adafruit learning system

Adafruit Music Maker Shield

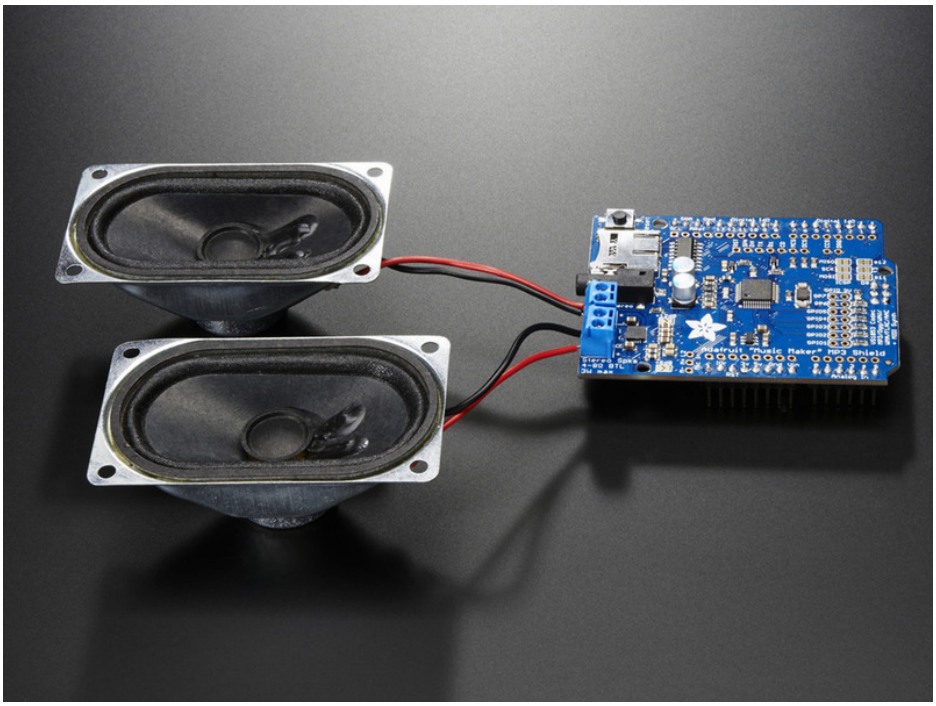
Created by lady ada

Last updated on 2019-10-23 11:33:38 PM UTC

Overview

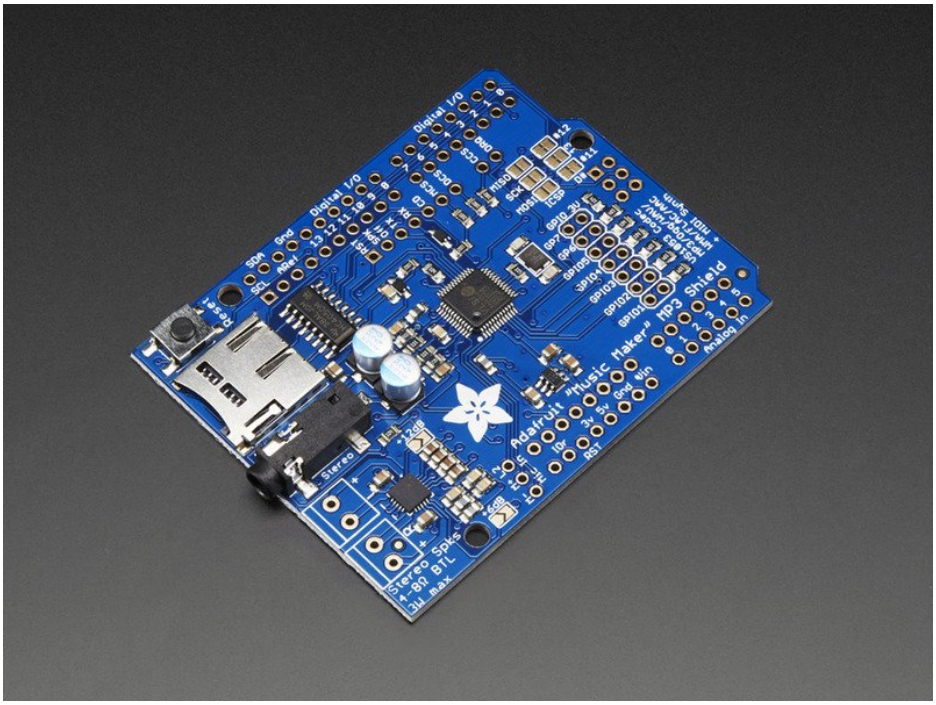
Bend all audio files to your will with the Adafruit Music Maker shield for Arduino! This powerful shield features the VS1053, an encoding/decoding (codec) chip that can decode a wide variety of audio formats such as MP3, AAC, Ogg Vorbis, WMA, MIDI, FLAC, WAV (PCM and ADPCM). It can also be used to record audio in both PCM (WAV) and compressed Ogg Vorbis. You can do all sorts of stuff with the audio as well such as adjusting bass, treble, and volume digitally.

All this functionality is implemented in a light-weight SPI interface so that any Arduino can play audio from an SD card. There's also a special MIDI mode that you can boot the chip into that will read 'classic' 31250Kbaud MIDI data from an Arduino pin and act like a synth/drum machine - there are dozens of built-in drum and sample effects! But the chip is a pain to solder, and needs a lot of extras. That's why we spun up the best shield, perfect for use with any Arduino Uno, Leonardo or Mega.



We have two versions if the shield. One version comes with an onboard 3W stereo amplifier so you can play amplified music with just some speakers. Another version comes without the amplifier, for cost-conscious projects.

Both use the same code, are the same shape, and have Stereo Headphone/Line Out for connecting to a headset or amplifier.

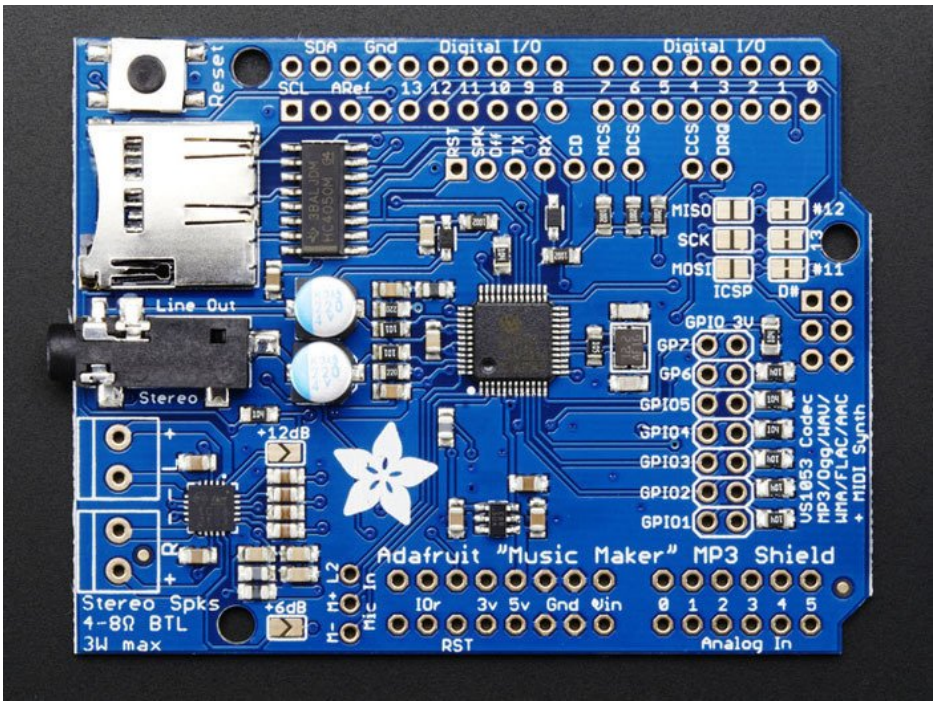


We believe this is the best MP3 playing shield you can get, and at a great price too. Here are some specs:

- Features the VS1053B codec chip - decodes Ogg Vorbis, MP3/MP2/MP1, MP4, AAC, WMA, FLAC, WAV/PCM, MIDI. Encodes Ogg or WAV/PCM

