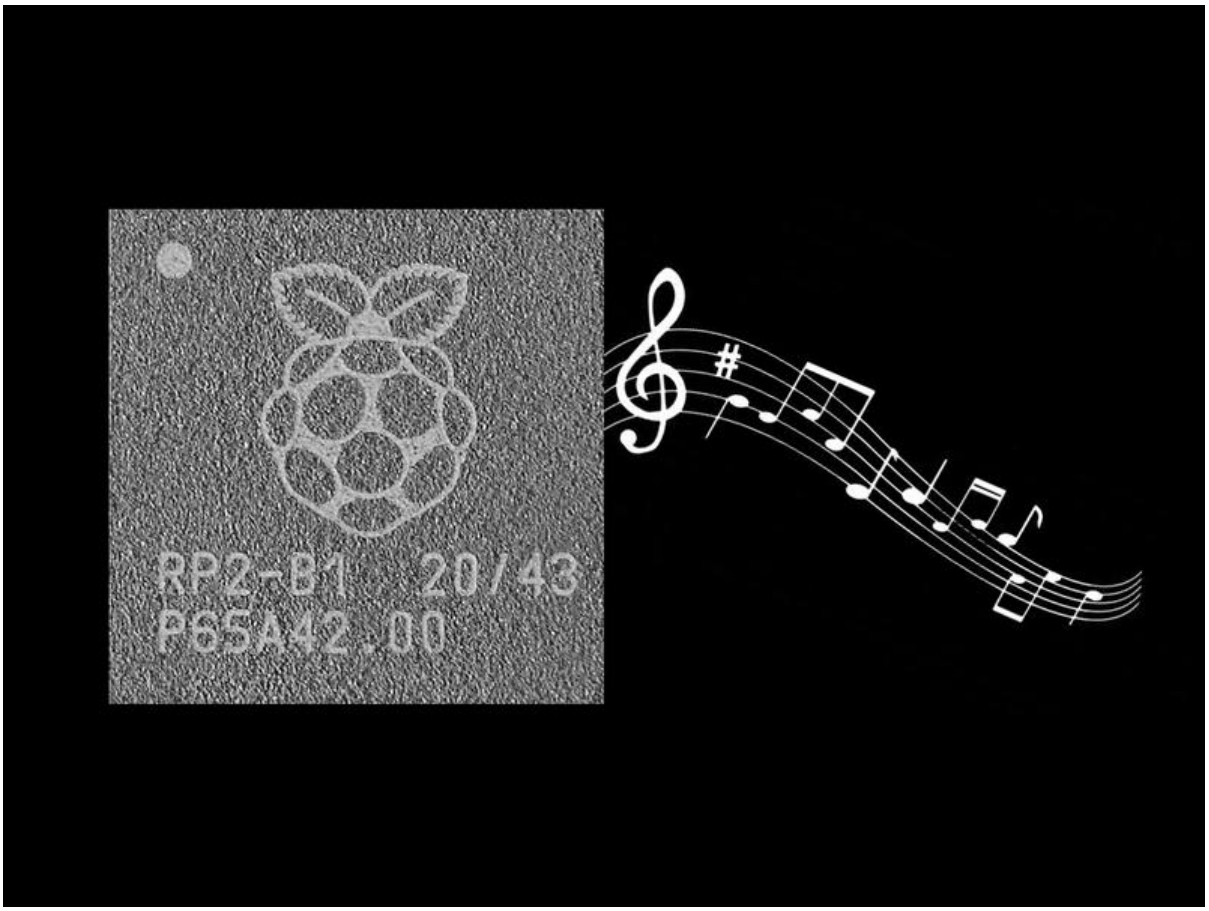
MP3 Playback on RP2040 with CircuitPython

Created by Kattni Rembor



https://learn.adafruit.com/mp3-playback-rp2040

Last updated on 2022-12-20 07:27:38 PM EST

Table of Contents

Pico PWM MP3 3

- The Speaker
- CircuitPython-Compatible MP3 Files
- Playing an MP3 File
- Playing Multiple MP3 Files
- CircuitPython MP3 Capable Pins

