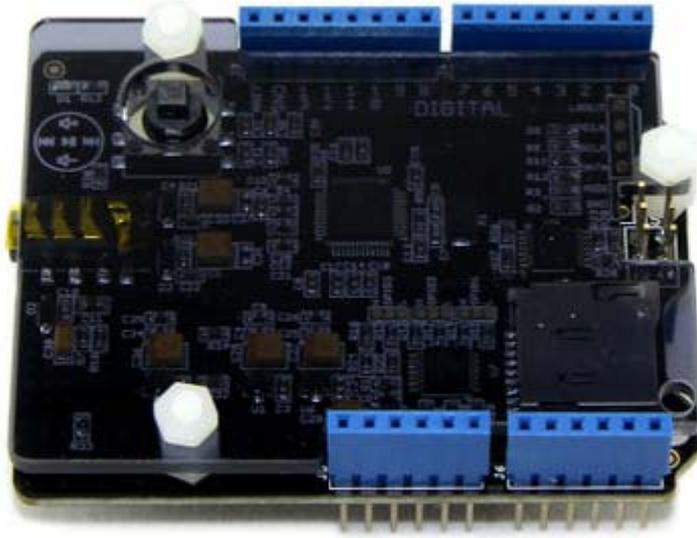


Music Shield V2.2



Time to build your real-time MIDI instrument/music player! It can play many format including MP3,WMA,WAV,AAC,MIDI,Ogg VorbisThe. Music Shield is an audio encoder/decoder compatible with Arduino, Seeeduino, Seeeduino Mega and Arduino Mega. It is based on the VC1053B chip, which enabled it to play sound files from SD card and do short-time recording as well. You can also use it to play MIDI notes by slightly changing its hardware installations. Due to the SPI communication mode, it keeps a minimum number of IO port that facilitates users' own developments of this device. Additionally, the new multifunction button provides greater convenience for users to control.

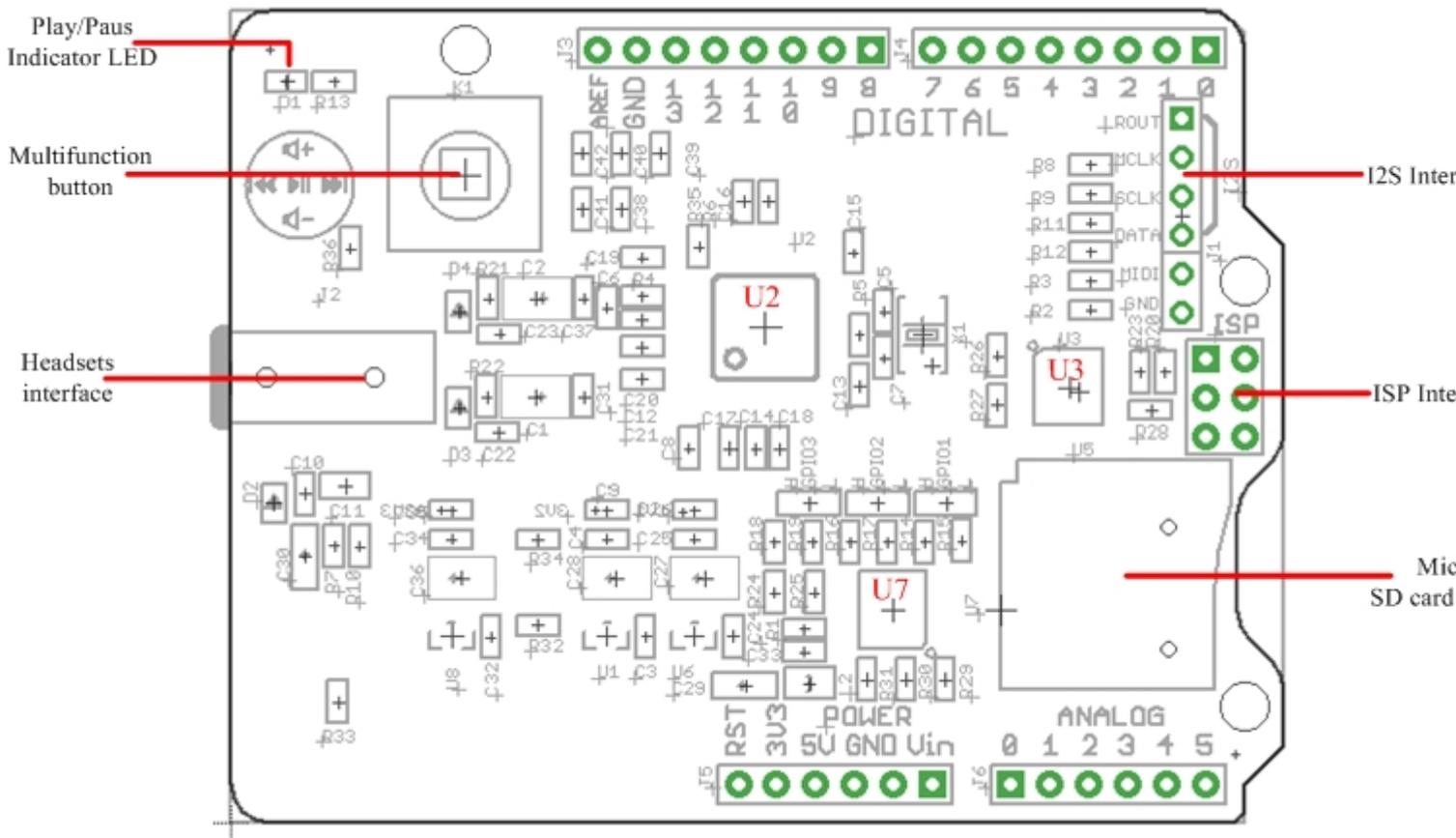
Notice: The recording function works with Seeeduino Mega and Arduino Mega only. And the maximum size SD card you can use is 2GB.