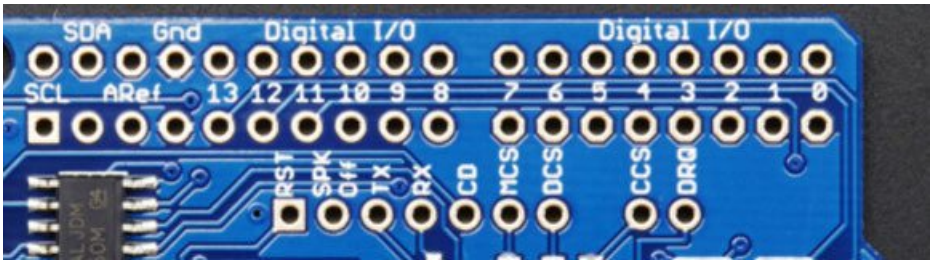
- Stereo audio out with proper audio filter caps and ground reference so it can be safely connected directly to headphones, a stereo system or other powered speakers
- 7 extra GPIO's that can be written or read through the Arduino Library for reading buttons or lighting LEDs
- MicroSD card socket, for any FAT16/FAT32 formatted SD card from 64Mb or greater.
- Full 3.3/5V level shifting for SD and MP3 chipsets
- Works with Arduino Uno, Mega, or Leonardo
- Built in MIDI synth/drum machine with dozens of instruments
- Plenty of optional breakouts for pins like the card-detect and microphone input

Pinouts



There's a lot of stuff going on in this shield! Lets look at the board and all the pinouts. The amplifier version and non-amplifier version both use the same PCB so the pinouts are the same. We'll talk about just the amplifier separately at the bottom

Main Control Breakouts



The Music Maker shield has a bunch of pins required for use. We pre-wire all of them for you but there's still some flexibility in case you want to rewire.

There are three 'totally fixed' pins, the hardware SPI pins:

- **SPI SCK** - connected to Digital #**13** (but can be connected to the ISP header with a jumper) - used by both the SD card and VS1053
- **SPI MISO** - connected to Digital #**12** (but can be connected to the ISP header with a jumper) - used by both the SD card and VS1053
- **SPI MOSI** - connected to Digital #**11** (but can be connected to the ISP header with a jumper) - used by both the SD card and VS1053

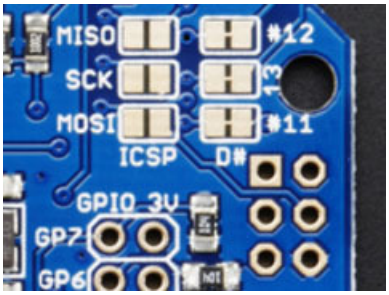
There are a couple other pins that are required for talking to the VS1053 to play MP3s and such

- **MCS** - this is the VS1053 chip select pin, connected to Digital **#7**
- **DCS** - this is the VS1053 data select pin, connected to Digital **#6**
- **CCS** - this is the SD Card chip select pin, connected to Digital **#4**
- **DREQ** - this is the VS1053 data request interrupt pin - connected to digital **#3**

Ther are also a few other pins that are not connected to any Arduino pin but are broken out:

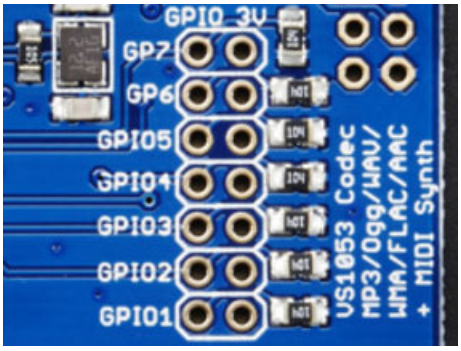
- **RST** - this is the VS1053 reset pin, we connected it to the Arduino reset pin so you don't need to use this unless you really want to.
- **SPK Off** - this disables the amplifier - if you have the amplifier version and want to 'mute' instantly
- **TX** - this is serial data transmit from the VS1053 - its not used for any of our demos
- **RS** - this is serial data into the VS1053 - its used for MIDI synth playing
- **CD** - this is the card detect pin, it is tied to ground when a card is inserted. Use a pullup on a digital pin to detect when a SD card is inserted. We dont use it.

SPI Jumpers



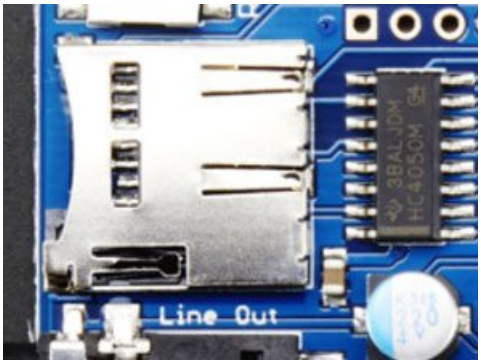
If you're using a Mega or Leonardo Arduino, you'll need to short the jumpers on the top right of the board, and install the 2x3 socket header. This is because those Arduinos use the 2x3 pin header for the hardware-SPI pins and hardware-SPI is required for the high-speed data transfer required by the VS1053 codec. We'll cover that in the Assembly step.

GPIO Breakouts



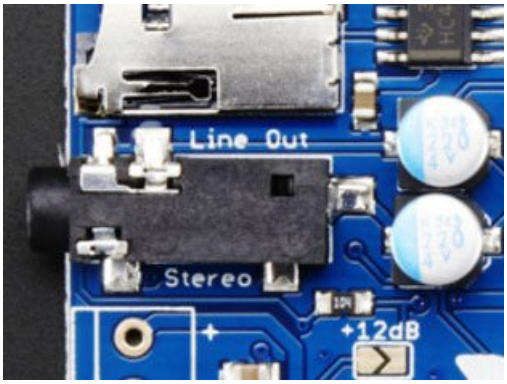
The VS1053 codec chip has 7 'General Purpose Input/Output' pins that you can use to detect button presses and/or light up a small LED. GPIO1 is also used to put it into MIDI mode. By default all these pins are pulled **low** to ground with 100K resistors. They are 3V logic level so if you want to attach a button, you can connect the two wires between the GPIO pin on the left and the 3V breakout on the right. If you don't want to use these pins just leave them be, they are not required for use!

MicroSD Card Socket



In order to play MP3, WAV, OGG, etc files, you'll need to store them on a MicroSD card. These are very very common, available in the Adafruit shop or any electronics store. You can use any FAT16/FAT32 formatted card from 64M up to 8G. Chances are its pre-formatted for this so you can just drop files on. This is a push-push socket, push the card in once to seat in, push again to pop out. The chip to the right is the level shifter to make it safe to use with 5V logic like Arduino

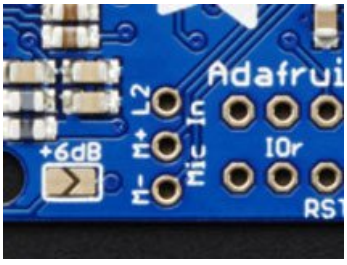
Line Out



For almost all purposes, the Stereo Line Out 3.5mm jack is what you'll want to get audio out of the shield. It's there on both versions of the shield. The two big silver capacitors on the right are DC blocking caps. This means that the audio is AC-coupled and is safe to use with any amplifier, headphone. etc. Line level is up to about 2V peak-to-peak. If you have a system that really needs 0.7Vpp or less, set the volume on the VS1053 in software.

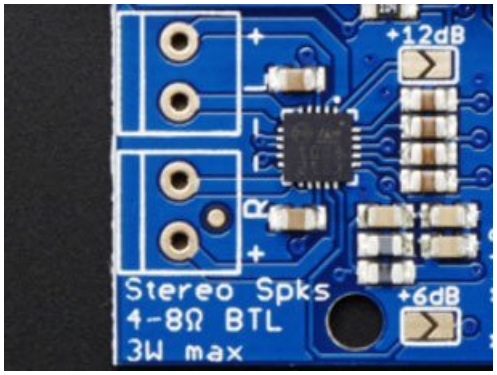
You can also plug in headphones, although it is not a very strong headphone driver, so may not sound loud if its lower than 32Ω impedance

Microphone In



We break out the microphone/line in inputs, for recording audio. Some basic analog filtering is required depending on the electret microphone or amplifier. Check the VS1053 datasheet for how to connect up a mic!

