adafruit learning system

Native MP3 decoding on Arduino

Created by Dean Miller

Last updated on 2018-08-22 04:04:11 PM UTC

Guide Contents

Overview

If you have an ARM Cortex M4 (or M3) based microcontroller board, and you want to rock out, this mini guide will be music to your ears. With the accompanying library, based off of Helix, you will be able to **decode and play stereo MP3 files without the need for an external chip!** That's right, no VLSI VS10xx chips required, you can do it on the fly!

MP3 files are very popular because of their widespread availability and small size. MP3 files are compressed, and can be 75 to 95% smaller in size than their original uncompressed form. This small size makes them great for embedded audio projects where storage can be an issue - WAV files are much easier to decode 'cause there is no compression, but that also made them very big.

Previously, the MP3 format was protected under patent, so anybody who wanted to sell products that used MP3 files would have to pay a licensing fee. Now those patents have expired and MP3 is freeeeee and you can embed the decoder into any project or product! (https://adafru.it/vDJ)

Because of this, we can now show you how to decode and play back MP3 files with just a powerful microcontroller like the Adafruit Metro M4 or Feather M4 (https://adafru.it/A5S), Teensy 3.1, Teensy 3.2 (https://adafru.it/j7a), and Teensy 3.6 (https://adafru.it/A5T)!

This is great news for cosplayers, halloween fans, prop-makers and all other types of DIY enthusiasts who want to add MP3 audio to their projects without the external hardware that was previously required (like the Adafruit Audio FX Sound Board (https://adafru.it/e82)).

TODO

A lot! This code isn't optimized, doesn't use DMA, requires two DACs at this time (doesn't support I2S)... but it is working proof-of-concept!

Compatible Microcontrollers

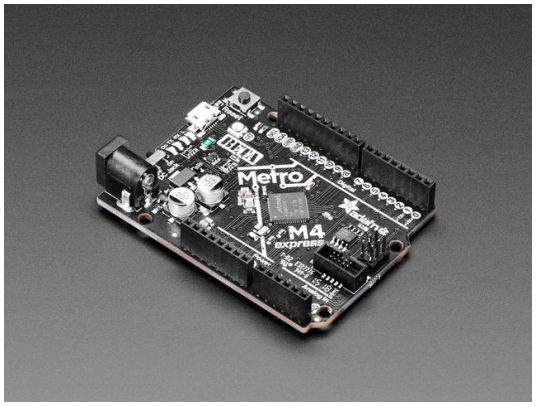
The Adafruit MP3 library is currently compatible with the Adafruit Feather / Metro M4 (https://adafru.it/A5S), Teensy 3.1, Teensy 3.2 (https://adafru.it/j7a), and Teensy 3.6 (https://adafru.it/A5T) only.

The reason we like these is because all boards have a powerful ARM Cortex M4 at the heart and 1 or 2 12 bit DAC outputs (teensy 3.6 and feather/metro M4 have 2). This combo of processing power and 2 analog outputs is perfect for stereo audio processing with decent resolution, and no additional cost. They also have plenty of RAM for buffering

data.

If you have a favorite Cortex M3/M4 chip, and you'd like to add support, please submit a pull request! (https://adafru.it/BtD)(We only had these two boards on-hand.) You'll need about 32KB of RAM available and about 32KB of FLASH.

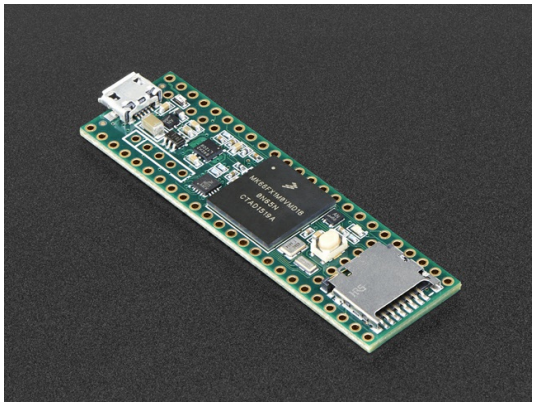
> This code will never work on an 8-bit microcontroller such as the Arduino UNO/Mega. And the 32KB of RAM requirement shuts out many other smaller Cortex chips!