Get One Now 

Contents

- [1 Specification](#)
- [2 Getting Started](#)
 - [2.1 Play music](#)
 - [2.2 Using MIDI,no need to modify the hardware](#)
- [3 Resources](#)
- [4 MIDI number to note reference list](#)

Specification



Multifunction button: Change volume and select songs

Play/Pause indicator LED (GREEN) : Blinks while playing.

Headsets interface: It can drive 16 ohm or 32 ohm earphone and could serve as a external audio input port.

Micro SD card: can be FAT16 or FAT32, The maximum size SD card you can use is 2GB.

U2: VS1053B IC,Ogg Vorbis/MP3/AAC/WMA/FLAC/MIDI audio codec.

U3,U7: 74VHC125 IC, Quad Buffer.

I2S: for digital audio input/output.

ISP interface: for bringing SPI port when using with Mega series products.

Pins usage on Arduino

Pins Used for Play Control:

D3 - receiving signal from button for Volume Up;

D4 - receiving signal from switch for Next Song function;

D5 - receiving signal from switch for Play&Stop and Record function;

D6 - receiving signal from switch for Previous Song function;

D7 - receiving signal from button for Volume Down.

D8 - Green Led instructions;

Pins Used for SPI Interface:

D10 - SPI Chip Select;

D11 - SPI MOSI;

D12 - SPI MISO;

D13 - SPI SCK;

Pins Used for VS1053 Interface:

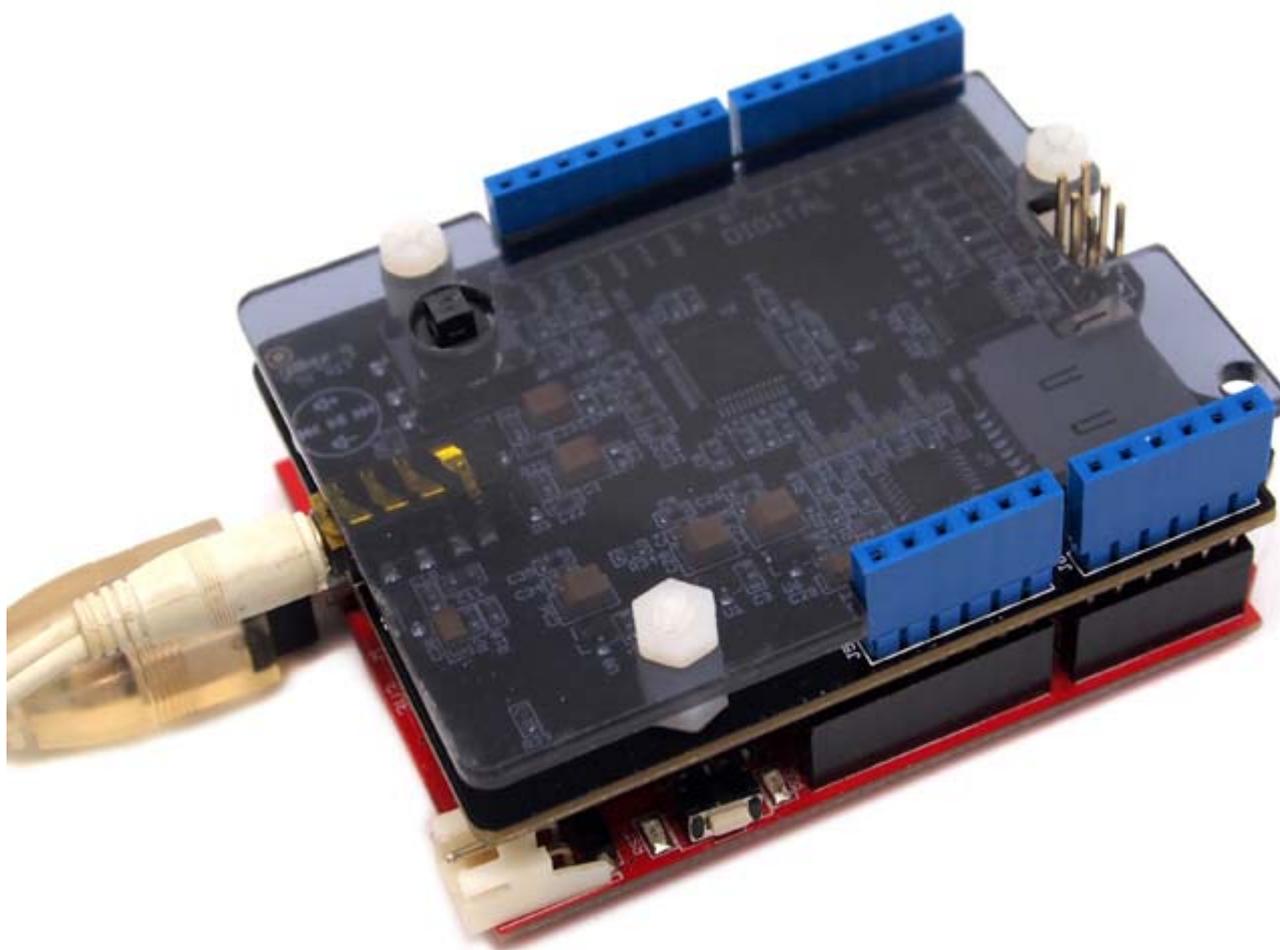
A0 - Reset of VS1053;

