
    scroll_idx = 0
    glyph_width, glyph_height = text_font.get_bounding_box()[:2]
    num_rows = min(len(seq), board.DISPLAY.height // glyph_height)
    max_glyphs = board.DISPLAY.width // glyph_width
    palette = displayio.Palette(2)
    palette[0] = 0
    palette[1] = 0xffffff
    labels = [displayio.TileGrid(text_font.bitmap, pixel_shader=palette,
                                 width=max_glyphs+1, height=1,
                                 tile_width=glyph_width,
                                 tile_height=glyph_height)
              for i in range(num_rows)]
    terminals = [terminalio.Terminal(li, text_font) for li in labels]
    cursor = adafruit_display_text.label.Label(text_font, color=0xddddff)
    base_y = 0
    caret_offset = glyph_height//2-1
    scene = displayio.Group()
    for i, label in enumerate(labels):
        label.x = round(glyph_width * 1.5)
        label.y = base_y + glyph_height * i
        scene.append(label)
    cursor.x = 0
    cursor.y = caret_offset
```

```
    cursor.text = ">"
    scene.append(cursor)

    last_scroll_idx = max(0, len(seq) - num_rows)

    board.DISPLAY.root_group = scene
    buttons.events.clear()
    i = 0
    old_scroll_idx = None

    while True:
        enable.value = speaker.playing
        event = buttons.events.get()
        if event and event.pressed:
            return sel_idx

        joystick.poll()
        if up_key.value:
            sel_idx -= 1
        if down_key.value:
            sel_idx += 1

        sel_idx = min(len(seq)-1, max(0, sel_idx))

        if scroll_idx > sel_idx or scroll_idx + num_rows <= sel_idx:
            scroll_idx = sel_idx - num_rows // 2
        scroll_idx = min(last_scroll_idx, max(0, scroll_idx))

        board.DISPLAY.auto_refresh = False
        if old_scroll_idx != scroll_idx:
            for i in range(scroll_idx, scroll_idx + num_rows):
                j = i - scroll_idx
                new_text = ''
                if i < len(seq):
                    new_text = seq[i][:max_glyphs]
                terminals[j].write('\r\033[K')
                terminals[j].write(new_text)
        cursor.y = caret_offset + base_y + glyph_height * (sel_idx - scroll_idx)
        board.DISPLAY.auto_refresh = True
        old_scroll_idx = scroll_idx

        time.sleep(1/20)
# pylint: enable=too-many-locals

S_IFDIR = const(16384)
def isdir(x):
    """Return True if 'x' is a directory"""
    return os.stat(x)[0] & S_IFDIR

def change_stream(filename):
    """Change the global MP3Decoder object to play a new file"""
    old_stream = mp3stream.file
    mp3stream.file = open(filename, "rb")
    old_stream.close()
    return mp3stream.file

def play_one_file(idx, filename, folder, title, playlist_size):
    """Play one file, reacting to user input"""
    board.DISPLAY.auto_refresh = False

    playback_display.set_bitmap([
        filename.rsplit('.', 1)[0] + ".bmp",
        filename.rsplit('/', 1)[0] + ".bmp",
        filename.rsplit('/', 1)[0] + "/cover.bmp",
    ])

    playback_display.text = "%s\n%s" % (folder, title)

    board.DISPLAY.refresh()
```

```
    result = None
    file_size = os.stat(filename)[6]
    mp3file = change_stream(filename)
    playback_display.play(mp3stream)
    board.DISPLAY.auto_refresh = True

    while speaker.playing:

        # pylint: disable=no-member
        if gc.mem_free() < 4096:
            gc.collect()

        playback_display.rms = mp3stream.rms_level
        playback_display.progress = mp3file.tell() / file_size

        joystick.poll()
        if left_key.value:
            playback_display.move(-1)
        if right_key.value:
            playback_display.move(1)

        event = buttons.events.get()
        if event and event.pressed:
            return_now = playback_display.press(idx)
            if return_now:
                result = return_now[0]
                break

    if result is None:
        if playback_display.shuffle:
            if playback_display.shuffle:
                #  Choose a random integer .. except for this one
                result = random.randrange(playlist_size-1)
                if result >= idx:
                    result += 1
        else:
            result = (idx + 1)
    speaker.stop()
    playback_display.rms = 0

    gc.collect()

    return result

def play_all(playlist, *, folder='', trim=0, location='/sd'):
    """Play everything in 'playlist', which is relative to 'location'.

