Walkmp3rson: Personal MP3 'Tape' Player

Created by John Park

https://learn.adafruit.com/walkmp3rson-personal-mp3-tape-player

Last updated on 2023-11-13 11:49:50 AM EST
Table of Contents

Overview 3
• Parts

CAD Files 7
• CAD Parts List
• Build Volume
• Design Source Files

Install CircuitPython 8
• CircuitPython Quickstart
• Safe Mode
• Flash Resetting UF2

Code the Walkmp3rson 12
• Text Editor
• Download the Project Bundle
• Audio Files
• How it Works
• Setup
• Main Loop

Build the Walkmp3rson Circuit 22
• Display Connection
• Amp Prep
• Amp Connections
• Amp Gain Resistor
• Headphone Out
• On/Off Switch
• NeoKey 1x4 and QT Rotary Encoder

Assemble the Walkmp3rson 28
• Case Parts
• Display Mount
• Feather Doubler Mount
• Switch Mount
• Button Mount
• Rotary Mount
• SD Card Extender
• On/Off Mount
• STEMMA QT Connections
• Final Assembly
Overview

Look the part while you walk around town listening to your favorite mixes. CircuitPython powers this personal music player, with a stylish 3D printed case, TFT display, mech keyswitch controls and more.

Pop in a different "mix tape" SD card when you're in the mood for some different tunes.

Parts

Adafruit Feather RP2040
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather...
https://www.adafruit.com/product/4884
2.0” 320x240 Color IPS TFT Display with microSD Card Breakout
This gorgeous IPS display breakout is the best way to add a small, colorful, and bright display to any project, with excellent visibility from any angle. Since the display uses 4-wire...
https://www.adafruit.com/product/4311

NeoKey 1x4 QT I2C - Four Mechanical Key Switches with NeoPixels
The only thing better than a nice mechanical key is, perhaps, FOUR mechanical keys that also can glow any color of the rainbow - and that's what the Adafruit...
https://www.adafruit.com/product/4980

Kailh Mechanical Key Switches - Clicky White - 10 pack
For crafting your very own custom keyboard, these Kailh White Linear mechanical key switches are deeee-luxe! With smooth actuation and Cherry MX...
https://www.adafruit.com/product/4955

Cyan MA Keycaps for MX Compatible Switches - 5 pack
Dress up your mechanical keys in your favorite colors with a wide selection of gumdrop-like, retro, curvy, and stylish MA profile keycaps. Here is a 5 pack of Cyan MA...
https://www.adafruit.com/product/5174
Adafruit I2C Stemma QT Rotary Encoder Breakout with NeoPixel
Rotary encoders are sooo much fun! Twist em this way, then twist them that way. Unlike potentiometers, they go all the way around and often have little detents for tactile feedback....
https://www.adafruit.com/product/4991

FeatherWing Doubler - Prototyping Add-on For All Feather Boards
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/2890

Lithium Ion Polymer Battery - 3.7v 1200mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/258