A1- Data Require of VS1053;

A2 - Data Select of VS1053;

A3 - Chip Select of VS1053;

Getting Started



Note: 1. If you want to use MIDI function, you need to change the hardware installation.

2. If you changed the hardware installation in order to use MIDI function, you are not able to use playback & recording functions until you restore it to the original condition.

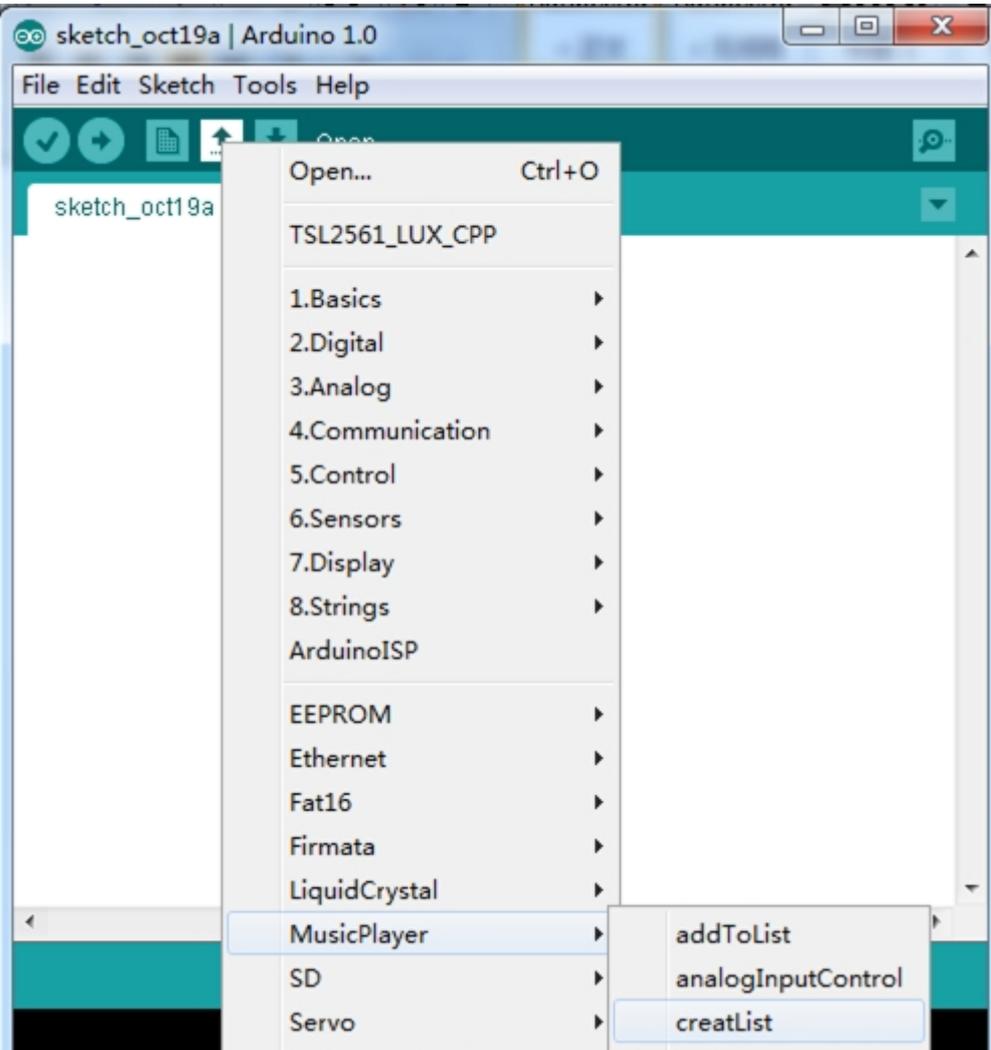
Play music

1. Make sure there are songs in the micro SD card;
 2. Download [Music shield V2.0 library](#)
 3. Unzip and copy the folder to Arduino's library path: ..\arduino-1.0\libraries;
- Note 1:** Change the folder name of the extracted library if Arduino throws an error while loading.
- Note 2:** Change the included header (to Arduino.h) in the example file if there is a '*arduino.h: No such file or directory*' error while compiling.