MacroPad MP3 8

- Playing an MP3 File
- Adding Your Own MP3 Files

Pico I2S MP3 11

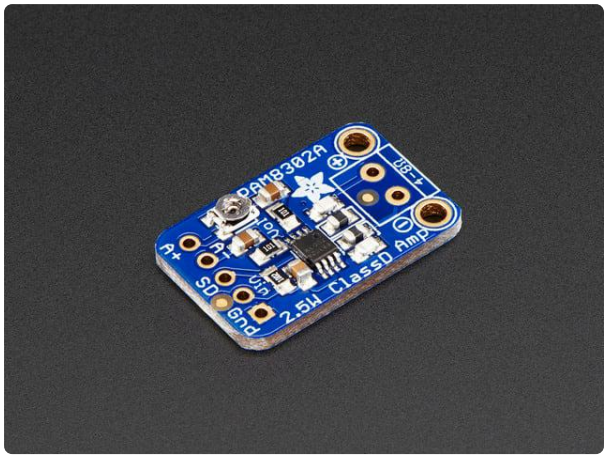
- I2S and CircuitPython
- Necessary Hardware
- Wiring the MAX98357A
- I2S Tone Playback
- I2S WAV File Playback
- I2S MP3 File Playback
- CircuitPython I2S-Compatible Pin Combinations

Pico PWM MP3

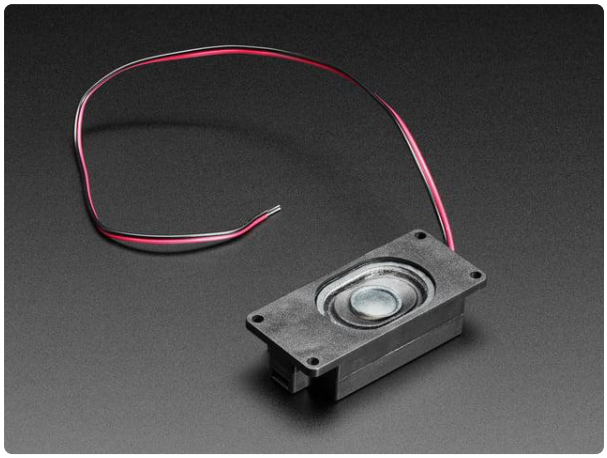
Compressed audio can be a nice alternative to uncompressed WAV files, especially when you have a small filesystem like that on many CircuitPython boards, as WAV files get sizeable quite quickly. You can listen to a much longer playlist with CircuitPython, using the built in MP3 playback capability!

Necessary Hardware

You'll need the following additional hardware to complete the examples on this page.



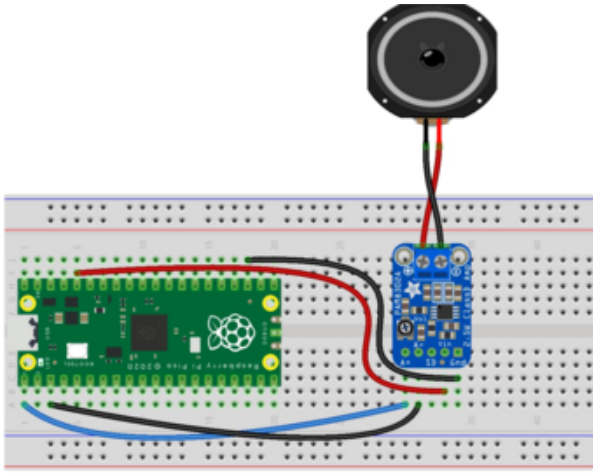
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302
This super small mono amplifier is surprisingly powerful - able to deliver up to 2.5 Watts into 4-8 ohm impedance speakers. Inside the miniature chip is a class D controller, able to...
https://www.adafruit.com/product/2130



Mono Enclosed Speaker with Plain Wires - 3W 4 Ohm
Listen up! This single 2.8" x 1.2" speaker is the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We...
https://www.adafruit.com/product/4445

The Speaker

To connect a speaker up to the Raspberry Pi Pico, you'll use a PAM8302 amplifier. Wire it up as shown below.