Adafruit I2S 3W Class D Amplifier Breakout - MAX98357A
Listen to this good news - we now have an all in one digital audio amp breakout board that works incredibly well with the
https://www.adafruit.com/product/3006
<table>
<thead>
<tr>
<th>Item Description</th>
<th>Code</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro SD Card PCB Extender Short</td>
<td>4210</td>
<td><a href="https://www.adafruit.com/product/4210">https://www.adafruit.com/product/4210</a></td>
</tr>
<tr>
<td>Short Headers Kit for Feather</td>
<td>4395</td>
<td><a href="https://www.adafruit.com/product/4395">https://www.adafruit.com/product/4395</a></td>
</tr>
<tr>
<td>Short Feather Male Headers</td>
<td>2940</td>
<td><a href="https://www.adafruit.com/product/2940">https://www.adafruit.com/product/2940</a></td>
</tr>
<tr>
<td>Panel Mount 1/8&quot; / 3.5mm TRS Audio Jack</td>
<td>3002</td>
<td><a href="https://www.adafruit.com/product/3002">https://www.adafruit.com/product/3002</a></td>
</tr>
<tr>
<td>Micro Potentiometer Knob, 4 pack, orange</td>
<td>5533</td>
<td><a href="https://www.adafruit.com/product/5533">https://www.adafruit.com/product/5533</a></td>
</tr>
<tr>
<td>Micro SD Memory Card, 128MB</td>
<td>5250</td>
<td><a href="https://www.adafruit.com/product/5250">https://www.adafruit.com/product/5250</a></td>
</tr>
<tr>
<td>100K Ohm Resistor, 5% 1/4W - Pack of 25</td>
<td>2787</td>
<td><a href="https://www.adafruit.com/product/2787">https://www.adafruit.com/product/2787</a></td>
</tr>
<tr>
<td>Hook-up Wire Spool Set, 22AWG Stranded-Core - 6 x 25ft</td>
<td>3111</td>
<td><a href="https://www.adafruit.com/product/3111">https://www.adafruit.com/product/3111</a></td>
</tr>
<tr>
<td>USB Type A to Type C Cable, Approx. 1 meter / 3 ft long</td>
<td>4474</td>
<td><a href="https://www.adafruit.com/product/4474">https://www.adafruit.com/product/4474</a></td>
</tr>
</tbody>
</table>
CAD Files

CAD Parts List
STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below:

wp-back-cover
wp-button-case
wp-button-cover
wp-front-case
wp-keysswitch-plate
wp-switch-holder
wp-trs-rotary

STLs.zip
Download CAD Source

Build Volume
The parts require a 3D printer with a minimum build volume.

110mm (X) x 70mm (Y) x 30mm (Z)
Design Source Files
The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit’s boards, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo.

Install CircuitPython

CircuitPython is a derivative of MicroPython 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 drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org

Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.
To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.

The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

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

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.
Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

**Entering Safe Mode**

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

**In Safe Mode**

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.
CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.
Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

**Flash Resetting UF2**

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep
clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

---

**Code the Walkmp3rson**

**Text Editor**

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](#).

Alternatively, you can use any text editor that saves simple text files.

**Download the Project Bundle**

Your project will use a specific set of CircuitPython libraries, sample .mp3s, graphic .bmp, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your board's CIRCUITPY drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

**Audio Files**

Copy the .mp3 files from the bundle onto your SD card using a USB SD card reader.

To make your own files, follow the info in [this guide](#) on converting audio files to mono .mp3 files.

If you hear clicks and pops, be sure that your .mp3 files are of 128kbit/s or lower bitrate, with sample rates from 8kHz to 24kHz.

```python
# SPDX-FileCopyrightText: 2022 John Park and Tod Kurt for Adafruit Industries
# SPDX-License-Identifier: MIT
# Walkmp3rson digital cassette tape player (ok fine it's just SD cards)
import time
import os
import board
import busio
```

©Adafruit Industries
import sdcardio
import storage
import audiomixer
import audiobusio
import audiomp3
from adafruit_neokey.neokey1x4 import NeoKey1x4
from adafruit_seesaw import seesaw, rotaryio
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_st7789 import ST7789
from adafruit_progressbar.progressbar import HorizontalProgressBar
from adafruit_progressbar.verticalprogressbar import VerticalProgressBar
displayio.release_displays()

# SPI for TFT display, and SD Card reader on TFT display
spi = board.SPI()
# display setup
tft_cs = board.D6
tft_dc = board.D9
tft_reset = board.D12
display_bus = displayio.FourWire(spi, command=tft_dc, chip_select=tft_cs,
reset=tft_reset)
display = ST7789(display_bus, width=320, height=240, rotation=90)

# SD Card setup
sd_cs = board.D13
sdcard = sdcardio.SDCard(spi, sd_cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")