Demo 1: Play songs (e.g. in shuffle mode)

In order to use the playback function, you need to create a playlist first.

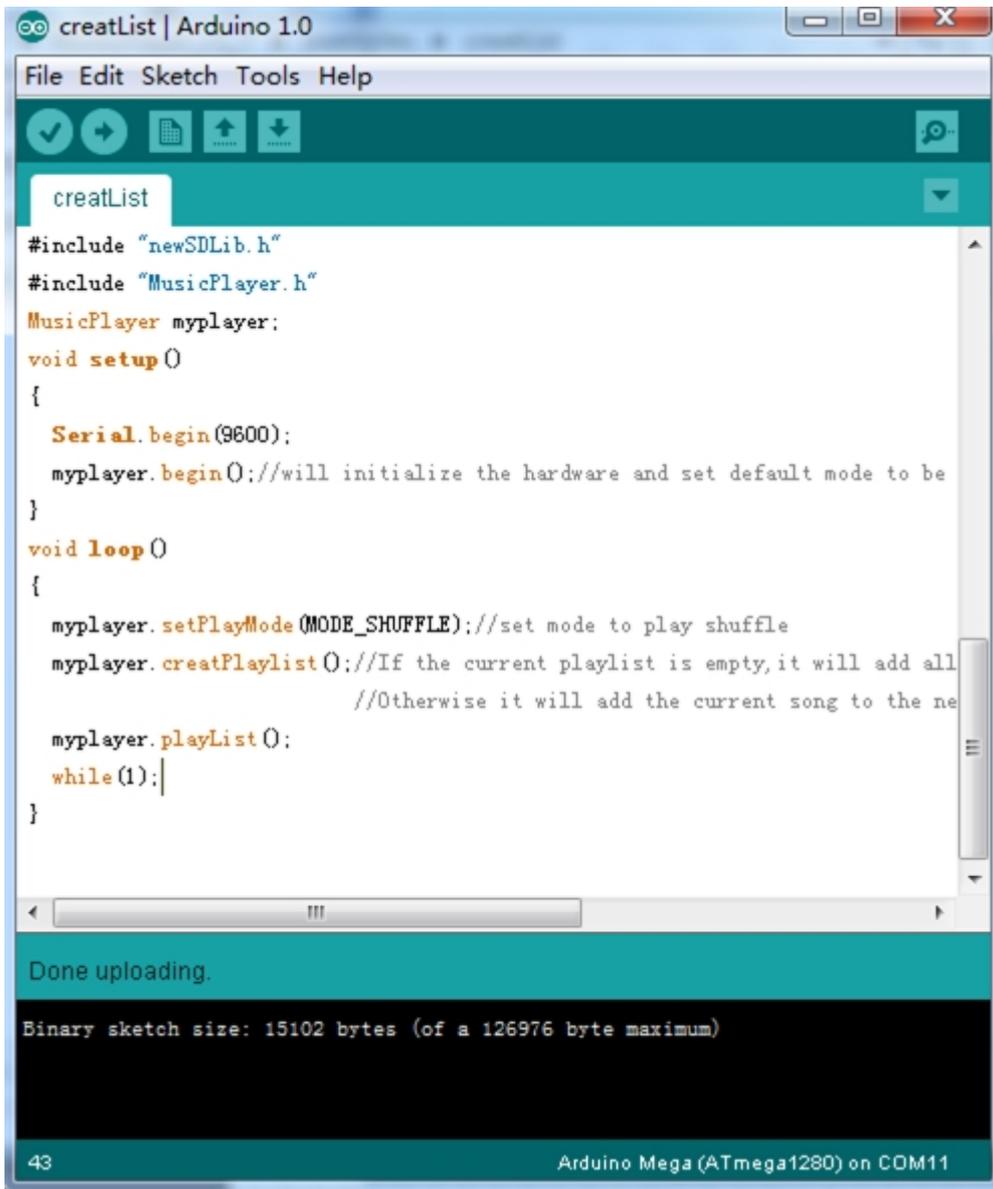
1. Restart the Arduino IDE. Open "creatList" example via the path: File --> Examples --> MusicPlayer --> creatList as below.



2. Set the play mode. In "creatList", the function we use is described as follow.

Name: setPlayMode(unsigned char playmode);

- Function:** Set the play mode. There are four modes you can set: MODE_NORMAL、MODE_SHUFFLE、MODE_REPEAT_LIST、MODE_REPEAT_ONE. Each mode stands for different playing orders.



3. Select the type of Arduino board that you are using by the path: Tools --> Board --> for example Arduino UNO.
4. Select the correct serial port you are using by the path: Tools --> Serial Port --> for example COM3.
5. Upload the code. Click to Serial Monitor when "Done uploading" appears, you will find the order of songs is randomized on the list.