    'folder' is a display name for the user."""
    i = 0
    board.DISPLAY.root_group = playback_display.group
    playback_display.iconbar.group.y = 112
    while 0 <= i < len(playlist):
        filename = playlist[i]
        i = play_one_file(i, join(location, filename), folder, filename[trim:-4],
len(playlist))
        if i == -1:
            break
        if playback_display.repeat and i == len(playlist):
            i = 0
    speaker.stop()
    clear_display()

def longest_common_prefix(seq):
    """Find the longest common prefix between all items in sequence"""
    seq0 = seq[0]
    for i, seq0i in enumerate(seq0):
        for j in seq:
            if len(j) < i or j[i] != seq0i:
```

```
                return i
    return len(seq0)

def play_folder(location):
    """Play everything within a given folder"""
    playlist = [d for d in os.listdir(location)
            if not d.startswith('.') and d.lower().endswith('.mp3')]
    if not playlist:
        # hmm, no mp3s in a folder?  Well, don't crash okay?
        del playlist
        gc.collect()
        return
    playlist.sort()
    trim = longest_common_prefix(playlist)
    enable.value = True
    play_all(playlist, folder=location.split('/')[-1], trim=trim, location=location)
    enable.value = False


def main():
    """The main function of the player"""
    try:
        mount_sd()
    except OSError as detail:
        text = "%s\n\nInsert or re-seat\nSD card\nthen press reset" % detail.args[0]
        error_text = adafruit_display_text.label.Label(font, text=text)
        error_text.x = 8
        error_text.y = board.DISPLAY.height // 2
        g = displayio.Group()
        g.append(error_text)
        board.DISPLAY.root_group = g

        while True:
            time.sleep(1)