PAM8302 A+ to Pico GP0
PAM8302 A- to Pico GND
PAM8302 VIN to Pico 3.3v
PAM8302 GND to Pico GND
PAM8302 screw terminal + to speaker +
PAM8302 screw terminal - to speaker -

CircuitPython-Compatible MP3 Files

CircuitPython supports any MP3 file, as long as it is the right bit rate and sample rate for your board.

Mono and stereo files less than 64kbit/s work, with sample rates from 8kHz to 24kHz. The RP2040 has a PWM output with 10 bits, so there's not much point in using high bit rates.

Be aware, doing things like updating a display, or having intense flash activity like reading and writing files can result in distorted sounds or noise during playback.

You can find an example of converting an MP3 file using Audacity in this guide (). The parameters suggested above may not be exactly what's in the guide, but the concept will be the same.

Playing an MP3 File

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should now look something like this:


```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
CircuitPython single MP3 playback example for Raspberry Pi Pico.
Plays a single MP3 once.
"""
import board
import audiomp3
import audiopwmio

audio = audiopwmio.PWMAudioOut(board.GP0)

decoder = audiomp3.MP3Decoder(open("slow.mp3", "rb"))

audio.play(decoder)
while audio.playing:
    pass

print("Done playing!")
```

As soon as you save, the MP3 will begin playing! It plays only once. Connect to the serial console (), and reload to play it again.

First, you import the necessary modules. All of these modules are built into CircuitPython, so this example does not require you to copy any external libraries to your board. Then, you setup the `audio` object and provide it the speaker pin.

Next, you create the `decoder` object and tell it the name of the MP3 file you'd like to play, in this case, `"slow.mp3"`.

Then, you use the `audio` object to play the decoded MP3 file. `while` the audio is playing, `pass`, or do nothing. (You can add other code here such as blinking an LED or whatever you like.)

Finally, you print `Done playing!` to the serial console to let you know the MP3 playback has concluded.

That's all there is to playing a single MP3 file using CircuitPython!

Playing Multiple MP3 Files

The previous example plays a single MP3 file. But what if you want to include a playlist? This example has you covered.

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
CircuitPython multiple MP3 playback example for Raspberry Pi Pico.
Plays two MP3 files consecutively, once time each.
"""
import board
import audiomp3
import audiopwmio

audio = audiopwmio.PWMAudioOut(board.GP0)

mp3files = ["slow.mp3", "happy.mp3"]

mp3 = open(mp3files[0], "rb")
decoder = audiomp3.MP3Decoder(mp3)

for filename in mp3files:
    decoder.file = open(filename, "rb")
    audio.play(decoder)
    print("Playing:", filename)
    while audio.playing:
        pass