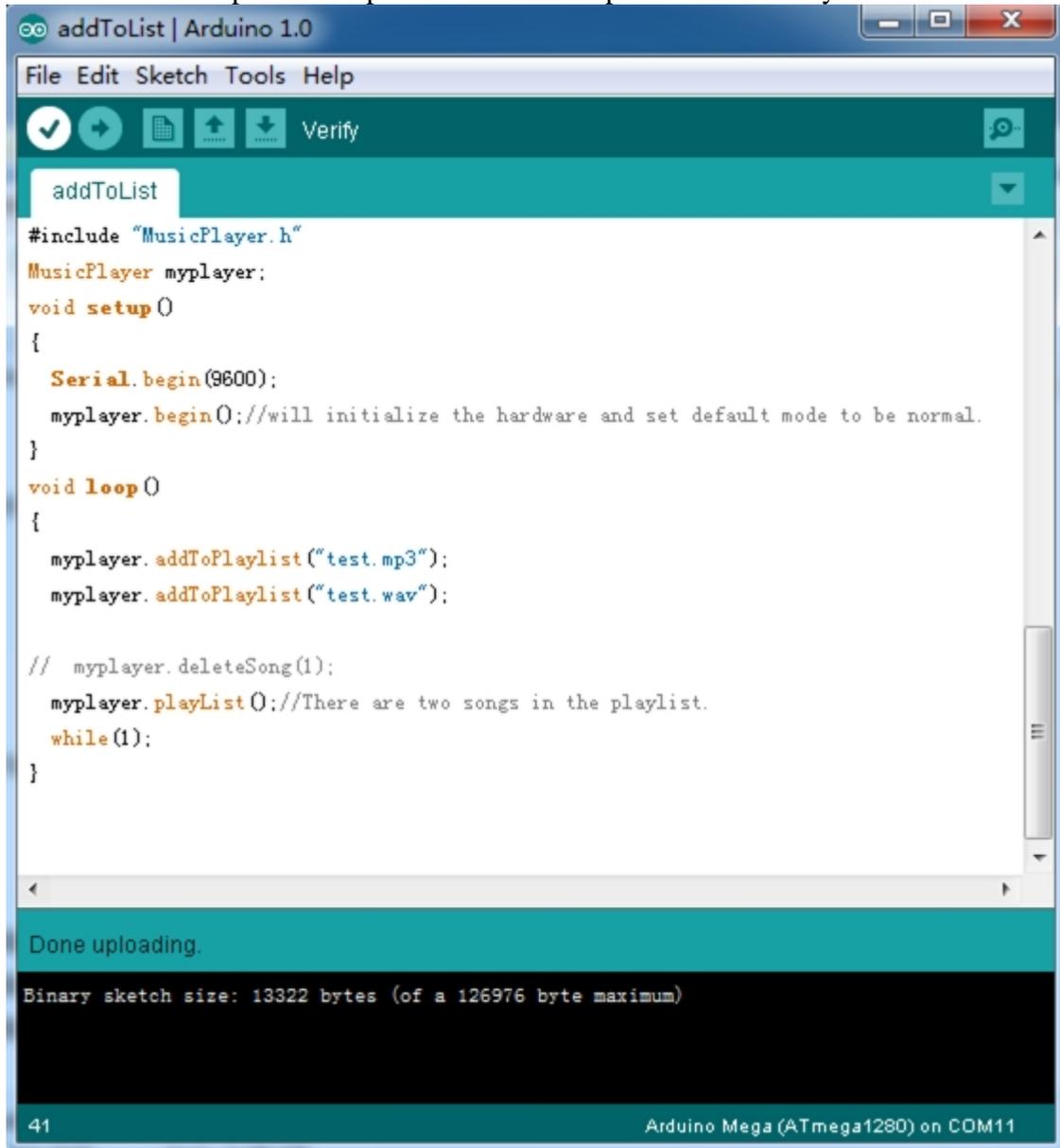


When press multifunction button to up or down, the volume will change. Of course, you can try others play modes.

Demo 2: Play selected songs

1. This demo will show you how to play part of the songs from all songs in the SD card. Open the

“addToList” example via the path: File --> Examples --> MusicPlayer --> addToList.



```
addToList | Arduino 1.0
File Edit Sketch Tools Help
Verify
addToList
#include "MusicPlayer.h"
MusicPlayer myplayer;
void setup()
{
  Serial.begin(9600);
  myplayer.begin(); //will initialize the hardware and set default mode to be normal.
}
void loop()
{
  myplayer.addToPlaylist("test.mp3");
  myplayer.addToPlaylist("test.wav");

  // myplayer.deleteSong(1);
  myplayer.playlist(); //There are two songs in the playlist.
  while(1);
}
Done uploading.
Binary sketch size: 13322 bytes (of a 126976 byte maximum)
41 Arduino Mega (ATmega1280) on COM11
```

