TF Card U Disk Mini MP3 Player Decoder Board Audio Voice Module FM For Arduino DF Play Min Board

Rated 4.9/5 based on 38 customer reviews 4.9 (38 votes)
65 orders
Price:
US $7.88 for 5 pieces, US $1.58 / piece
Free Shipping to Australia via AliExpress Standard Shipping
Estimated Delivery Time:20-41days

- Brand Name: Agoal
- Battery Specification: External Power
- Balanced Out: No
- Screen: No
- Supports Recording Function: No
- Supports FM: No
- Supports EBook Reading: No
- Body Material: Metal
- Package: No
- Audio Format Support: WMA,WAV,MP3, WAV,MP3
- Bluetooth: No
- Style: Pure Audio MP3
- WIFI: No
- Storage Type: Flash Memory
- Battery Life: > 20 hours
- Operation Mode: Touch Tone
- Model Number: DFplayer mini
- External Memory: Yes
- Have Speakers or not: No
- Balanced Out: YSE

Description:

DFPlayer Mini is a compact and inexpensive MP3 module can be directly connected to the speaker. Module with battery power, speaker, and keypad can be used alone, or through the serial port control, as the Arduino UNO or any microcontroller with a serial port module. The module itself perfectly integrated hardware decode MP3, WAV, WMA's. While the software supports TF card driver to support FAT16, FAT32 file system. Can be done through simple serial commands specified music player, as well as how to play music and other functions, without the cumbersome underlying operating, easy to use, stable and reliable.

Technical Specifications:

Supports sampling rates (KHz): 8 / 11.025 / 12/16 / 22.05 / 24/32 / 44.1 / 48
24-bit DAC output, support dynamic range: 90dB, SNR support: 85dB
Fully supports FAT16, FAT32 file system, maximum support 32G TF card, support U disk to 32G, 64M bytes NORFLASH

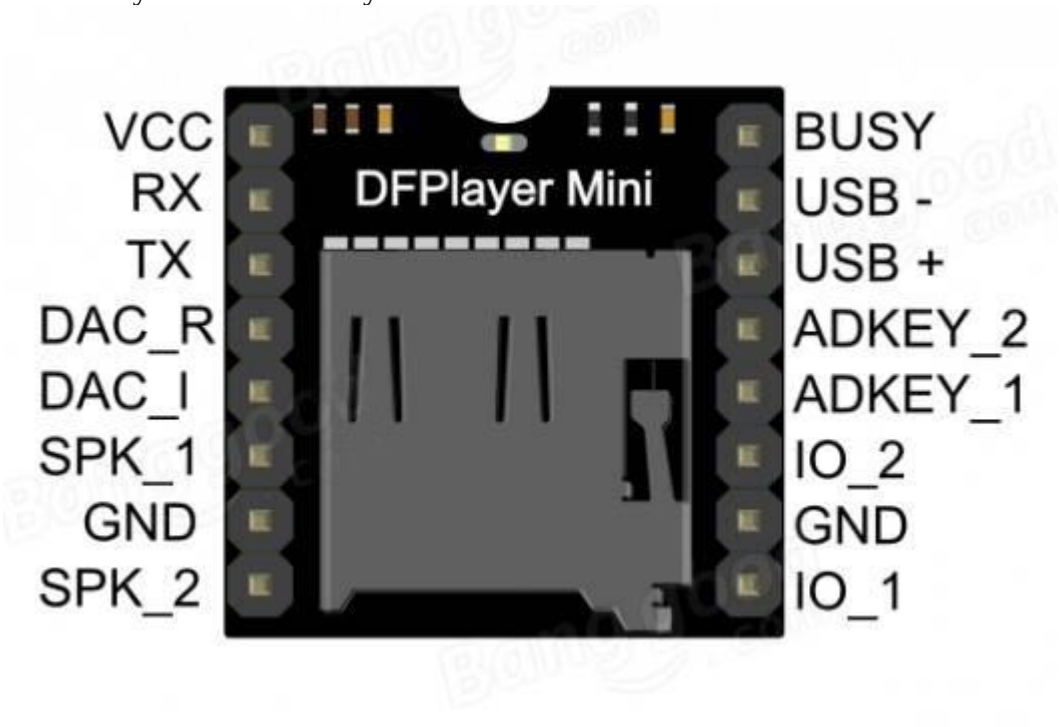
A variety of control modes are available. IO control, serial port, AD button control mode
Radio spots language function, you can pause the background music being played. Advertising finished playing background sound continues to play back
Audio data is sorted by folder, supports up to 100 folders, folders can be assigned to every 255 Tracks
30 level adjustable volume, six adjustable EQ