Adafruit Metro M4 feat. Microchip ATSAMD51

$27.50
IN STOCK

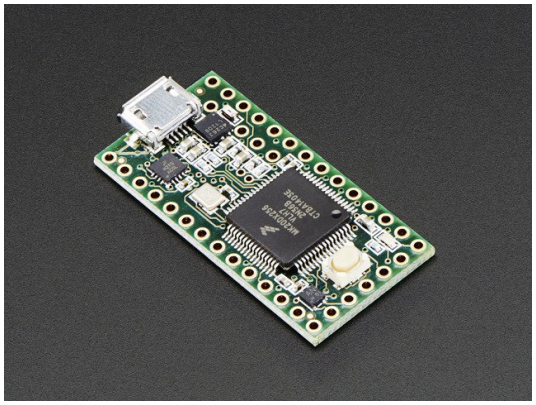
ADD TO CART



Teensy 3.6 without headers

$29.95
IN STOCK

ADD TO CART



Teensy 3.2 + header

$19.95
IN STOCK

ADD TO CART

Basis

The Adafruit_MP3 library is a wrapper around the excellent Helix (https://adafru.it/Cha) mp3 decoder, modified for ease

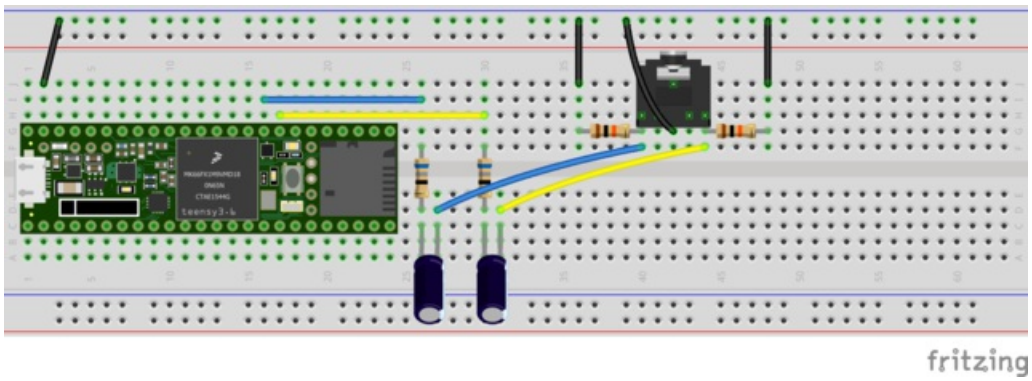
of use.

Arduino Wiring & Test

This example will use the Teensy 3.6 and will read an MP3 file off of an SD card and play it back through the onboard 2-channel DAC. The audio output goes through an impedance matching series resistor and an RC low pass filter to remove the DC offset.

Wiring

- **DAC0**- connect DAC0 on the teensy to the positive (longer) lead of a 10uF electrolytic capacitor through a 68ohm resistor. The resistor value isn't critical, try to pick something in the 47 - 100 ohm range. Connect the negative (shorter) lead of the electrolytic capacitor to the left lead of the headphone jack. Connect a 10k resistor from the left lead of the headphone jack to AGND on the teensy.
- **DAC1** - connect DAC1 on the teensy to the positive (longer) lead of a 10uF electrolytic capacitor through a 68ohm resistor. Again, the resistor value isn't critical, try to pick something in the 47 - 100 ohm range. Connect the negative (shorter) lead of the electrolytic capacitor to the right lead of the headphone jack. Connect a 10k resistor from the right lead of the headphone jack to AGND on the teensy.
- Connect the ground (center) pin on the headphone jack to AGND on the teensy.