# I2C NeoKey setup
i2c = busio.I2C(board.SCL, board.SDA)
neokey = NeoKey1x4(i2c, addr=0x30)
amber = 0x300800
red = 0x900000
green = 0x009000
neokey.pixels.fill(amber)
keys = [
    (neokey, 0, green),
    (neokey, 1, red),
    (neokey, 2, green),
    (neokey, 3, green),
]
# states for key presses
key_states = [False, False, False, False]

# STEMMA QT Rotary encoder setup
rotary_seesaw = seesaw.Seesaw(i2c, addr=0x36)  # default address is 0x36
encoder = rotaryio.IncrementalEncoder(rotary_seesaw)
last_encoder_pos = 0

# file system setup
mp3s = []
for filename in os.listdir("/sd"):  
    if filename.lower().endswith(".mp3") and not filename.startswith( ":"):
        mp3s.append("/sd/"+filename)

mp3s.sort()  # sort alphanumerically for mixtape order, e.g., "1_King_of_Rock.mp3"
for mp3 in mp3s:
    print(mp3)

track_number = 0
mp3_filename = mp3s[track_number]
mp3_bytes = os.stat(mp3_filename)[6]  # size in bytes is position 6
mp3_file = open(mp3_filename, "rb")
mp3stream = audiomp3.MP3Decoder(mp3_file)

def tracktext(full_path_name, position):
    return full_path_name.split('_')[position].split('.')[0]

# LRC is word_select, BCLK is bit_clock, DIN is data_pin.
# Feather RP2040
audio = audiobusio.I2SOut(bit_clock=board.D24, word_select=board.D25, data=board.A3)
# Feather M4
# audio = audiobusio.I2SOut(bit_clock=board.D1, word_select=board.D10, data=board.D11)
mixer = audiomixer.Mixer(voice_count=1, sample_rate=22050, channel_count=1, bits_per_sample=16, samples_signed=True, buffer_size=32768)
mixer.voice[0].level = 0.15

# Colors
blue_bright = 0x17afcf
blue_mid = 0x0d6173
blue_dark = 0x041f24

orange_bright = 0xda8c57
orange_mid = 0xa46032
orange_dark = 0x472a16

# display
main_display_group = displayio.Group()  # everything goes in main group
display.root_group = main_display_group  # show main group (clears screen, too)

# background bitmap w OnDiskBitmap
tape_bitmap = displayio.OnDiskBitmap(open("mp3_tape.bmp", "rb"))
tape_tilegrid = displayio.TileGrid(tape_bitmap, pixel_shader=tape_bitmap.pixel_shader)
main_display_group.append(tape_tilegrid)

# song name label
song_name_text_group = displayio.Group(scale=3, x=90, y=44)  # text label goes in this Group
song_name_text = tracktext(mp3_filename, 2)
song_name_label = label.Label(terminalio.FONT, text=song_name_text, color=orange_bright)
song_name_text_group.append(song_name_label)  # add the label to the group
main_display_group.append(song_name_text_group)  # add to the parent group

# artist name label
artist_name_text_group = displayio.Group(scale=2, x=92, y=186)
artist_name_text = tracktext(mp3_filename, 1)
artist_name_label = label.Label(terminalio.FONT, text=artist_name_text, color=orange_bright)
artist_name_text_group.append(artist_name_label)
main_display_group.append(artist_name_text_group)

# song progress bar
progress_bar = HorizontalProgressBar((72, 144), (174, 12), bar_color=blue_bright, outline_color=blue_mid, fill_color=blue_dark,)
main_display_group.append(progress_bar)

# volume level bar
volume_bar = VerticalProgressBar((304, 40), (8, 170), bar_color=orange_bright, outline_color=orange_mid, fill_color=orange_dark,
main_display_group.append(volume_bar)
volume_bar.value = mixer.voice[0].level * 100

def change_track(tracknum):
    # pylint: disable=global-statement
    global mp3_filename
    mp3_filename = mp3s[tracknum]
    song_name_fc = tracktext(mp3_filename, 2)
    artist_name_fc = tracktext(mp3_filename, 1)
    mp3_file_fc = open(mp3_filename, "rb")
    mp3stream_fc = audiomp3.MP3Decoder(mp3_file_fc)
    mp3_bytes_fc = os.stat(mp3_filename)[6]  # size in bytes is position 6
    return (mp3_file_fc, mp3stream_fc, song_name_fc, artist_name_fc, mp3_bytes_fc)