Application:

Car navigation voice broadcast
Road transport inspectors, toll stations voice prompts
Train, bus safety inspection voice prompts
Electricity, communications, financial operating room voice prompts
Vehicles into and out of the channel to verify the voice prompts
** frontier channel voice prompts
Multi-channel voice alarm or voice guidance equipment operation
Electric sightseeing bus safety with voice announcement
Electrical and mechanical equipment failure alarm
Fire alarm voice prompts
Automatic broadcast equipment, regular broadcast

Package included:

5 x DFPlayer Mini MP3 Player Module For Arduino



Packaging Details

- Unit Type: lot (5 pieces/lot)
- Package Weight: 0.1kg (0.22lb.)
- Package Size: 5cm x 5cm x 10cm (1.97in x 1.97in x 3.94in)

DFPlayer Mini SKU:DFR0299

DFPlayer - A Mini MP3 Player For Arduino

Contents

Introduction

The **DFPlayer Mini MP3 Player For Arduino** is a small and low price MP3 module with an simplified output directly to the speaker. The module can be used as a standalone module with attached battery, speaker and push buttons or used in combination with an **Arduino UNO** or any other with RX/TX capabilities.

Specification

- supported sampling rates (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- 24 -bit DAC output, support for dynamic range 90dB , SNR support 85dB
- fully supports FAT16 , FAT32 file system, maximum support 32G of the TF card, support 32G of U disk, 64M bytes NORFLASH
- a variety of control modes, I/O control mode, serial mode, AD button control mode
- Advertising sound waiting function, the music can be suspended. when advertising is over in the music continue to play
- audio data sorted by folder, supports up to 100 folders, every folder can hold up to 255 songs
- 30 level adjustable volume, 6 -level EQ adjustable