print("Done playing!")
```

As soon as you save, the MP3s will begin playing! They play only once. Connect to the serial console (), and reload to play them again.

The code starts out the same, with the same imports and audio setup.

This time, however, you create a list of `mp3files`, with each file name as a string, including the .mp3. There are only two in this list, but you could add as many as you like to the list, copy the associated files to your CIRCUITPY drive, and the code will play them each consecutively.

Then, you open the first MP3 file in the list, and use it to construct the `decoder` object. This is necessary to be able to reuse the `decoder` object multiple times later.

Then, for each file in the mp3files list, you open the file in the decoder, use the audio object to play the decoded MP3 file. You print to the serial console `Playing:` and the name of the file currently playing. Then, `while` the audio is playing, `pass`, or do nothing.

Finally, you print `Done playing!` to the serial console to let you know the MP3 playback has concluded.

That's all there is to playing multiple MP3 files using CircuitPython!

CircuitPython MP3 Capable Pins

MP3 playback is supported on specific pins. The good news is, there's a simple way to find out which pins support audio playback.

Save the following file as code.py on your CIRCUITPY drive. Then, connect to the serial console () to see a list of pins printed out. This file runs only once, so if you do not see anything in the output, press CTRL+D to reload and run the code again.

This microcontroller also support I2S, which allows use of a higher quality external DAC. The sample rates and bit rates are still limited to those shown above, but the audio quality can be substantially better. This script does not provide I2S-capable pins.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
# SPDX-License-Identifier: MIT
"""
CircuitPython Audio-capable pin identifying script
"""
import board
from microcontroller import Pin
try:
    from audioio import AudioOut
except ImportError:
    from audiopwmio import PWMAudioOut as AudioOut


def is_audio(audio_pin_name):
    try:
        p = AudioOut(audio_pin_name)
        p.deinit()
        return True
    except ValueError:
        return False
    except RuntimeError:
        return False
```

```
def get_unique_pins():
    exclude = [
        getattr(board, p)
        for p in [
            # This is not an exhaustive list of unexposed pins. Your results
            # may include other pins that you cannot easily connect to.
            "NEOPIXEL",
            "DOTSTAR_CLOCK",
            "DOTSTAR_DATA",
            "APA102_SCK",
            "APA102_MOSI",
            "LED",
            "SWITCH",
            "BUTTON",
        ]
        if p in dir(board)
    ]
    pins = [
        pin
        for pin in [getattr(board, p) for p in dir(board)]
        if isinstance(pin, Pin) and pin not in exclude
    ]
    unique = []
    for p in pins:
        if p not in unique:
            unique.append(p)
    return unique


for audio_pin in get_unique_pins():
    if is_audio(audio_pin):
        print("Audio pin:", audio_pin)
```

MacroPad MP3

The Adafruit MacroPad runs a Raspberry Pi RP2040 microcontroller and has a built in speaker that makes it simple to play tones and audio clips. The Adafruit CircuitPython MacroPad () library includes `play_file()` which supports MP3 playback, making it super simple to play MP3s on your MacroPad. This page explores using your MacroPad to play MP3 files with the Adafruit CircuitPython MacroPad library.

To get your MacroPad setup with the latest CircuitPython, check out the CircuitPython page in the MacroPad guide (). You can find details on using the MacroPad library the MacroPad CircuitPython Library page in the MacroPad guide ().

To install the Adafruit CircuitPython MacroPad library and its dependencies, follow the instructions below to download the Project Bundle. If you choose to install the library and its dependencies manually, please check out this guide ().

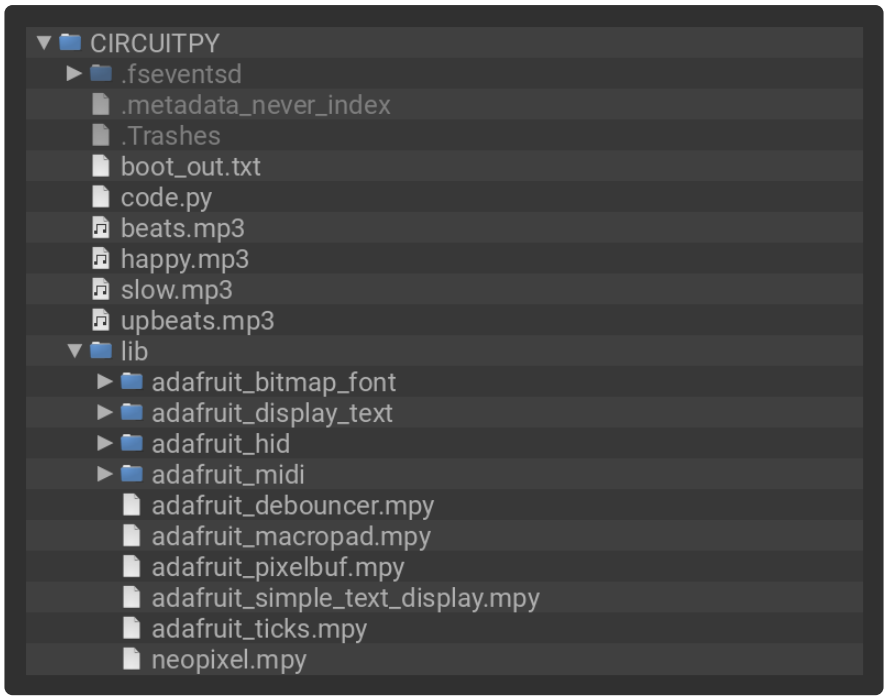
Playing an MP3 File

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
"""
MacroPad MP3 playback demo. Plays one of four different MP3 files when one of the
first four keys
is pressed. All keys light up a color of the rainbow when pressed, but no audio is
played for the
rest of the keys.
"""
from rainbowio import colorwheel
from adafruit_macropad import MacroPad

macropad = MacroPad()