print("Walkmp3rson")
play_state = False  # so we know if we're auto advancing when mixer finishes a song
last_debug_time = 0  # for timing track position
reels_anim_frame = 0
last_percent_done = 0.01
audio.play(mixer)

while True:
    encoder_pos = -encoder.position
    if encoder_pos != last_encoder_pos:
        encoder_delta = encoder_pos - last_encoder_pos
        volume_adjust = min(max((mixer.voice[0].level + (encoder_delta*0.005)), 0.0), 1.0)
        mixer.voice[0].level = volume_adjust
        last_encoder_pos = encoder_pos
    volume_bar.value = mixer.voice[0].level * 100

    if play_state is True:  # if not stopped, auto play next song
        if time.monotonic() - last_debug_time > 0.2:  # so we can check track
            last_debug_time = time.monotonic()
            bytes_played = mp3_file.tell()
            percent_done = (bytes_played / mp3_bytes)
            progress_bar.value = min(max(percent_done * 100, 0), 100)

        if not mixer.playing:
            print("next song")
            audio.pause()
            track_number = ((track_number + 1) % len(mp3s))
            mp3_file, mp3stream, song_name, artist_name, mp3_bytes =
            change_track(track_number)

            song_name_label.text = song_name
            artist_name_label.text = artist_name
            mixer.voice[0].play(mp3stream, loop=False)
            time.sleep(.1)
            audio.resume()

        if key_number == 0:  # previous track
            audio.pause()
            track_number = ((track_number - 1) % len(mp3s))
            mp3_file, mp3stream, song_name, artist_name, mp3_bytes =
            change_track(track_number)

            song_name_label.text = song_name
            artist_name_label.text = artist_name
            mixer.voice[0].play(mp3stream, loop=False)
            play_state = True
            time.sleep(.1)
            audio.resume()
if key_number == 1:  # Play/pause
    if play_state:
        audio.pause()
        play_state = False
    else:
        audio.resume()
        play_state = True

if key_number == 2:  # Play track from beginning
    audio.pause()
    mixer.voice[0].play(mp3stream, loop=False)
    song_name_label.text = tracktext(mp3_filename, 2)
    artist_name_label.text = tracktext(mp3_filename, 1)
    play_state = True
    time.sleep(.1)
    audio.resume()

if key_number == 3:  # next track
    audio.pause()
    track_number = ((track_number + 1) % len(mp3s))
    mp3_file, mp3stream, song_name, artist_name, mp3_bytes = change_track(track_number)
    song_name_label.text = song_name
    artist_name_label.text = artist_name
    mixer.voice[0].play(mp3stream, loop=False)
    play_state = True
    time.sleep(.1)
    audio.resume()

    if not neokey[key_number] and key_states[key_number]:
        neokey.pixels[key_number] = amber
        key_states[key_number] = False

How it Works

First, there are a dozen or so libraries to install! These help us use the SD card, TFT display, audio mixer and mp3 decoder, NeoKeys, rotary encoder, and more.

import time
import os
import board
import busio
import sdcardio
import storage
import audiomixer
import audiobusio
import audiomp3
from adafruit_neokey.neokey1x4 import NeoKey1x4
from adafruit_seesaw import seesaw, rotaryio
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_st7789 import ST7789
from adafruit_progressbar.progressbar import HorizontalProgressBar
from adafruit_progressbar.verticalprogressbar import VerticalProgressBar

Setup

Next, the TFT display is set up on the SPI bus.
displayio.release_displays()
#
# SPI for TFT display, and SD Card reader on TFT display
spi = board.SPI()
#
# display setup
tft_cs = board.D6
tft_dc = board.D9
tft_reset = board.D12
display_bus = displayio.FourWire(spi, command=tft_dc, chip_select=tft_cs,
reset=tft_reset)
display = ST7789(display_bus, width=320, height=240, rotation=90)

Then, the SD card reader is set up, also on SPI.
#
# SD Card setup
sd_cs = board.D13
sdcard = sdcardio.SDCard(spi, sd_cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd"

NeoKey

The NeoKey is set up on the I2C bus next, including color definitions and key states.