Application

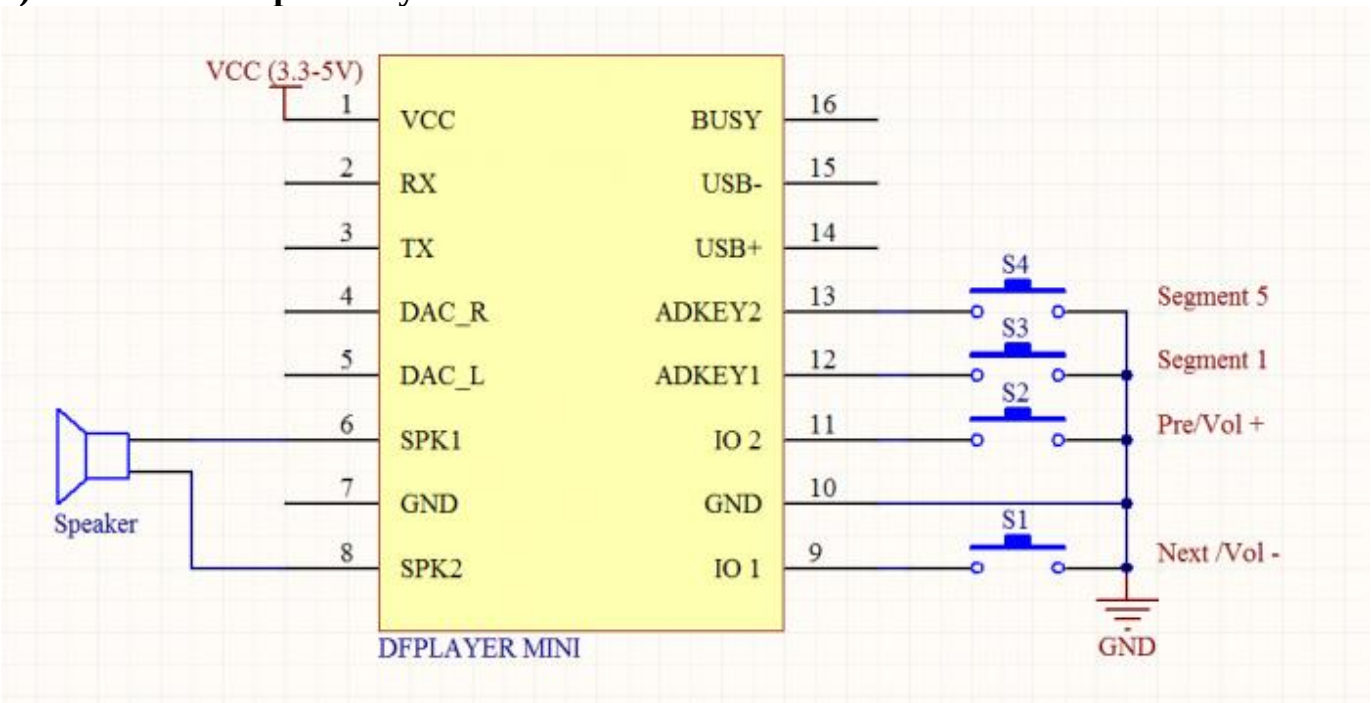
- Car navigation voice broadcast;
- Road transport inspectors, toll stations voice prompts;
- Railway station, bus safety inspection voice prompts;
- Electricity, communications, financial business hall voice prompts;
- Vehicle into and out of the channel verify that the voice prompts;
- The public security border control channel voice prompts;
- Multi-channel voice alarm or equipment operating guide voice;
- The electric tourist car safe driving voice notices;
- Electromechanical equipment failure alarm;
- Fire alarm voice prompts;
- The automatic broadcast equipment, regular broadcast.

Pin Map

Pin	Description	Note
VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
RX	UART serial input	
TX	UART serial output	
DAC_R	Audio output right channel	Drive earphone and amplifier
DAC_L	Audio output left channel	Drive earphone and amplifier
SPK2	Speaker-	Drive speaker less than 3W
GND	Ground	Power GND
SPK1	Speaker+	Drive speaker less than 3W
IO1	Trigger port 1	Short press to play previous（long press to decrease volume）
GND	Ground	Power GND
IO2	Trigger port 2	Short press to play next （long press to increase volume）
ADKEY1	AD Port 1	Trigger play first segment
ADKEY2	AD Port 2	Trigger play fifth segment
USB+	USB+ DP	USB Port
USB-	USB- DM	USB Port
BUSY	Playing Status	Low means playing \High means no