# To include more MP3 files, add the names to this list in the same manner as the
others.
# Then, press the key associated with the file's position in the list to play the
file!
audio_files = ["slow.mp3", "happy.mp3", "beats.mp3", "upbeats.mp3"]

while True:
    key_event = macropad.keys.events.get()

    if key_event:
        if key_event.pressed:
            macropad.pixels[key_event.key_number] = colorwheel(
                int(255 / 12) * key_event.key_number
            )
            if key_event.key_number < len(audio_files):
                macropad.play_file(audio_files[key_event.key_number])

        else:
            macropad.pixels.fill((0, 0, 0))
```

Your CIRCUITPY drive should resemble the following image.



Now, press any one of the first four keys to hear some beats! Once the associated file is finish playing, you can press another of the first four keys to listen to a different one. You can press any of the rest of the keys to see them light up, but no audio will be played.

The video above plays only the slow.mp3 file as a demonstration of what MP3 playback sounds like on the MacroPad.

First, you import `rainbowio` and `adafruit_macropad`, and instantiate the MacroPad library.

Next, you create a list called `audio_files`, and include the names of the MP3 files you wish to play with the full name including .mp3 in quotes.

Inside the loop, the first thing you do is setup to look for the key press by creating the `key_event` variable and assigning it to `macropad.keys.events.get()`. Then, you check to see if there is a `key_event` (i.e. a key being pressed). If it is a key being pressed (`key_event.pressed`), you light up the key using the `key_number` to generate a `colorwheel()` value.

Then, if the `key_number` is less than the length of the list, you play the file in the list associated with the key number. This works because Python begins counting at 0. Therefore, the first key is number 0 and the fourth key is number 3. The length of the list is 4 (there are four items), so the key numbers associated with the files in the list will always be one less than the length of the list!

Finally, you turn off the NeoPixels when the keys are no longer being pressed and the files are no longer playing.

That's all there is to playing MP3s using the Adafruit CircuitPython MacroPad library and the Adafruit MacroPad!

Adding Your Own MP3 Files

Including additional or different MP3 files is simple. To include additional files, add them to the following line:

```
audio_files = ["slow.mp3", "happy.mp3", "beats.mp3", "upbeats.mp3"]
```

To include files other than the default ones, simply change the name of one of the current files to match the name of the file you wish to include. If you wanted the first key to instead play bleeps.mp3, you would update `audio_files =` to the following:

```
audio_files = ["bleeps.mp3", "happy.mp3", "beats.mp3", "upbeats.mp3"]
```

If you wanted to include an additional MP3 file called bloops.mp3, you would update `audio_files =` to the following:

```
audio_files = ["slow.mp3", "happy.mp3", "beats.mp3", "upbeats.mp3", "bloops.mp3"]
```

You can add up to twelve files, one for each key!

Pico I2S MP3

I2S, or Inter-IC Sound, is a standard for transmitting digital audio data. It requires at least three connections. The first connection is a clock, called bit clock (BCLK, or sometimes written as serial clock or SCK). The second connection, which determines the channel (left or right) being sent, is called word select (WS). When stereo data is sent, WS is toggled so that the left and right channels are sent alternately, one data word at a time. The third connection, which transmits the data, is called serial data (S D).

Typically, there is a transmitter device which generates the bit clock, word select signal, and the data, and sends them to a receiver device. In this case, your microcontroller acts as the transmitter, and an I2S breakout acts as the receiver. The

MAX98357A () is an example of an I2S class D amplifier that allows you to connect directly to a speaker such as this one ().

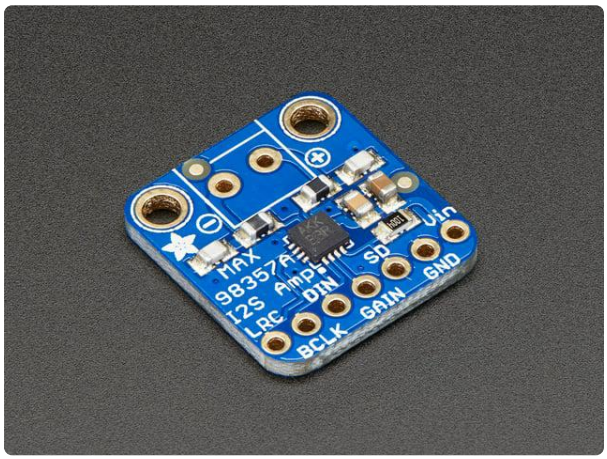
I2S and CircuitPython

CircuitPython supports sending I2S audio signals using the `audiobusio` module, making it simple to use the I2S interface with your microcontroller.

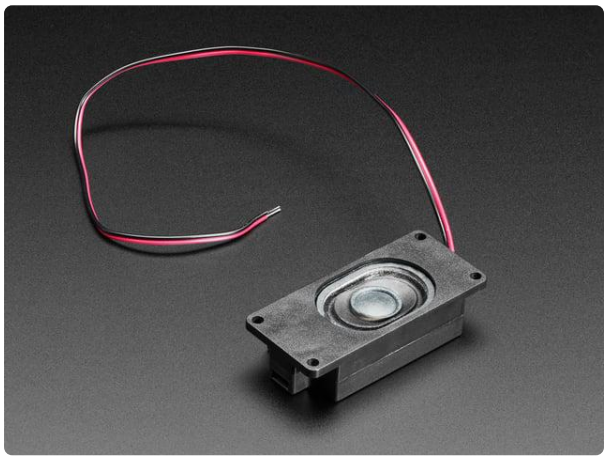
In this section, you'll learn how to use CircuitPython to play different types of audio using I2S, including tones, WAV files and MP3 files.

Necessary Hardware

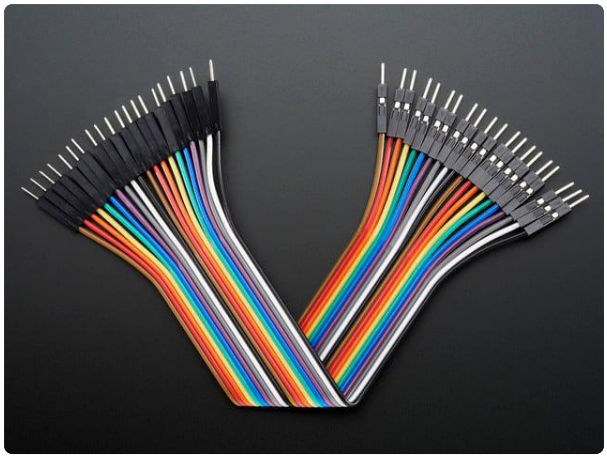
You'll need the following additional hardware to complete the examples on this page.



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



Mono Enclosed Speaker with Plain Wires - 3W 4 Ohm
Listen up! This single 2.8" x 1.2" speaker is the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We...
https://www.adafruit.com/product/4445

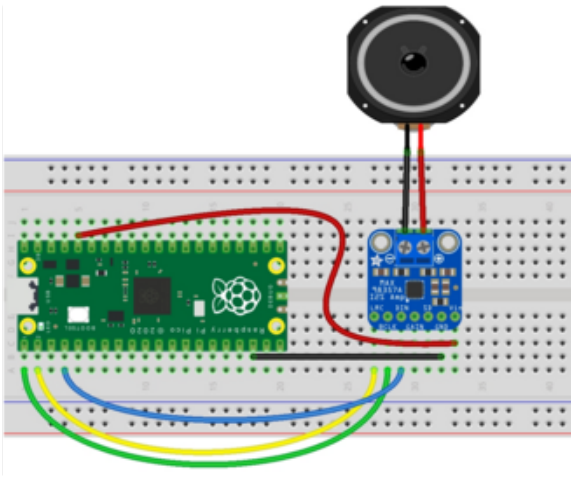


Premium Male/Male Jumper Wires - 20 x 6" (150mm)
These Male/Male Jumper Wires are handy for making wire harnesses or jumpering between headers on PCB's. These premium jumper wires are 6" (150mm) long and come in a...
https://www.adafruit.com/product/1957

Wiring the MAX98357A

Connect the MAX98357A breakout to your microcontroller as follows.

The bit clock and word select pins must be on consecutive pins! They can be on any pins you like, but they must be in consecutive order, for example, A0 for bit clock and A1 for word select.



Pico 3V3 to breakout VIN
Pico GND to breakout GND
Pico GP0 to breakout BCLK
Pico GP1 to breakout LRC
Pico GP2 to breakout DIN
Speaker + to screw terminal +
Speaker - to screw terminal -

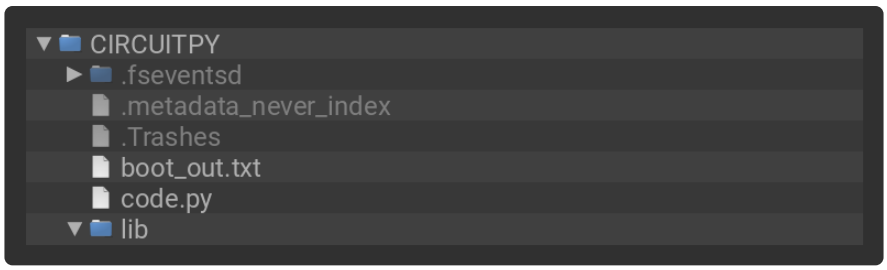
I2S Tone Playback

The first example generates one period of a sine wave and then loops it to generate a tone. You can change the volume and the frequency (in Hz) of the tone by changing the associated variables. Inside the loop, you play the tone for one second and stop it for one second.

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:




```
# SPDX-FileCopyrightText: 2018 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
CircuitPython I2S Tone playback example.
Plays a tone for one second on, one
second off, in a loop.
"""
import time
import array
import math
import audiocore
import board
import audiobusio

audio = audiobusio.I2SOut(board.GP0, board.GP1, board.GP2)

tone_volume = 0.1  # Increase this to increase the volume of the tone.
frequency = 440  # Set this to the Hz of the tone you want to generate.
length = 8000 // frequency
sine_wave = array.array("h", [0] * length)
for i in range(length):
    sine_wave[i] = int((math.sin(math.pi * 2 * i / length)) * tone_volume * (2 ** 15
- 1))
sine_wave_sample = audiocore.RawSample(sine_wave)

while True:
    audio.play(sine_wave_sample, loop=True)
    time.sleep(1)
    audio.stop()
    time.sleep(1)
```

Now you'll hear one second of a 440Hz tone, and one second of silence.

You can try changing the `440` Hz of the tone to produce a tone of a different pitch. Try changing the number of seconds in `time.sleep()` to produce longer or shorter tones.

I2S WAV File Playback

The second example plays a WAV file. You open the file in a readable format. Then, you play the file and, once finished, print `Done playing!` to the serial console. You can use any supported wave file ().

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the StreetChicken.wav file and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:


```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
CircuitPython I2S WAV file playback.
Plays a WAV file once.
"""
import audiocore
import board
import audiobusio

audio = audiobusio.I2SOut(board.GP0, board.GP1, board.GP2)

wave_file = open("StreetChicken.wav", "rb")
wav = audiocore.WaveFile(wave_file)