Amplifier Section



In the bottom left is an amplifier section, this is a stereo 3W amplifier, with bridge-tied load output. **It's only meant for driving speakers directly! Do not connect to another amplifier, use the line out for that! Also, you cannot bridge-tie R and L together -** if you need only one speaker, leave the unused one disconnected.

For the amplifier, we're using the TS2012 class D chipset, the same used in this amplifier board (https://adafru.it/dst)

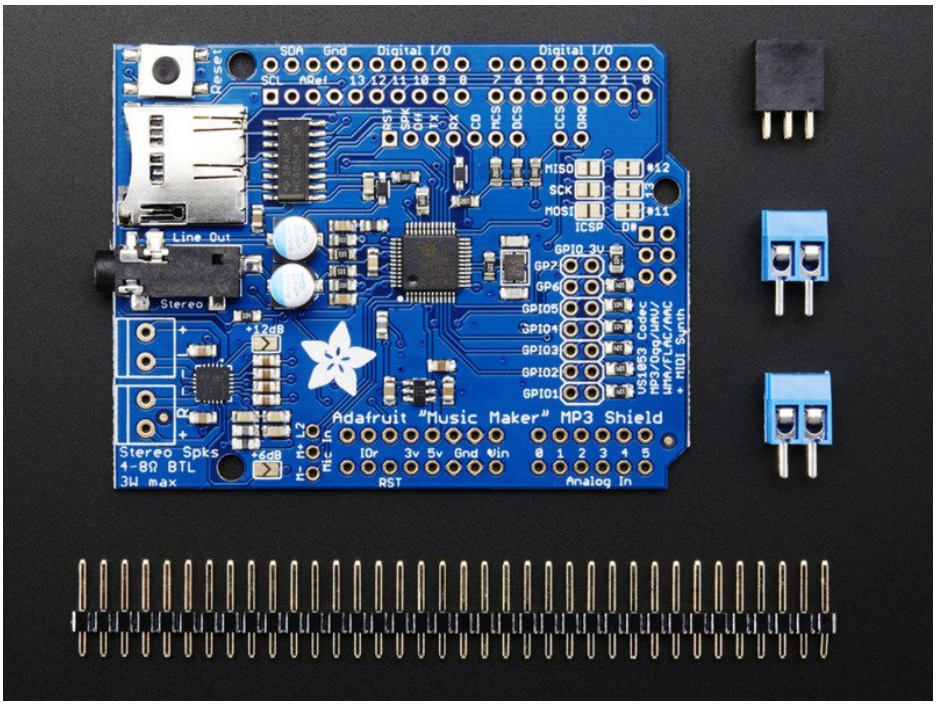
Speaker connects

On the left are two sets of breakouts for Right and Left. Connect these directly to your 4 or 8Ω speakers. Ideally they are 4Ω 3W speakers or 8Ω 1W speakers. You will get louder audio with 4Ω speakers since the amplifier voltage is maxed out at 5V from the Arduino.
If you have both speakers attached and you're playing loud audio, you may need to power the Arduino from DC power jack instead of USB since USB can only provide 5W (5V @ 1A) max and two 3W speakers = 6W!

+dB jumpers

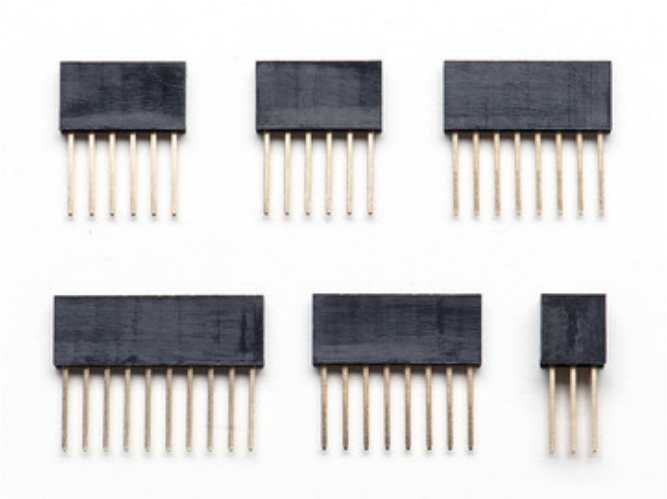
The default amplification for the speakers is +6dB. This gives nice unclipped audio from the VS1053 even at highest volume. If by chance you can't amplify the audio (its not normalized right) or there's some other reason to need a higher amplification, you can short the +6dB or +12dB jumpers to increase the gain. **Don't do this unless you're really sure!** You can end up with really heavy clipping which sounds bad!

Assembly



Stack Alert

If you want to stack a shield on top of the Music Maker, you'll want to pick up some stacking headers and use those instead of the plain header shown here!

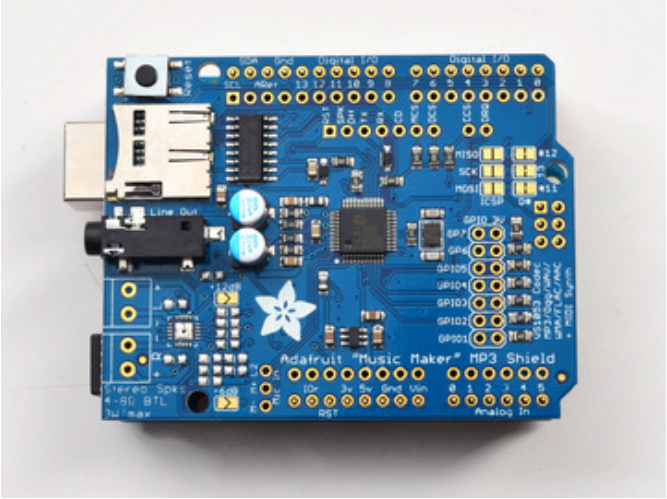


Wanna stack? This tutorial shows how to use the plain header to connect to an Arduino. If you want to use stacking headers (https://adafru.it/dsu), don't follow these steps!

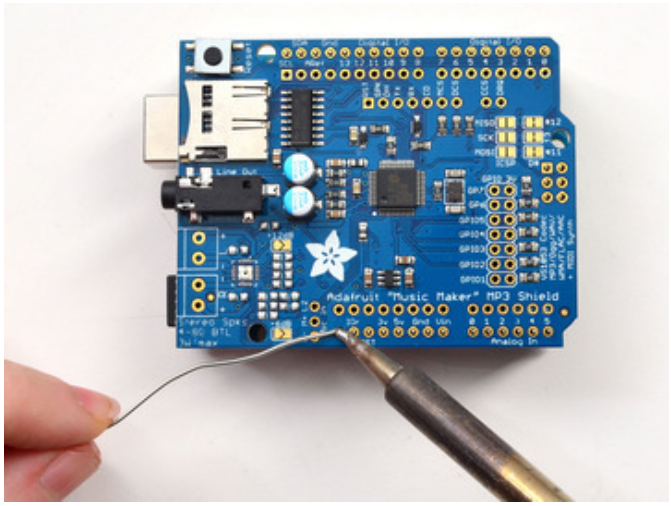
Attaching Headers (All Arduinos)

Begin by breaking the 36-pin male header into four pieces: one 10-pin, two 8-pin and one 6-pin. Stick the header into the Arduino sockets with the long pins down.