Work Mode

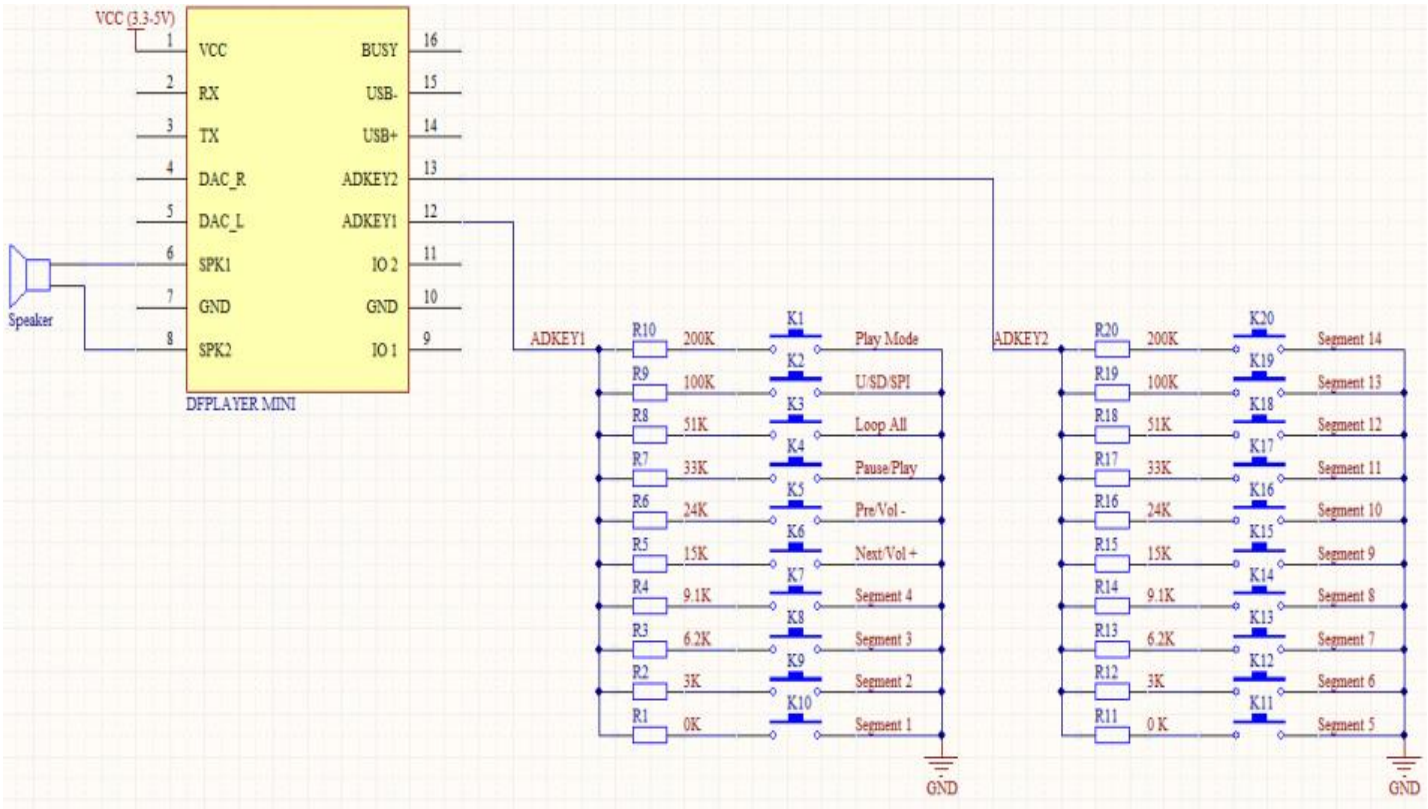
1) I/O Mode - simplest way to use this module.



Note: For pins 9 & 11 a short press is previous/next, long press means volume- ,volume +