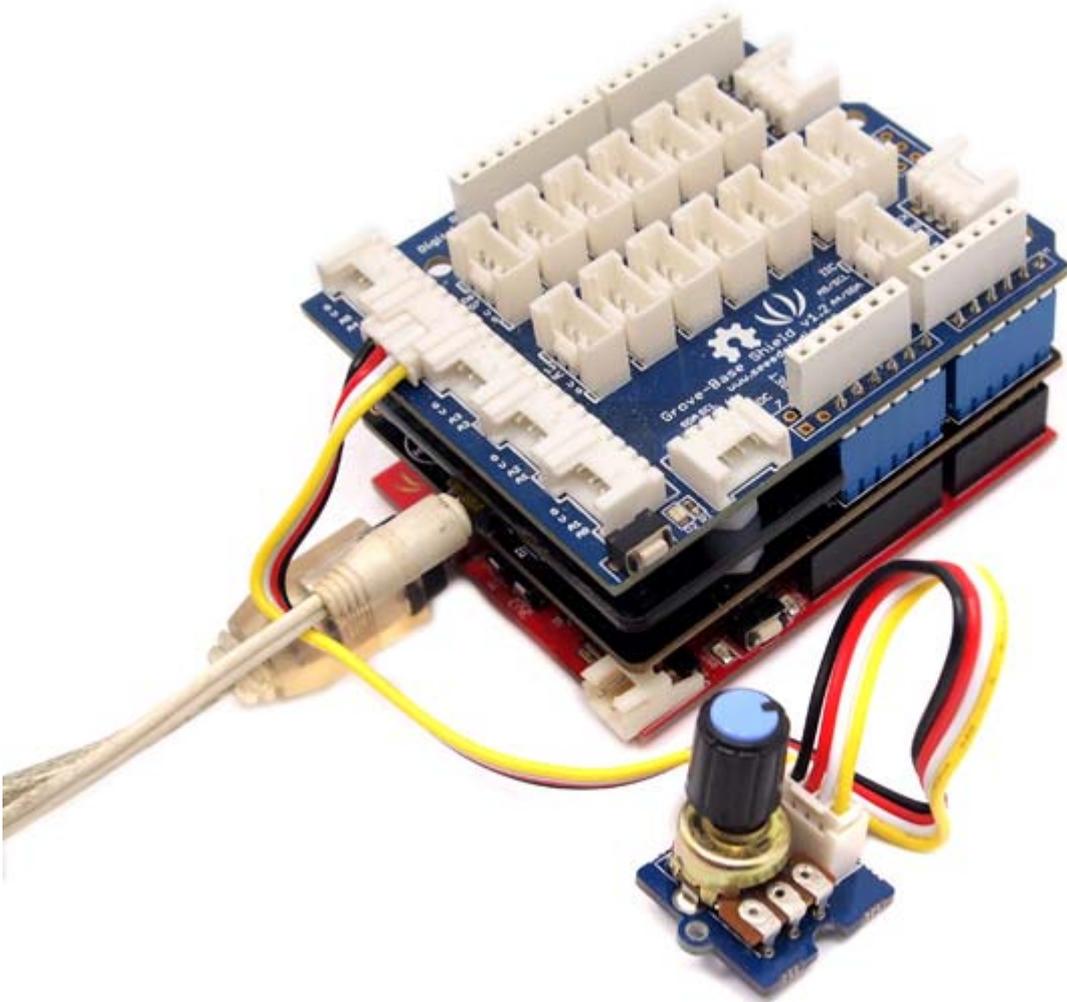
2. Select songs from the playlist. You just need to list songs you want to play by name correctly in the function `addToPlaylist(char *songName)`.

But you must ensure that the song has been stored in the SD card and the format of those songs must be one of MP3, WMA, WAV, AAC, MIDI, Ogg Vorbis.

3. Upload code. When you complete the upload, new add songs will be played.

Demo 3: Control Volume by analog port

1. Plug the Grove-Base Shield onto the Music shield, Connect the Grove socket of the Rotary and analog port 4 of the Base Shield with a Grove cable. You can change to the digital port as well. But don't forget to change the port number in the definition of the demo code at the same time.



2. Open the “analogInputControl” example and upload it onto your Arduino Board.
3. Rotate the knob to change music volume.

Demo 4: Record music:(Only support ATmega1280 and ATmega2560 based board)

1. Upload any sketch in Music Shield library, for example the sketch "creatList". Open the Serial Monitor and it will play audio files on SD card.
2. Press down the multifunction button for 5 seconds, then the indicator LED will light off.
3. Press down the multifunction button for 5 seconds again, then the music shield will begin to record, the green indicator LED will blink.
4. Quickly press down the multifunction button again, it will stop recording.
5. Record will be played in the last place.