    while True:
        folder = playback_display.choose_folder()
        play_folder(folder)
main()
```

---

# Load your tracks

JEplayer can only play DRM-free files in the "MP3" format.  It works best with files that are stereo, a bitrate of 128kbit/s or lower, and with a sample rate of 44.1kHz.  If you have other formats or bit rates, you can use free software like Audacity (https://adafru.it/Lee) to do the conversion.

Insert the Micro SD card into the USB reader, then insert the reader into your computer.

Next, create a folder in the Micro SD card with the name of your playlist or the title of the album.

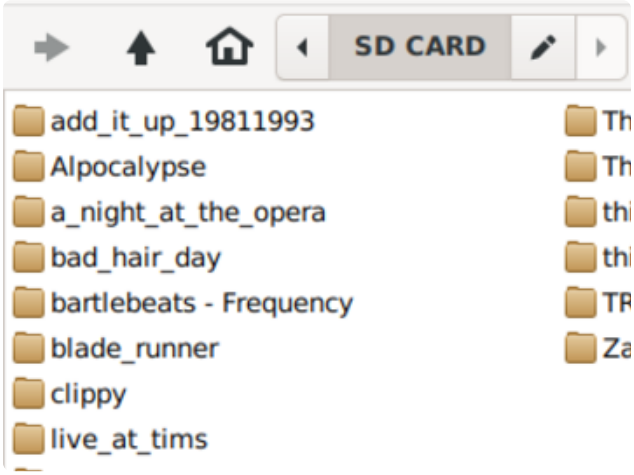
Then, copy MP3 files into the folder.

Don't put the MP3 files on the "CIRCUITPY" drive, JEplayer won't look for them there!

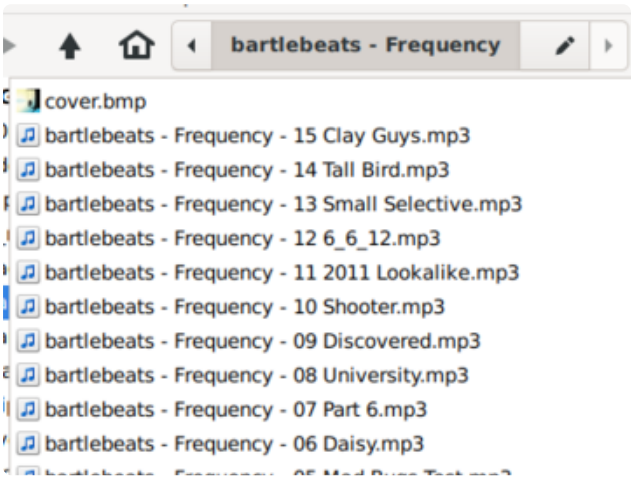
If you want to ensure that they are played in a particular order, name them "01 - First Track Name.mp3", "02 - Next Track Name.mp3" and so on.

For each additional album or playlist, create a fresh folder in the Micro SD card.

JEplayer only looks at folders in the top level, so remember not to create a folder inside of another folder!




After you've added a bunch of albums, the top level of your Micro SD card might look like this.




Here's what a typical folder full of music should look like in the file browser. Remember, this needs to be a folder inside the Micro SD card.

Need some music? We can heartily recommend Bartlebeats - Frequency (https://adafru.it/Lef), the official soldering soundtrack of Adafruit.

---

# Add Artwork

JEplayer can show artwork to go along with each track or album. Make your artwork up to 160x96 pixels and be sure to save it in uncompressed ".bmp" (Windows Bitmap) format.

Put the image alongside the mp3 files in a folder.  To apply it to all tracks, call it "cover.bmp".  To apply it to a single track give it the same name as the mp3 file.  For instance, if your track is named "03 Neat Please.mp3" make sure the image is in that folder and called "03 Neat Please.bmp".

If you downloaded Bartlebeats, here's the album artwork converted for you:

**cover.bmp**

https://adafru.it/Leg

---

# Using JEplayer

Due to technical limitations, JEplayer doesn't feature software volume control.  If you use headphones, choose a set with an in-line volume control and always listen at a safe volume.

Once you've loaded your tracks and art, eject the SD card from your computer, insert it in the PyGamer, and turn it on or press the reset switch once briefly.

Hold the PyGamer horizontally, with the stick at the left.

The screen of the PyGamer will show information about what's going on right now, and let you make selections.  Here, the playback screen is visible.

All 4 buttons always do the same thing, but what they do depends on what mode you're in.

In the album list, move the stick up and down to to choose an album, then click any button to play it.

In playback, move the stick left and right to select an icon, then click any button to activate it.

When JEplayer is playing audio, the LEDs will pulse with the intensity of the music. That really makes it a party.

When it starts, JEplayer will show you a list of folders. Move the stick up and down to choose one, or just pick the top option to get a random playlist. Press any button to start playing. The NeoPixel LEDs will pulse in time to the music.

During playback, control JEplayer using the icons at the bottom of the screen. Press left and right on the stick select an icon—the current one is outlined with a red square—and then press any button to invoke the desired icon.

Some buttons can be "active", in which case they are shown with a blue outline.

The first set of buttons control playback:

Play: Select this icon to play or pause the current track.  Active when playing audio.

Pause: Select this icon to play or pause the current track.  Active when playback is paused.

Stop: Select this icon to return to the folder listing.  You can also turn the PyGamer off at any time using the power switch.

Previous Track: Go to the previous track (or a random track, when shuffle is on)

If "All Folders" is on, going before the first track will go to another folder.  Otherwise, going before the first track will return you to the folder listing.

Next Track: Go to the next track (or a random track, when shuffle is on)

If "All Folders" is on, going past the last track will return to the first track. Otherwise, going past the last track will return you to the folder listing.

The last set of buttons control what will happen when the current track finishes.

If none of those modes are selected, then play will continue through all the tracks in the folder, and then return to the album listing.

Repeat: Switch "Repeat" mode on or off. When active, after the last track in the folder, play continues with the first track. Mutually exclusive with Shuffle and All Folders.

Shuffle: Switch "Shuffle" mode on or off. When active, after each track, select a different random track from the folder. Mutually exclusive with Repeat and All Folders.

Shuffle: Switch "Shuffle" mode on or off. When active, after each track, select a different random track from the folder. Mutually exclusive with Repeat and All Folders