2) AD KEY Mode

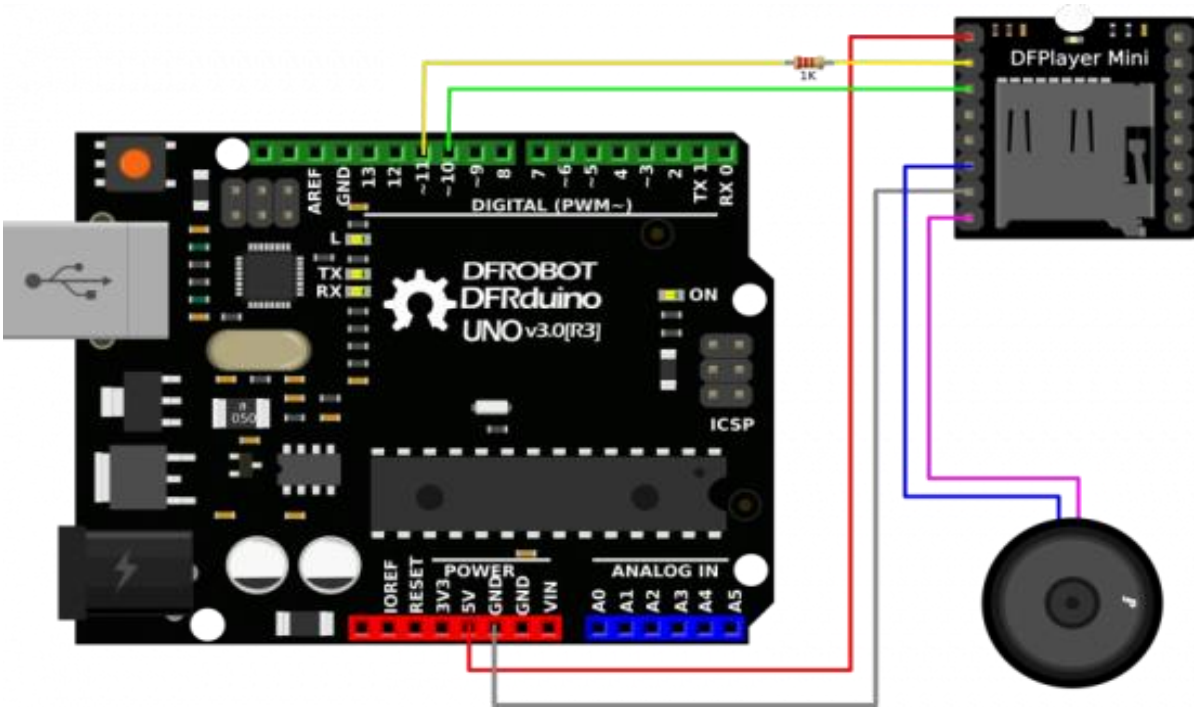
AD module keys are available instead of the traditional method of matrix keyboard connection. Taking advantage of increasingly powerful MCU AD functionality. The default configuration is 2 AD ports providing a 20 key resistance division.



3) Serial Mode

Support for asynchronous serial communication mode via PC serial sending commands
Communication Standard:9600 bps
Data bits :1 Checkbit :none Flow Control :none

Connection Diagram - Note: If sound level is too high attach a 1K resistor to the TX pin.



- Instruction Description

Format: $S VER Len CMD Feedback para1 para2 checksum $O		
$S	Start bit 0x7E	Each command feedback begin with $, that is 0x7E
VER	Version	Version Information
Len	the number of bytes after "Len"	Checksums are not counted
CMD	Commands	Indicate the specific operations, such as play / pause, etc.
Feedback	Command feedback	If need for feedback, 1: feedback, 0: no feedback
para1	Parameter 1	Query high data byte
para2	Parameter 2	Query low data byte
checksum	Checksum	Accumulation and verification [not include start bit $]
$O	End bit	End bit 0xEF

For example, if we specify play NORFLASH, you need to send: 7E FF 06 09 00 00 04 FF DD EF
Data length is 6, which are 6 bytes [FF 06 09 00 00 04]. Not counting the start, end, and verification.

- Serial Control Cmd

CMD	Function Description	Parameters(16 bit)
0x01	Next	
0x02	Previous	
0x03	Specify tracking(NUM)	0-2999
0x04	Increase volume	
0x05	Decrease volume	
0x06	Specify volume	0-30
0x07	Specify EQ(0/1/2/3/4/5)	Normal/Pop/Rock/Jazz/Classic/Base
0x08	Specify playback mode (0/1/2/3)	Repeat/folder repeat/single repeat/ random
0x09	Specify playback source(0/1/2/3/4)	U/TF/AUX/SLEEP/FLASH
0x0A	Enter into standby – low power loss	
0x0B	Normal working	
0x0C	Reset module	
0x0D	Playback	
0x0E	Pause	
0x0F	Specify folder to playback	1~10(need to set by user)
0x10	Volume adjust set	{DH=1:Open volume adjust }{DL: set volume gain 0~31}
0x11	Repeat play	{1:start repeat play}{0:stop play}