Using MIDI,no need to modify the hardware

The VS1058X's real time MIDI mode:

The "real time MIDI mode",in which it will instantly execute MIDI commands send to it through either SPI or UART,can be enabled with the method below:

Method: At the beginning,send a small software patch through SPI port.

```

    /*software patch for MIDI Play*/
const unsigned short gVS1053_MIDI_Patch[28]={
    /*if you don't let GPIO1 = H,please send this patch by spi*/
    0x0007, 0x0001, 0x8050, 0x0006, 0x0014, 0x0030, 0x0715, 0xb080, /* 0 */
    0x3400, 0x0007, 0x9255, 0x3d00, 0x0024, 0x0030, 0x0295, 0x6890, /* 8 */
    0x3400, 0x0030, 0x0495, 0x3d00, 0x0024, 0x2908, 0x4d40, 0x0030, /* 10 */
    0x0200, 0x000a, 0x0001, 0x0050,
};
using that function to load:
/*
**@ function name: loadMidiPatch
**@ usage:load a software patch for vs10xx
**@ input:none
**@ retval:none
*/
void VS10XX::loadMidiPlugin(void)
{
    int i=0;
    Serial.print("load MIDI Plugin...\r\n");
    while(i < sizeof(gVS1053_MIDI_Patch)/sizeof(gVS1053_MIDI_Patch[0]))
    {
        unsigned short addr, n, val;
        addr = gVS1053_MIDI_Patch[i++];
        n = gVS1053_MIDI_Patch[i++];
        while(n--)
        {
            val = gVS1053_MIDI_Patch[i++];
            writeRegister(addr, val >> 8, val&0xff);
        }
    }
    Serial.print("done\r\n");
}

```

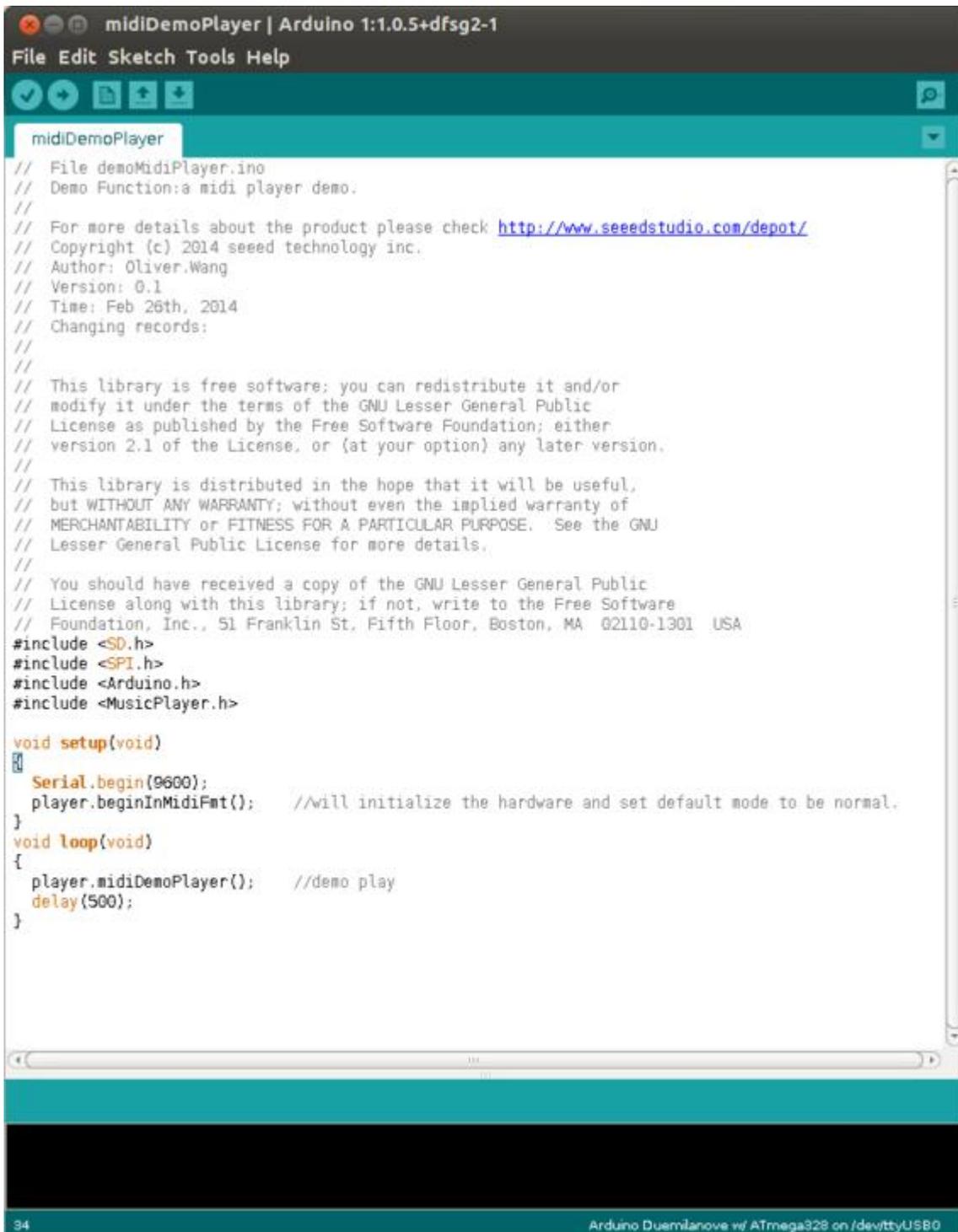
I would like to tell you that there is an open source library called `jdksmidi`,by which you can make your own MIDI decoder through some small changes. `jdksmidi` git-hub page:<https://github.com/jdkoftinoff/jdksmidi> we offer you some real time mode MIDI APIs(`MusicPlayer.cpp`):

```

midiNoteOn()
midiNoteOff()
midiWriteData()