print("Playing wav file!")
audio.play(wav)
while audio.playing:
    pass
print("Done!")
```

Now you'll hear the wave file play, and on completion, print `Done Playing!` to the serial console.

You can play a different WAV file by updating `"StreetChicken.wav"` to be the name of your CircuitPython-compatible WAV file.

You can do other things while the WAV file plays! There is a `pass` in this example where you can include other code, such as code to blink an LED.

I2S MP3 File Playback

The third example plays an MP3 file. First, you open the file in a readable format. Then you play the MP3 and, once finished, print `Done playing!` to the serial console.

CircuitPython supports any MP3 file, as long as it is the right bit rate and sample rate for your board.

Mono and stereo files less than 64kbit/s work, with sample rates from 8kHz to 24kHz.

You can find an example of converting an MP3 file using Audacity in this guide (). The parameters suggested above may not be exactly what's in the guide, but the concept will be the same.

Update your code.py to the following.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the folder that matches your CircuitPython version, and copy the slow.mp3 file and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:


```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
CircuitPython I2S MP3 playback example.
Plays a single MP3 once.
"""
```

```
import board
import audiomp3
import audiobusio

audio = audiobusio.I2SOut(board.GP0, board.GP1, board.GP2)

mp3 = audiomp3.MP3Decoder(open("slow.mp3", "rb"))

audio.play(mp3)
while audio.playing:
    pass

print("Done playing!")
```

Now you'll hear the MP3 play, and on completion, print `Done Playing!` to the serial console.

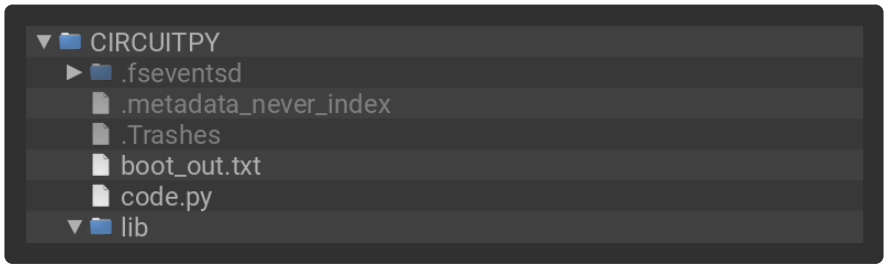
You can play a different CircuitPython-compatible MP3 by updating `"slow.mp3"` to the name of your MP3 file.

CircuitPython I2S-Compatible Pin Combinations

I2S audio is supported on specific pins. The good news is, there's a simple way to find out which pins support audio playback.

In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the directory CircuitPython_Templates/i2s_find_pins/ and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:



Then, connect to the serial console () to see a list of pins printed out. This file runs only once, so if you do not see anything in the output, press CTRL+D to reload and run the code again.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
# SPDX-License-Identifier: MIT
"""
CircuitPython I2S Pin Combination Identification Script
```

```
"""
import board
import audiobusio
from microcontroller import Pin


def is_hardware_i2s(bit_clock, word_select, data):
    try:
        p = audiobusio.I2SOut(bit_clock, word_select, data)
        p.deinit()
        return True
    except ValueError:
        return False


def get_unique_pins():
    exclude = [
        getattr(board, p)
        for p in [
            # This is not an exhaustive list of unexposed pins. Your results
            # may include other pins that you cannot easily connect to.
            "NEOPIXEL",
            "DOTSTAR_CLOCK",
            "DOTSTAR_DATA",
            "APA102_SCK",
            "APA102_MOSI",
            "LED",
            "SWITCH",
            "BUTTON",
        ]
        if p in dir(board)
    ]
    pins = [
        pin
        for pin in [getattr(board, p) for p in dir(board)]
        if isinstance(pin, Pin) and pin not in exclude
    ]
    unique = []
    for p in pins:
        if p not in unique:
            unique.append(p)
    return unique


for bit_clock_pin in get_unique_pins():
    for word_select_pin in get_unique_pins():
        for data_pin in get_unique_pins():
            if bit_clock_pin is word_select_pin or bit_clock_pin is data_pin or
word_select_pin \
                    is data_pin:
                continue
            if is_hardware_i2s(bit_clock_pin, word_select_pin, data_pin):
                print("Bit clock pin:", bit_clock_pin, "\t Word select pin:",
word_select_pin,
                      "\t Data pin:", data_pin)
            else:
                pass
```

For details about the I2S API, check out the CircuitPython docs ().