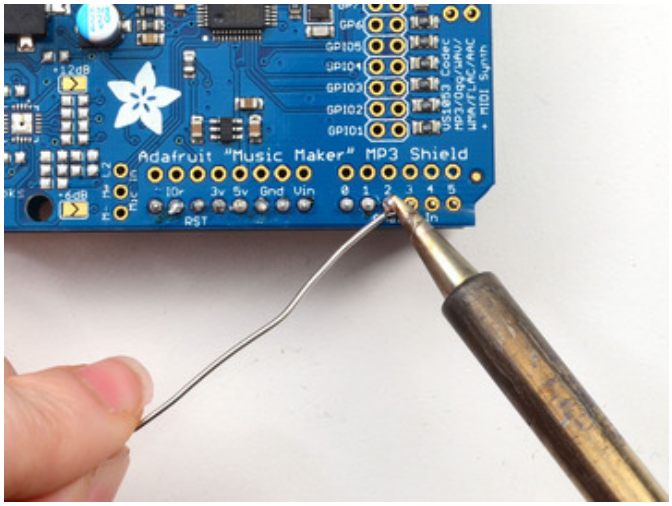
Also place the 2x3 female socket header into the ICSP header on the right of the board

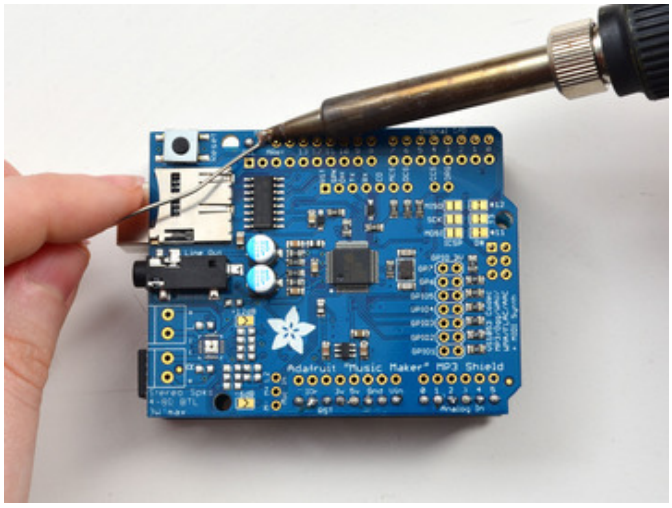


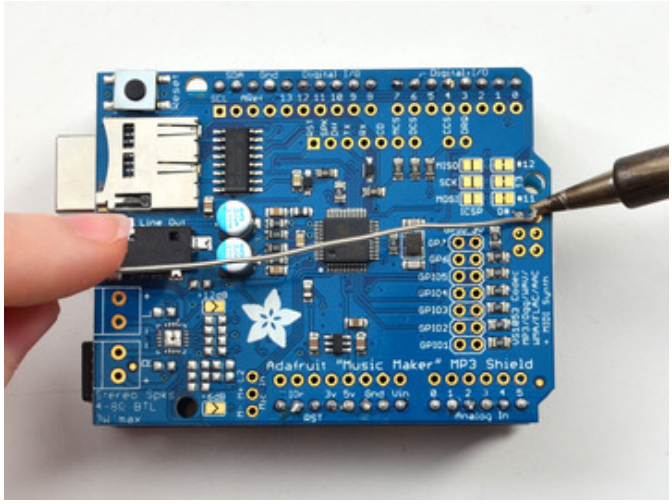
Place the shield on top so that all the little pins stick out through the matching holes in the shield. It should match up perfectly!



Solder in all the header



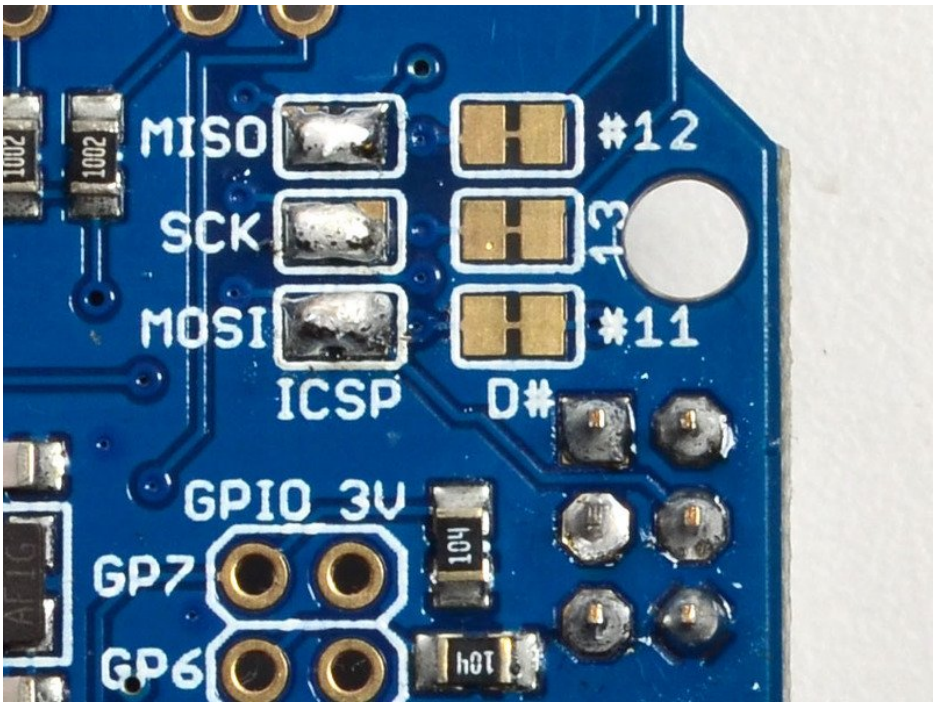




Don't forget the 6-pin socket!

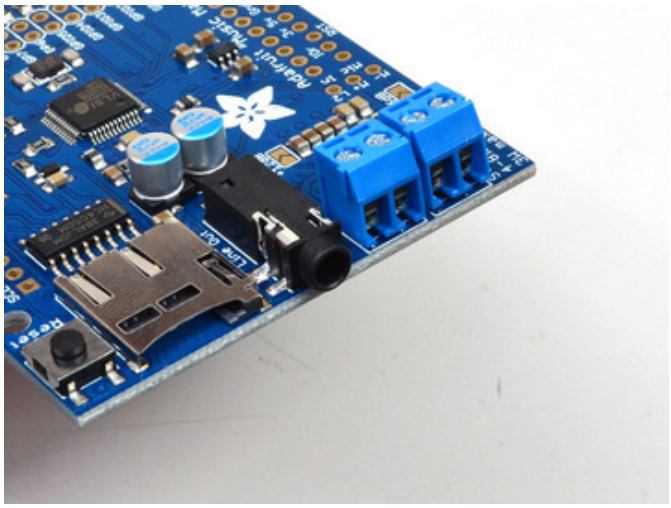
ICSP Jumpers (Leonardo & Mega)

If you have a Leonardo or Mega, you'll need to close the three solder jumpers next to the ISP header. This configures the shield to use the ISP header for SPI communication. Its easy! Simply melt some solder to close the three jumpers.

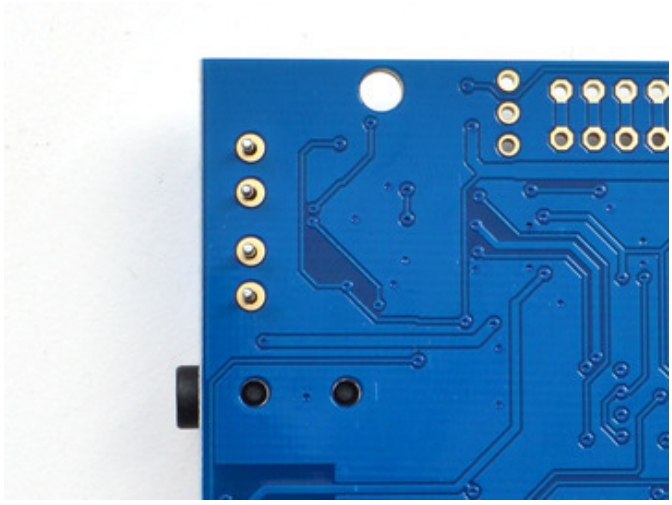


You can also cut the mini wire between the other three jumpers to 'release' the digital #11, 12 and 13 pins from being tied to the SPI pins. You can do this after you've gotten things working.