# I2C NeoKey setup
i2c = busio.I2C(board.SCL, board.SDA)
neokey = NeoKey1x4(i2c, addr=0x30)
amber = 0x300800
red = 0x900000
green = 0x009000
neokey.pixels.fill(amber)
keys = [
    (neokey, 0, green),
    (neokey, 1, red),
    (neokey, 2, green),
    (neokey, 3, green),
]
#
# states for key presses
key_states = [False, False, False, False]

Rotary Encoder

The rotary encoder is set up as a seesaw device.

# STEMMA QT Rotary encoder setup
rotary_seesaw = seesaw.Seesaw(i2c, addr=0x36)  # default address is 0x36
coder = rotaryio.IncrementalEncoder(rotary_seesaw)
last_encoder_pos = 0

SD Card Filesystem

To read the .mp3 files from the SD card, the filesystem is defined and listed.
# file system setup
mp3s = []
for filename in os.listdir('/sd'):
    if filename.lower().endswith('.mp3') and not filename.startswith('.'):  
        mp3s.append("/sd/"+filename)
mp3s.sort()  # sort alphanumerically for mixtape order, e.g., "1_King_of_Rock.mp3"
for mp3 in mp3s:
    print(mp3)

track_number = 0
mp3_filename = mp3s[track_number]
mp3_bytes = os.stat(mp3_filename)[6]  # size in bytes is position 6
mp3_file = open(mp3_filename, "rb")
mp3stream = audiomp3.MP3Decoder(mp3_file)

Tracktext Function

This function is used to parse the track names for display, removing the track number from the beginning, and returning the Artist Name and Song Name for display. This is the naming convention:

number_artist_song.mp3

For example:

03_Bartlebeats_Daisy.mp3

That will be the third song on the "mix tape" SD card, with the artist name Bartlebeats and song name Daisy.

```python
def tracktext(full_path_name, position):
    return full_path_name.split('_')[position].split('.')[0]
```

Audio

The audio is set up as an I2S audio output on the audiobus with pins defined for word select, bit clock, and data.

```python
# LRC is word select, BCLK is bit_clock, DIN is data_pin.
# Feather RP2040
audio = audiobusio.I2SOut(bit_clock=board.D24, word_select=board.D25, data=board.A3)
# Feather M4
# audio = audiobusio.I2SOut(bit_clock=board.D1, word_select=board.D10, 
data=board.D11)
```
Mixer

The mixer object is created, with the specific settings we're using for the .mp3 files and an initial volume level.

```python
mixer = audiomixer.Mixer(voice_count=1, sample_rate=22050, channel_count=1, 
bits_per_sample=16, samples_signed=True)
mixer.voice[0].level = 0.15
```

Screen Colors

Colors are defined for screen text and graphics.

```python
# Colors
blue_bright = 0x17afcf
blue_mid = 0x0d6173
blue_dark = 0x041f24
orange_bright = 0xda8c57
orange_mid = 0xa46032
orange_dark = 0x472a16
```

Displayio

The displayio group and on disk bitmap are defined next.

```python
display
main_display_group = displayio.Group()  # everything goes in main group
display.root_group = main_display_group  # show main group (clears screen, too)

# background bitmap w OnDiskBitmap
tape_bitmap = displayio.OnDiskBitmap(open("mp3_tape.bmp", "rb"))
tape_tilegrid = displayio.TileGrid(tape_bitmap, 
pixel_shader=tape_bitmap.pixel_shader)
main_display_group.append(tape_tilegrid)
```