Loading the SD card

Rename your MP3 file **test.mp3** and load it onto a standard formatted microSD card.

Insert the microSD card into the SD slot on the teensy.

Download the Library

To begin playing files, you will need to download **Adafruit_MP3** from our github repository. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

https://adafru.it/A5U

https://adafru.it/A5U

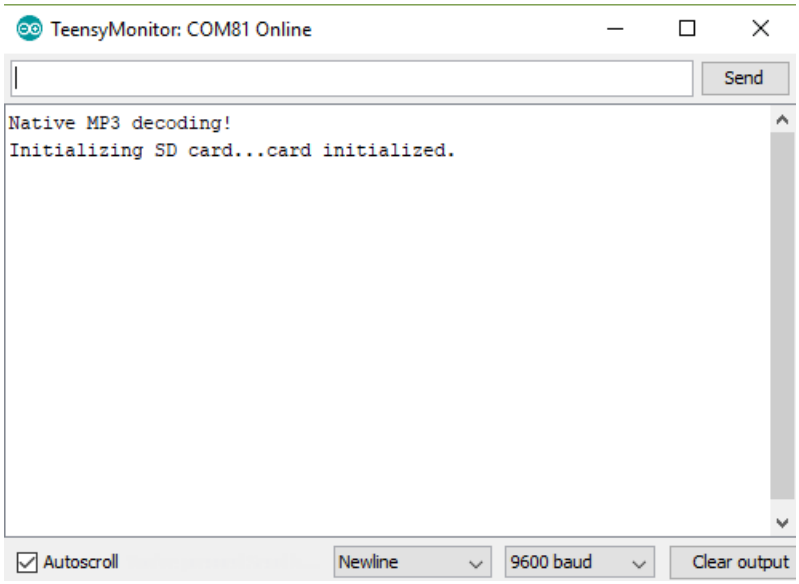
Rename the uncompressed folder **Adafruit_MP3** and check that the **Adafruit_MP3** folder contains **Adafruit_MP3 .cpp** and **Adafruit_MP3 .h**

Place the **Adafruit_MP3** library folder your **arduinosketchfolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

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

Load Play from SD example

Open **File->Examples->Adafruit_MP3->play_from_SD** and upload to your Teensy. Open up the serial console and make sure it is able to read the SD card.



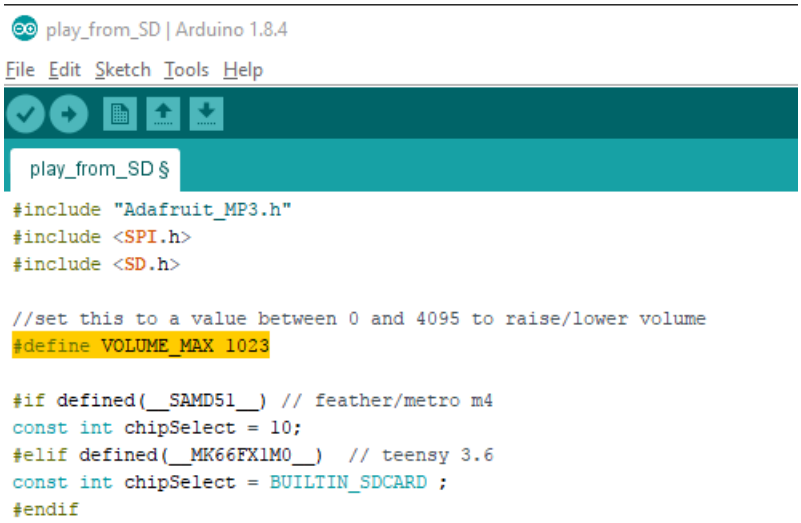
You should hear the file play through whatever is plugged into the 3.5mm jack.

NOTE: if playback of your file is choppy, try a different file with a lower bitrate.