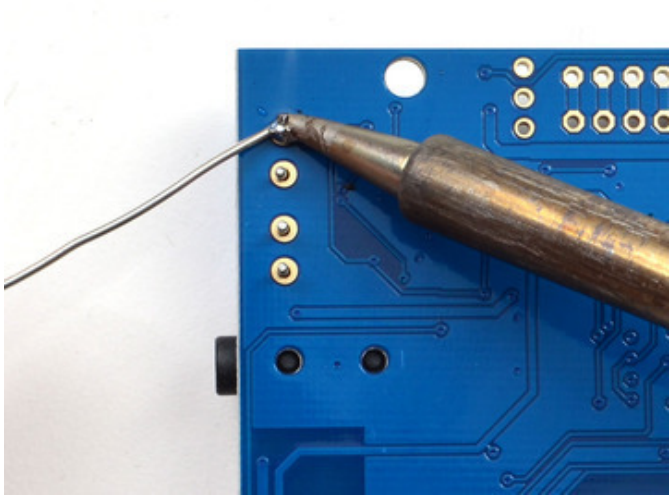
If you have the Amplified version....



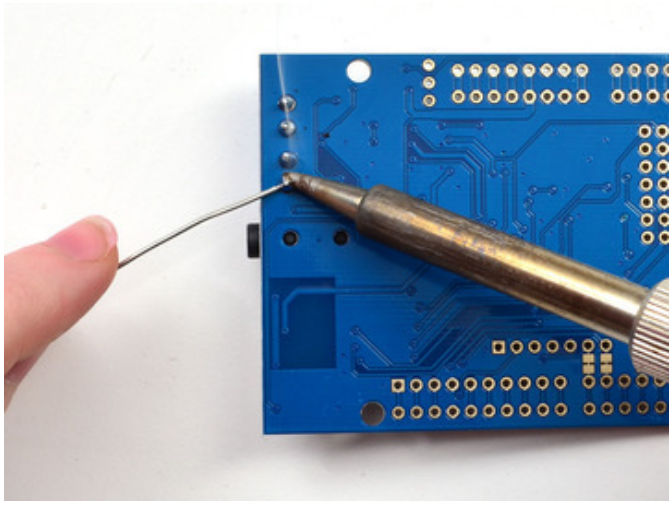
Place the two blue terminal blocks in the slots next to the headphone jack



Flip the board over, you can even have it on a table or use scotch tape to keep the terminal blocks in place



Solder the four pins with plenty of solder

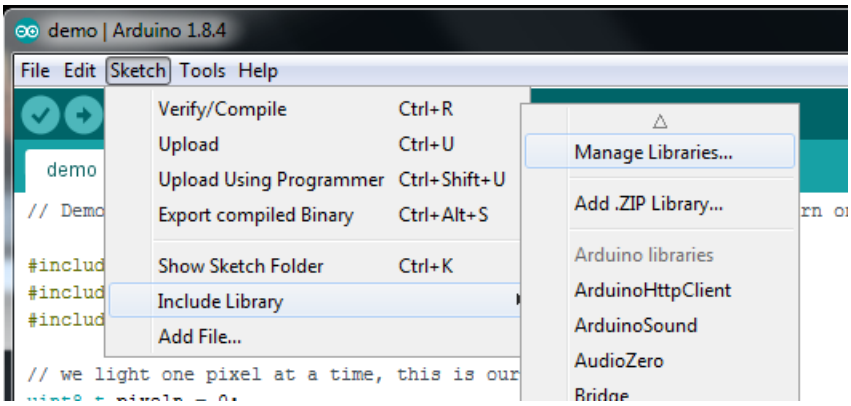


To keep the rightmost speaker pin from bumping into the DC jack you may need to clip it with diagonal cutters

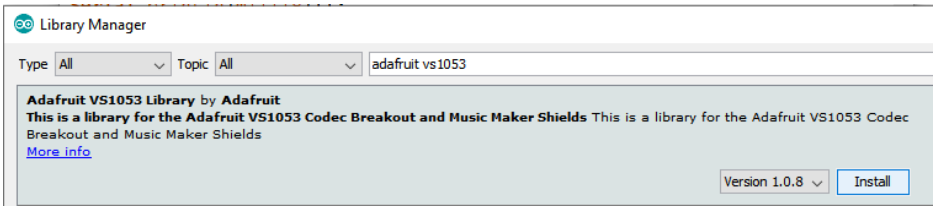
Installing Software

To get started with the Music Maker, you'll need to control the built-in VS1053 chip by installing the Adafruit_VS1053 library (https://adafru.it/cIE)

Open up the Arduino library manager:



Search for **Adafruit_VS1053** library and install it

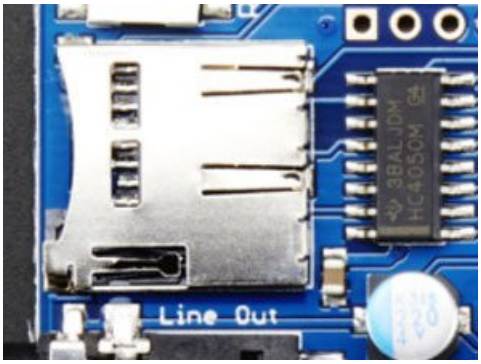


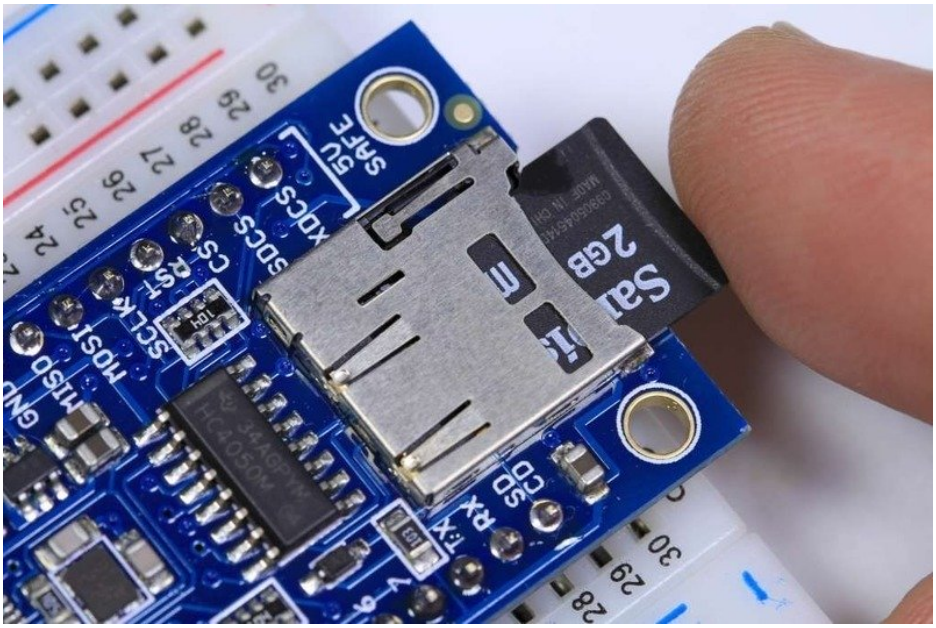
Play Music

Load some MP3 files

Copy 2 MP3 files to a micro SD card and name them **track001.mp3** and **track002.mp3** (this is just for the test, you can re-name them later). Then push the uSD card into the slot on the shield

The SD library for Arduino can only handle 8.3 names, that means you can name your file track001.mp3 (8 letters dot 3 letters) but not MyFavoriteMusic.mp3