Text Labels

The text for song and artist are set up here.

```python
# song name label
song_name_text_group = displayio.Group(scale=3, x=90, y=44)  # text label goes in this Group
song_name_text = tracktext(mp3_filename, 2)
song_name_label = label.Label(terminalio.FONT, text=song_name_text, 
color=orange_bright)
song_name_text_group.append(song_name_label)  # add the label to the group
main_display_group.append(song_name_text_group)  # add to the parent group

# artist name label
artist_name_text_group = displayio.Group(scale=2, x=92, y=186)
artist_name_text = tracktext(mp3_filename, 1)
```
Progress Bars

We’re using the horizontal progress bar to visualize song length and a vertical progress bar for volume.

```python
# song progress bar
progress_bar = HorizontalProgressBar(  
    (72, 144),  
    (174, 12),  
    bar_color=blue_bright,  
    outline_color=blue_mid,  
    fill_color=blue_dark,
)
main_display_group.append(progress_bar)

# volume level bar
volume_bar = VerticalProgressBar(  
    (304, 40),  
    (8, 170),  
    bar_color=orange_bright,  
    outline_color=orange_mid,  
    fill_color=orange_dark,
)
main_display_group.append(volume_bar)
volume_bar.value = mixer.voice[0].level * 100
```

**change_track() Function**

This function is called whenever a track change happens. This can be when the previous or next song buttons are pressed, or when one song ends and the next needs to begin.

```python
def change_track(tracknum):
    global mp3_filename
    mp3_filename = mp3s[tracknum]
    song_name_fc = tracktext(mp3_filename, 2)
    artist_name_fc = tracktext(mp3_filename, 1)
    mp3_file_fc = open(mp3_filename, "rb")
    mp3stream_fc = audiomp3.MP3Decoder(mp3_file_fc)
    mp3_bytes_fc = os.stat(mp3_filename)[6]  # size in bytes is position 6
    return (mp3_file_fc, mp3stream_fc, song_name_fc, artist_name_fc, mp3_bytes_fc)
```

**States**

State variables are set, and then the audio mixer is turned on.

```python
play_state = False  # so we know if we're auto advancing when mixer finishes a song
last_debug_time = 0  # for timing track position
```
Main Loop

The main loop of the program does the following:

- Checks for encoder input to change volume
- Checks the NeoKeys for input
- Plays the current song until it is paused, finishes, is restarted, or next/previous buttons are pressed

```python
while True:
    encoder_pos = -encoder.position
    if encoder_pos != last_encoder_pos:
        encoder_delta = encoder_pos - last_encoder_pos
        volume_adjust = min(max((mixer.voice[0].level + (encoder_delta*0.005)), 0.0), 1.0)
        mixer.voice[0].level = volume_adjust
        last_encoder_pos = encoder_pos
        volume_bar.value = mixer.voice[0].level * 100

    if play_state is True:  # if not stopped, auto play next song
        if time.monotonic() - last_debug_time > 0.2:  # so we can check track progress
            last_debug_time = time.monotonic()
            bytes_played = mp3_file.tell()
            percent_done = (bytes_played / mp3_bytes)
            progress_bar.value = min(max(percent_done * 100, 0), 100)

        if not mixer.playing:
            print("next song")
            audio.pause()
            track_number = ((track_number + 1) % len(mp3s))
            mp3_file, mp3stream, song_name, artist_name, mp3_bytes = change_track(track_number)
            song_name_label.text = song_name
            artist_name_label.text = artist_name
            mixer.voice[0].play(mp3stream, loop=False)
            time.sleep(.1)
            audio.resume()

# Use the NeoKeys as transport controls
for k in range(len(keys)):
    neokey, key_number, color = keys[k]
    if neokey[key_number] and not key_states[key_number]:
        key_states[key_number] = True
        neokey.pixels[key_number] = color

    if key_number == 0:  # previous track
        audio.pause()
        track_number = ((track_number - 1) % len(mp3s))
        mp3_file, mp3stream, song_name, artist_name, mp3_bytes = change_track(track_number)
        song_name_label.text = song_name
        artist_name_label.text = artist_name
        mixer.voice[0].play(mp3stream, loop=False)
        play_state = True
        audio.resume()
```