```

Now,it's time to build your real-time MIDI instrument/music player in any format(single-channel or multi-channel). Your contribution is appreciated. A demo MIDI player was add to the latest library. MIDI Demo(upload the code. When completed, you will hear Fancy MIDI music):

The image shows a screenshot of the Arduino IDE interface. The title bar reads "midiDemoPlayer | Arduino 1:1.0.5+dfsg2-1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for saving, undo, redo, and search. The main editor window displays the code for "midiDemoPlayer". The code includes comments about the product, copyright (c) 2014 seeed technology inc., author (Oliver.Wang), version (0.1), and time (Feb 26th, 2014). It also includes a license notice for the GNU Lesser General Public License. The code includes headers for <SD.h>, <SPI.h>, <Arduino.h>, and <MusicPlayer.h>. The setup function initializes the serial port at 9600 baud and sets the player to MIDI format. The loop function calls player.midiDemoPlayer() and delays for 500ms.

```
// File demoMidiPlayer.ino
// Demo Function:a midi player demo.
//
// For more details about the product please check http://www.seeedstudio.com/depot/
// Copyright (c) 2014 seeed technology inc.
// Author: Oliver.Wang
// Version: 0.1
// Time: Feb 26th, 2014
// Changing records:
//
//
// This library is free software; you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public
// License as published by the Free Software Foundation; either
// version 2.1 of the License, or (at your option) any later version.
//
// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
// Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this library; if not, write to the Free Software
// Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
#include <SD.h>
#include <SPI.h>
#include <Arduino.h>
#include <MusicPlayer.h>

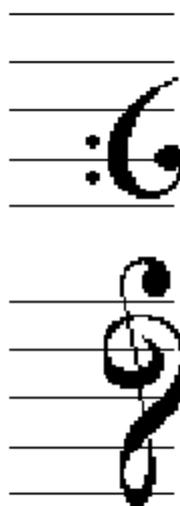
void setup(void)
{
  Serial.begin(9600);
  player.beginInMidiFmt(); //will initialize the hardware and set default mode to be normal.
}
void loop(void)
{
  player.midiDemoPlayer(); //demo play
  delay(500);
}
```

Resources

- [Music Shield V2.2 Eagle Files](#)
- [Music Shield V2.2 Schematic.pdf](#)
- [VS1053 IC.pdf](#)
- [Music Shield libraries](#)

MIDI number to note reference list

MIDI number	Note name	Keyboard	Frequency Hz	Period ms
21	22	A0	27.500	36.36
23	B0		30.868	29.135
24	25	C1	32.703	30.58
26	27	D1	36.708	34.648
28	E1		41.203	38.891
29	30	F1	43.654	22.91
31	32	G1	48.999	46.249
33	34	A1	55.000	51.913
35	B1		61.735	58.270
36	37	C2	65.406	15.29
38	39	D2	73.416	69.296
40	E2		82.407	77.782
41	42	F2	87.307	11.45
43	44	G2	97.999	92.499
45	46	A2	110.00	103.83
47	B2		123.47	116.54
48	49	C3	130.81	7.645
50	51	D3	146.83	138.59
52	E3		164.81	155.56
53	54	F3	174.61	6.068
55	56	G3	196.00	5.727
57	58	A3	220.00	5.102
59	59	B3	246.94	207.65
60	61	C4	261.63	3.822
62	63	D4	293.67	233.08
64	E4		329.63	4.050
65	66	F4	349.23	2.863
67	68	G4	392.00	3.405
69	70	A4	440.00	2.273
71	B4		493.88	3.034
72	73	C5	523.25	2.863
74	75	D5	587.33	2.551
76	E5		659.26	2.703
77	78	F5	698.46	4.153
79	80	G5	783.99	4.661
81	82	A5	880.00	2.025
83	B5		987.77	2.145
84	85	C6	1046.5	1.910
86	87	D6	1174.7	1.703
88	E6		1318.5	1.804
89	90	F6	1396.9	1.607
91	92	G6	1568.0	1.432
93	94	A6	1760.0	1.276
95	B6		1975.5	1.204
96	97	C7	2093.0	1.136
98	99	D7	2349.3	1.012
100	E7		2637.0	0.9556
101	F7		2793.0	0.8513
101	102	G7	3136.0	0.7584
103	104	A7	3520.0	0.7159
105	106	B7	3951.1	0.6378
107	B7		3951.1	0.5682
108	C8	J. Wolfe, UNSW	4186.0	0.5062



- 89.175.72.2
- [Talk for this IP address](#)
- [Log in](#)
- [Page](#)
- [FAQ](#)
- [Read](#)
- [View source](#)
- [View history](#)

[Bazaar](#)

Navigation

- [Main page](#)
- [Random page](#)

- [Recent changes](#)

Collections

- [Motor](#)
- [Arduino](#)
- [Grove](#)
- [Shield](#)
- [Kit](#)
- [Xadow](#)
- [Rephone](#)

Toolbox

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Printable version](#)
- [Permanent link](#)



-