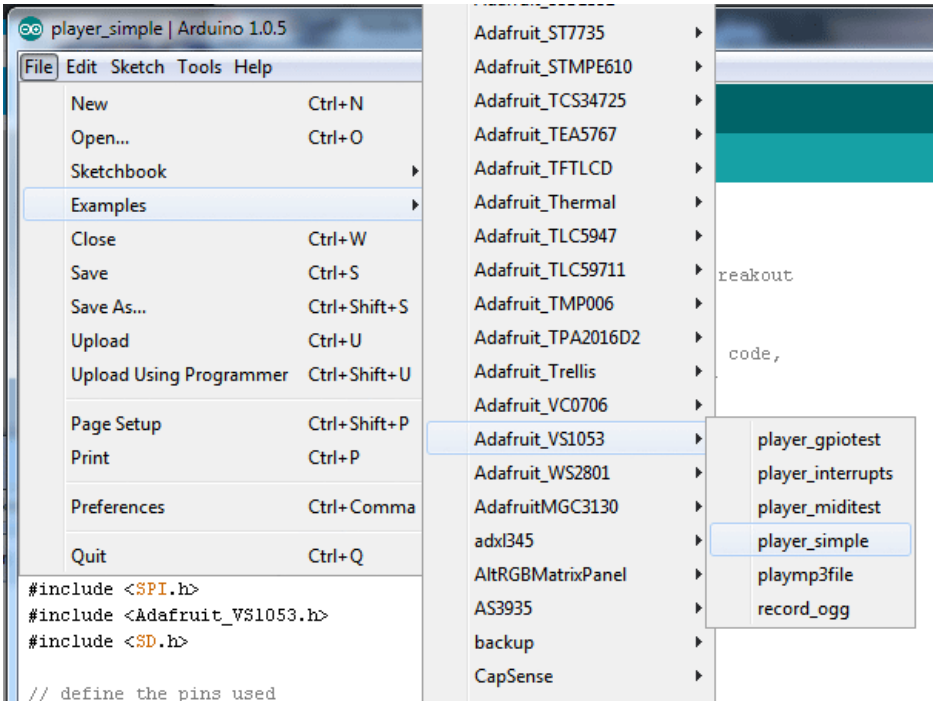




Make sure you have a good quality SD card, some cheap SD cards won't work, causing confusion! Especially 'non-brand' knockoffs.

Simple Audio Player Sketch

Connect the Arduino to your computer with a USB cable and plug your headphones into the headphone jack. Select **File->Examples->Adafruit_VS1053->player_simple** to load the example code.



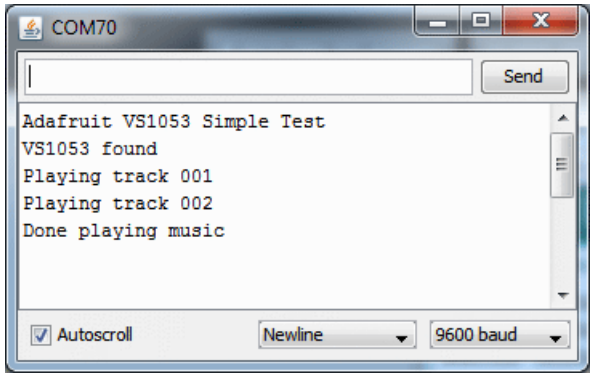
We originally wrote the library for use with the Breakout board. Since the pinout's a little different we just need to make a minor change. Find this line:

```
Adafruit_VS1053_FilePlayer musicPlayer =
  // create breakout-example object!
  Adafruit_VS1053_FilePlayer(BREAKOUT_RESET, BREAKOUT_CS, BREAKOUT_DCS, DREQ, CARDCS);
  // create shield-example object!
  //Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);
```

and change it to:

```
Adafruit_VS1053_FilePlayer musicPlayer =
  // create breakout-example object!
  //Adafruit_VS1053_FilePlayer(BREAKOUT_RESET, BREAKOUT_CS, BREAKOUT_DCS, DREQ, CARDCS);
  // create shield-example object!
  Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);
```

To use the shield pinouts. Now upload the example. You should see the following:



And audio playing from the headphone jack.
If you get

> Adafruit VS1053 Simple Test
> Couldn't find VS1053, do you have the right pins defined?

Check that you commented out the breakout line and uncommented the shield line so it knows you're using a shield!

Interrupt/Background Version

Advanced users can also run **File->Examples->Adafruit_VS1053->player_interrupts.** This example demonstrates playing files in the background using interrupts. This allows you to do other things in your sketch while the music plays! It also has more error reporting and lists all the files found in the SD card

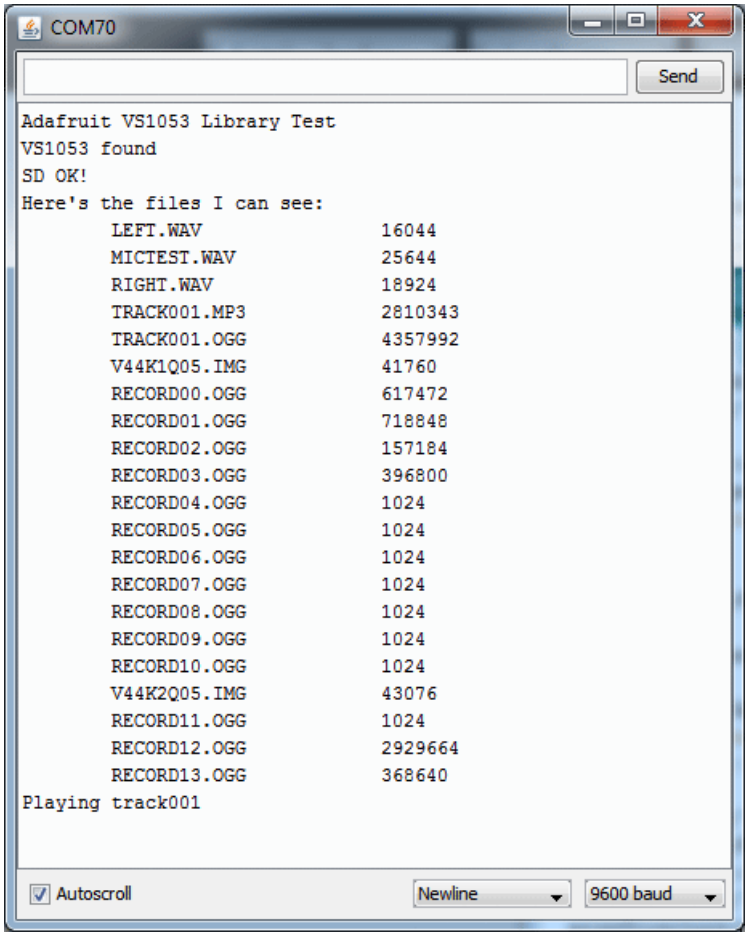
Don't forget to do the same thing, updating the:

```
Adafruit_VS1053_FilePlayer musicPlayer =
  // create breakout-example object!
  Adafruit_VS1053_FilePlayer(BREAKOUT_RESET, BREAKOUT_CS, BREAKOUT_DCS, DREQ, CARDCS);
  // create shield-example object!
  //Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);
```

to:

```
Adafruit_VS1053_FilePlayer musicPlayer =
  // create breakout-example object!
  //Adafruit_VS1053_FilePlayer(BREAKOUT_RESET, BREAKOUT_CS, BREAKOUT_DCS, DREQ, CARDCS);
  // create shield-example object!
  Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);
```

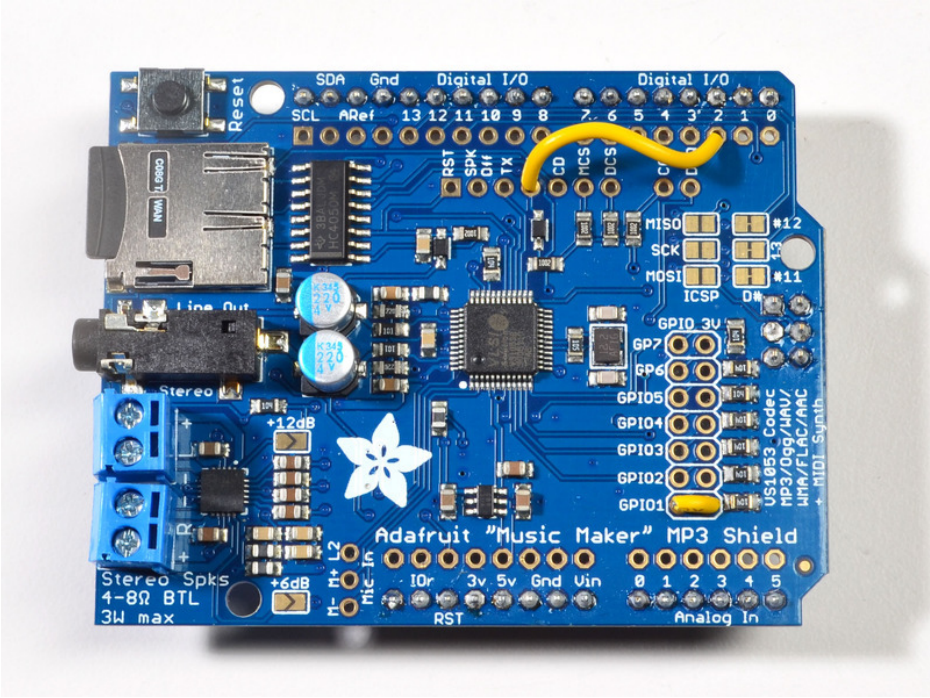
You can see the output is more detailed - it also lists the names of the cards on the SD which can help if you're having problems naming the file:


COM70
Send
Adafruit VS1053 Library Test
VS1053 found
SD OK!
Here's the files I can see:
LEFT.WAV 16044
MICTEST.WAV 25644
RIGHT.WAV 18924
TRACK001.MP3 2810343
TRACK001.OGG 4357992
V44K1Q05.IMG 41760
RECORD00.OGG 617472
RECORD01.OGG 718848
RECORD02.OGG 157184
RECORD03.OGG 396800
RECORD04.OGG 1024
RECORD05.OGG 1024
RECORD06.OGG 1024
RECORD07.OGG 1024
RECORD08.OGG 1024
RECORD09.OGG 1024
RECORD10.OGG 1024
V44K2Q05.IMG 43076
RECORD11.OGG 1024
RECORD12.OGG 2929664
RECORD13.OGG 368640
Playing track001
Autoscroll
Newline
9600 baud

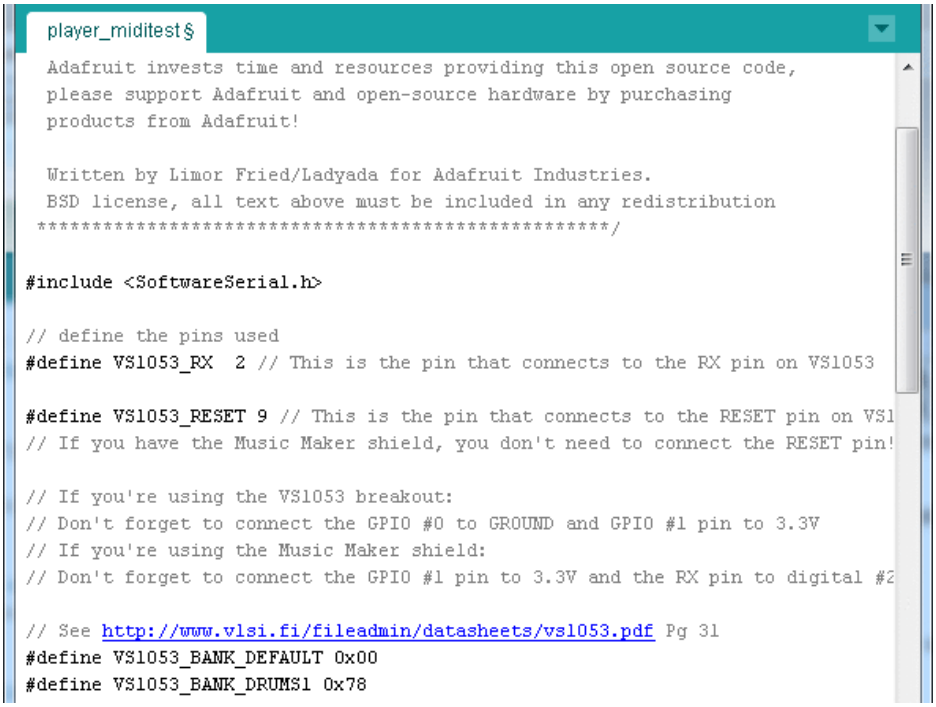
MIDI Synth

With a few jumper connections, the board will boot up in MIDI mode that will read 'classic' 31250Kbaud MIDI data on a UART pin and act like a synth/drum machine - there are dozens of built-in drum and sample effects.

By default, MIDI mode is not 'activated' - but its very easy to turn on. Start by soldering a jumper wire between **GPIO1** pin and **3V** on the shield and a wire from **Digital #2** to the **RX** pin on the shield, see the two yellow wires here:



Now run **File->Examples->Adafruit_VS1053->player_miditest**



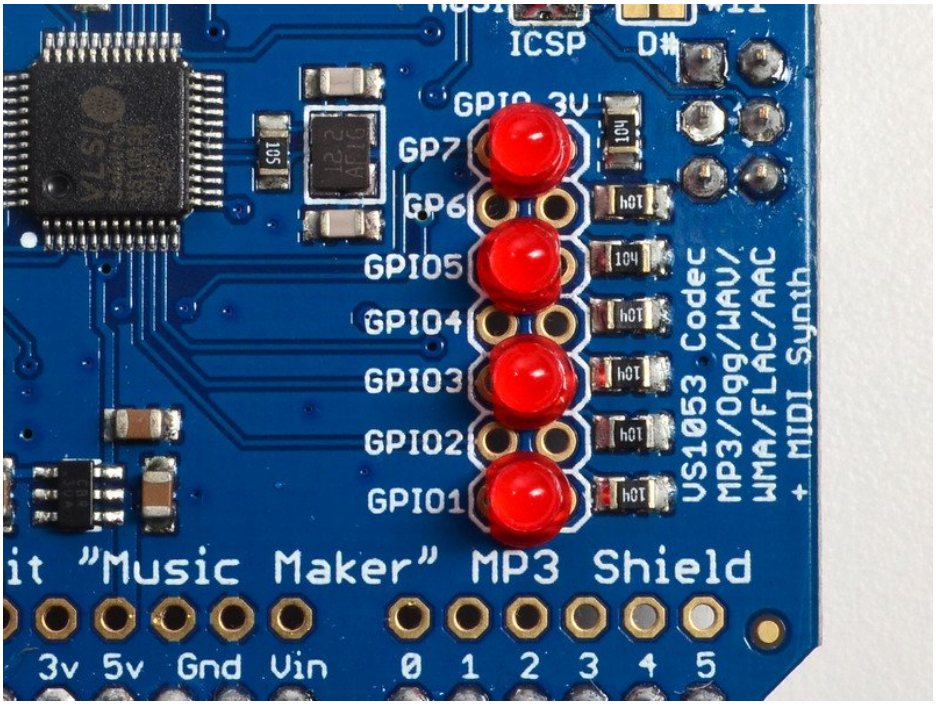
Upload to the Arduino + Shield and listen on the headphone jack for the Ocarina scale being played. You can check the datasheet for a list of all the instruments (there's a lot!)

GPIO Pins

The VS1053 has 7 GPIO pins that can be read and written via the library. The player_gpiotest sketch demonstrates how to do this. Be careful about pulling up GPIO1 - if the shield restarts when GPIO1 is connected to 3V logic, it will boot into 'MIDI' mode

The 7 GPIOs are by default pulled low with 100K resistors, and can only take up to 3V logic!

We can quickly demo the shield by slipping 3mm LEDs into alternating slots. Connect the positive (anode) to the 3V side of the dual strip



What? No current limiting resistors?

Strictly speaking, best practice is to use a current limiting resistor when driving an LED from a GPIO pin. In this case, the example sketch pulses each led only briefly, so there is no danger of damage. For more general use, you should select a resistor appropriate for the led you are using. See All About LEDs (https://adafru.it/clH) for more detail.

Run the player_gpiotest sketch

Connect the Arduino to your computer with a USB cable. Select **File->Examples->Adafruit_VS1053->player_gpiotest** to load the example code.

Don't forget to uncomment the

```
Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);
```

line just like you did with the other examples.

If you have headphones, you will hear a beep at the start to indicate that the sketch is running. Then you should see

the LEDs flashed in sequence.

If you open the Serial Monitor, you can see the values that are written to and read from each GPIO pin.

class Adafruit_VS1053_FilePlayer

The Adafruit_VS1053_FilePlayer class is derived from the Adafruit_VS1053 class and provides high level functions for playing files stored on the VS1053 breakout SD Card reader.