©Adafruit Industries
time.sleep(.1)
audio.resume()

if key_number == 1:  # Play/pause
    if play_state:
        audio.pause()
        play_state = False
    else:
        audio.resume()
        play_state = True

if key_number == 2:  # Play track from beginning
    audio.pause()
    mixer.voice[0].play(mp3stream, loop=False)
    song_name_label.text = tracktext(mp3_filename, 2)
    artist_name_label.text = tracktext(mp3_filename, 1)
    play_state = True
    time.sleep(.1)
audio.resume()

if key_number == 3:  # next track
    audio.pause()
    track_number = ((track_number + 1) % len(mp3s))
    mp3_file, mp3stream, song_name, artist_name, mp3_bytes = change_track(track_number)
    song_name_label.text = song_name
    artist_name_label.text = artist_name
    mixer.voice[0].play(mp3stream, loop=False)
    play_state = True
    time.sleep(.1)
audio.resume()

if not neokey[key_number] and key_states[key_number]:
    neokey.pixels[key_number] = amber
    key_states[key_number] = False

---

Build the Walkmp3rson Circuit

---

©Adafruit Industries
Display Connection

Wire the display as shown in the Fritzing diagram and the photos here to the FeatherWing Doubler.

There are many ways to do this -- I used right angled headers and Dupont connector silicone jumper wires.
Amp Prep
Solder on header pins to the amp board so you can mount it to the Doubler.

I chose to solder a small socket header to connect the amp breakout to the headphone jack using jumper cables later.

Amp Connections
Solder the amp breakout to the long free row of the doubler as shown.

Then, use short wires to connect the legs to their respective pins on the Feather via the Doubler connections.
Amp Gain Resistor
Solder a 100KΩ resistor from the Gain pin to the Vin pin. This sets the amplifier gain to 3dB, which works well for the headphone output.

Headphone Out
To keep the project simple we're using a single mono amplifier. Since most headphones have a stereo TRS 3.5mm plug, you'll run the audio signal to both the tip and ring connectors.

Run a single wire to the ground connector.

I used Dupont connector cables for this and dressed them with heat shrink tubing.
On/Off Switch
Solder a wire from the Doubler/Feather Enable pin to one side of a SPDT switch, and another wire from GND to the center position of the switch.
NeoKey 1x4 and QT Rotary Encoder

Use STEMMA QT cables to connect the Feather to the NeoKeys and rotary encoder breakout.

Plug in the battery as well -- since the amp Vin is connected to the Feather's Bat output you'll need the battery connected even when plugged into USB for coding.
Assemble the Walkmp3rson

Case Parts

Print the case parts in 80s-tastic colors (or any colors you like). The files and notes for printing can be found on the CAD Files page.

Then, screw in the nylon standoffs for the Doubler mounting.
Display Mount
Mount the display as shown, using M2.5 hardware.

Feather Doubler Mount
Connect the Doubler to the standoffs as shown.
Switch Mount
Mount the switch into the switch housing thingamajob.
Button Mount

Insert the keyswitches into the mounting plate.

Then, insert the assembly into the case top.

Carefully press the switches into the NeoKey PCB.

Once assembled, add your keycaps.
Rotary Mount
Fit the rotary encoder breakout into the case side, then place the washer over the shaft. Screw on the nut to secure it in place.

Add a knob to the shaft as shown.
SD Card Extender
Insert the prepared SD card mix tape into the SD card extender, then insert it all into the display as shown.
On/Off Mount
Use M2.5 hardware to fasten the switch mount to the case as shown.
STEMMA QT Connections
Use 100mm STEMMA QT cables to connect the Feather RP2040 to the rotary encoder and NeoKey breakout.

Insert the panel mount headphone jack and screw on the retaining nut.
Final Assembly

Press fit the front and back together and use the side piece to hold them together.

Then, press the top piece into place.