Adjusting the Volume

The DAC on the Teensy and Feather/Metro M4 has 12 bit resolution. This means the output can have 4096 different values ranging from 0 (ground) to 4095 (3.3V). In this simple example we can raise or lower the volume by changing the mapping of the value that is written to the DAC.

Change VOLUME_MAX in the sketch to a desired value between 0 (inaudible) and 4095 (very loud).




```
#include "Adafruit_MP3.h"
#include <SPI.h>
#include <SD.h>

//set this to a value between 0 and 4095 to raise/lower volume
#define VOLUME_MAX 1023

#if defined(__SAMD51__) // feather/metro m4
const int chipSelect = 10;
#elif defined(__MK66FX1M0__)  // teensy 3.6
const int chipSelect = BUILTIN_SDCARD ;
#endif
```

How it Works

The theory of operation of the Adafruit MP3 library is simple. After instantiating your Adafruit_MP3 object, call player.begin() from your setup function. This does necessary timer and buffer setup.

```
#include "Adafruit_MP3.h"

Adafruit_MP3 player;

void setup() {
  player.begin();
```

The library uses a timer set to fire whenever a new sample should be played. For most MP3 files, this is 44.1 kHz. When this timer fires, a user-specified function is called. This function should write the samples to the DAC. This function is specified with

```
player.setSampleReadyCallback(writeDacs);
```

Where writeDacs is a function:

```
void writeDacs(int16_t l, int16_t r){
  /* TODO: map the 16 bit values to the DAC resolution
   * and write to the left and right channel DACs
   */
}
```

This function is called from an interrupt, so it should be short and sweet. It's getting called 44,100 to 48,000 times per second for most MP3 files.

Another callback needs to be specified with instructions for loading the MP3 data into the players input buffer. This is specified with

```
player.setBufferCallback(getMoreData);
```

where getMoreData is a function:

```
int getMoreData(uint8_t *writeHere, int available){
  /* TODO: write up to 'available' bytes of MP3 data
   * to the location 'writeHere'. Return the actual number of bytes
   * that were written
   */
}
```

This function should not take too long (especially if it needs to disable interrupts on the microcontroller) to avoid choppy playback.

Once these two callbacks are specified, the 'tick()' function needs to be called as fast as possible from within your 'loop()' function.

```
void loop() {
  player.tick();
}
```

The addition of DMA support would be awesome, we have not done that yet :)

Porting to different microcontrollers

To port the Adafruit MP3 library to a different microcontroller, we first need to make sure the helix player can support it. Open the file `mp3dec.h` and if necessary add an `#elif` to the switch at the top of the file for your platform.

Next open the file `assembly.h` and if necessary add an `#elif` with some assembly code for the required functions:

```
static __inline int MULSHIFT32(int x, int y);
static __inline int FASTABS(int x);
static __inline int CLZ(int x);
static __inline Word64 MADD64(Word64 sum64, int x, int y);
static __inline Word64 SAR64(Word64 x, int n);
```

where Word64 is:

```
typedef signed long long int    Word64;  // 64-bit signed integer.
```

Adafruit_MP3 class modifications

The Adafruit_MP3 library uses a timer to control playback. This timer should be able to fire at least 44,100 times per second.

Because different microcontroller platforms implement timers differently, you will need to add an `#elif` to the timer functions in `Adafruit_MP3.cpp` with instructions on how to configure, enable, and disable timers on your microcontroller platform.

These functions are:

```
//enable the playback timer
static inline void enableTimer();

//disable the playback timer
static inline void disableTimer();

//configure the playback timer. Make sure to enable the interrupt.
static inline void configureTimer();

//if necessary, clear the interrupt flag.
static inline void acknowledgeInterrupt();
```

You may also need to redefine the name of the interrupt handler that the timer triggers. This is done in `Adafruit_MP3.h`. For example, on the SAMD51 (the chip that the Metro / Feather M4 uses), this is done with:

```
#define MP3_Handler TC2_Handler
```