Public Methods:

Adafruit_VS1053_FilePlayer (int8_t mosi, int8_t miso, int8_t clk, int8_t rst, int8_t cs, int8_t dcs, int8_t dreq, int8_t cardCS) - Software SPI constructor. Uses Software SPI, so you must specify all SPI pins.

Adafruit_VS1053_FilePlayer (int8_t rst, int8_t cs, int8_t dcs, int8_t dreq, int8_t cardCS) - Hardware SPI constructor. Uses Hardware SPI and assumes the default SPI pins. This is what you'll likely use if you're using the shield.

boolean begin(void) - Initialize communication and reset the chip. Returns true if a VS1053 is found

boolean useInterrupt(uint8_t type) - Specifies the interrupt to use for interrupt-driven playback. Valid arguments are:

- VS1053_FILEPLAYER_TIMER0_INT
- VS1053_FILEPLAYER_PIN_INT

boolean startPlayingFile(char *trackname) - Begin playing the specified file from the SD card using interrupt-driven playback. This allows your program to perform other tasks as the file is playing.

boolean playFullFile(char *trackname) - Play the complete file. This function will not return until the playback is complete.

Public Member Variables:

File currentTrack - File currently being played

boolean playingMusic - True if playback in progress

class Adafruit_VS1053

The Adafruit_VS1053 class implements an interface to the basic VS1053 functionality. For more detail on the operation of the VS1053 chip, please refer to the documentation on the Downloads page (see the link to the left). Its a little more powerful but it's also harder to use. We suggest sticking to the FilePlayer class which abstracts a lot of this out for you

public Methods:

Adafruit_VS1053(uint8_t mosi, uint8_t miso, uint8_t clk, uint8_t rst, uint8_t cs, uint8_t dcs, uint8_t dreq) - Software SPI constructor - must specify all pins.

Adafruit_VS1053(uint8_t rst, uint8_t cs, uint8_t dcs, uint8_t dreq) - Hardware SPI constructor - assumes hardware SPI pins.

uint8_t begin(void) - Initialize SPI communication and (hard) reset the chip.

void reset(void) - Performs a hard reset of the chip.

void softReset(void) - Attempts a soft reset of the chip.

uint16_t sciRead(uint8_t addr) - Reads from the specified register on the chip.

void sciWrite(uint8_t addr, uint16_t data) - Writes to the specified register on the chip.

void sineTest(uint8_t n, uint16_t ms) - Generate a sine-wave test signal.

void spiwrite(uint8_t d) - Low-level SPI write operation.

uint8_t spiread(void) - Low-level SPI read operation.

uint16_t decodeTime(void) - Reads the DECODETIME register from the chip.

void setVolume(uint8_t left, uint8_t right) - Set the output volume for the chip.

void dumpRegs(void) - Prints the contents of the MODE, STATUS, CLOCKF and VOLUME registers.

void playData(uint8_t *buffer, uint8_t buffsiz) - Decode and play the contents of the supplied buffer.

boolean readyForData(void) - Test if ready for more data.

void applyPatch(const uint16_t *patch, uint16_t patchsize) - Apply a code patch (See datasheet for details).

uint16_t loadPlugin(char *fn) - Load the specified plug-in.

void GPIO_digitalWrite(uint8_t i, uint8_t val) - Write to a GPIO pin.

void GPIO_digitalWrite(uint8_t i) - Write to all 8 GPIO pins at once.

uint16_t GPIO_digitalRead(void) - Read all 8 GPIO pins at once.

boolean GPIO_digitalRead(uint8_t i) - Read a single GPIO pin.

void GPIO_pinMode(uint8_t i, uint8_t dir) - Set the Pin Mode (INPUT/OUTPUT) for a GPIO pin.

boolean prepareRecordOgg(char *plugin) - Initialize chip for OGG recording.

void startRecordOgg(boolean mic) - Start recording (mic = true for microphone input).

void stopRecordOgg(void) - Stop the recording.

uint16_t recordedWordsWaiting(void) - Returns the number of words recorded.

uint16_t recordedReadWord(void) - Reads the next word from the buffer of recorded words.

uint16_t recordedReadWord(void) - Reads the next word from the buffer of recorded words.

Downloads

Library:

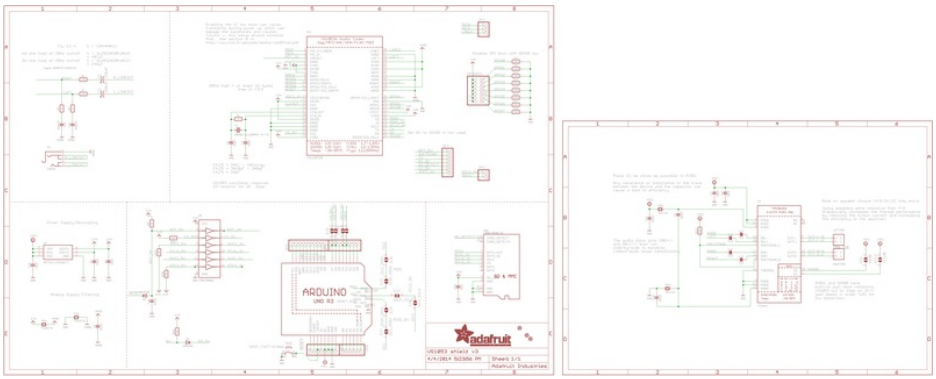
- Adafruit VS1053 Library (https://adafru.it/clE)

Datasheets & Files

- VS1053B (Codec chip) datasheet (https://adafru.it/cll)
- TS2012 3W Class D amplifier datasheet (https://adafru.it/d86)
- Details about the Ogg vorbis encoder/recorder (https://adafru.it/clJ)
- Fritzing objects in Adafruit Fritzing library (https://adafru.it/aP3)
- EagleCAD PCB files in GitHub (https://adafru.it/rgA)

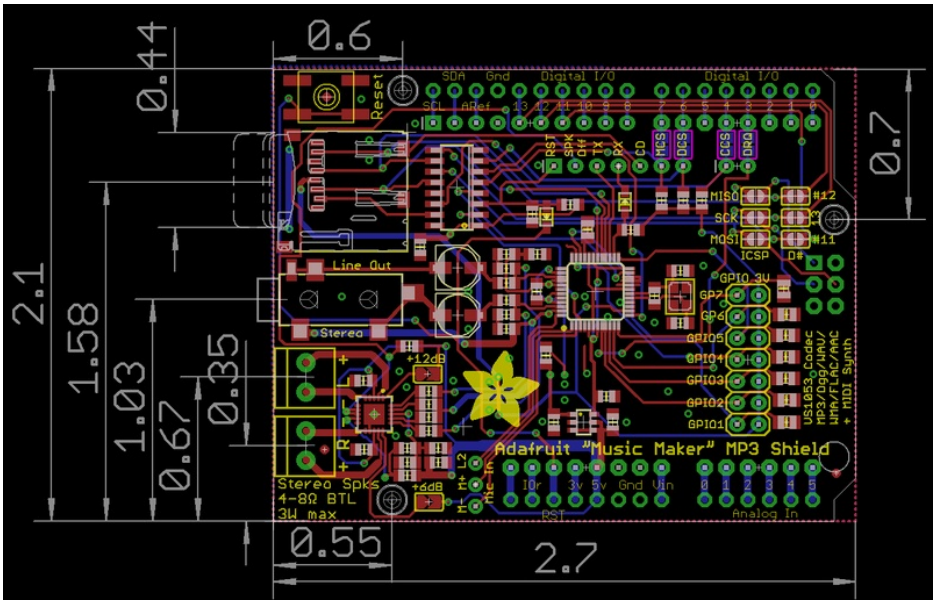
Schematic

optional amplifier is in right hand sheet



Fab print

Dimensions in inches



How can I use the MM shield with an IR receiver or RFID reader?

The MM chip (VS1053) wants to be fed data constantly so that it keeps the music playing. You can take small pauses to do stuff in between playing, and for real-time sensors like IR receivers, you'll have to not-use interrupt based playing and instead pause to read. See this thread for ideas