- Serial Query Cmd

Commands	Function Description	Parameters(16 bit)
0x3C	STAY	
0x3D	STAY	
0x3E	STAY	
0x3F	Send initialization parameters	0 - 0x0F(each bit represent one device of the low-four bits)
0x40	Returns an error, request retransmission	
0x41	Reply	
0x42	Query the current status	
0x43	Query the current volume	
0x44	Query the current EQ	
0x45	Query the current playback mode	
0x46	Query the current software version	
0x47	Query the total number of TF card files	
0x48	Query the total number of U-disk files	
0x49	Query the total number of flash files	
0x4A	Keep on	
0x4B	Queries the current track of TF card	
0x4C	Queries the current track of U-Disk	
0x4D	Queries the current track of Flash	

Copy your mp3 into you micro SD card

NOTE: The order you copy the mp3 into micro SD card will affect the order mp3 played, which means play(1) function will play the first mp3 copied into micro SD card.

For Mac User

NOTE: If you are using Mac OS X to copy the mp3, the file system will automatically add hidden files like: "._0001.mp3" for index, which this module will handle as valid mp3 files. It is really annoying. So you can run following command in terminal to eliminate those files.

```
dot_clean /Volumes/<SDVolumeName>
```

Please replace the <SDVolumeName> to the volume name of your SD card.

Sample Code

Arduino library for DFPlayer Mini DFRobotDFPlayerMini
Connect the hardware as the picture above shown and play with the sample code..

Sample code "GetStarted", switching to next song every 3 seconds

```
/***************************************************
DFPlayer - A Mini MP3 Player For Arduino
 <https://www.dfrobot.com/index.php?route=product/product&product_id=1121>

 ***************************************************
 This example shows the basic function of library for DFPlayer.

 Created 2016-12-07
 By [Angelo qiao](Angelo.qiao@dfrobot.com)

 GNU Lesser General Public License.
 See <http://www.gnu.org/licenses/> for details.
 All above must be included in any redistribution
 ****************************************************/

/***********Notice and Trouble shooting***************
 1.Connection and Diagram can be found here
 <https://www.dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299#Connection_Diagram>
 2.This code is tested on Arduino Uno, Leonardo, Mega boards.
 ****************************************************/

#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"

SoftwareSerial mySoftwareSerial(10, 11); // RX, TX
DFRobotDFPlayerMini myDFPlayer;
void printDetail(uint8_t type, int value);

void setup()
{
  mySoftwareSerial.begin(9600);
  Serial.begin(115200);

  Serial.println();
  Serial.println(F("DFRobot DFPlayer Mini Demo"));
  Serial.println(F("Initializing DFPlayer ... (May take 3~5 seconds)"));

  if (!myDFPlayer.begin(mySoftwareSerial)) {  //Use softwareSerial to communicate with
mp3.
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1.Please recheck the connection!"));
    Serial.println(F("2.Please insert the SD card!"));
    while(true);
  }
  Serial.println(F("DFPlayer Mini online."));

  myDFPlayer.volume(10);  //Set volume value. From 0 to 30
  myDFPlayer.play(1);  //Play the first mp3
}

void loop()
{
  static unsigned long timer = millis();

  if (millis() - timer > 3000) {
    timer = millis();
    myDFPlayer.next();  //Play next mp3 every 3 second.
  }
```

```
  if (myDFPlayer.available()) {
    printDetail(myDFPlayer.readType(), myDFPlayer.read()); //Print the detail message
from DFPlayer to handle different errors and states.
  }
}

void printDetail(uint8_t type, int value){
  switch (type) {
    case TimeOut:
      Serial.println(F("Time Out!"));
      break;
    case WrongStack:
      Serial.println(F("Stack Wrong!"));
      break;
    case DFPlayerCardInserted:
      Serial.println(F("Card Inserted!"));
      break;
    case DFPlayerCardRemoved:
      Serial.println(F("Card Removed!"));
      break;
    case DFPlayerCardOnline:
      Serial.println(F("Card Online!"));
      break;
    case DFPlayerPlayFinished:
      Serial.print(F("Number:"));
      Serial.print(value);
      Serial.println(F(" Play Finished!"));
      break;
    case DFPlayerError:
      Serial.print(F("DFPlayerError:"));
      switch (value) {
        case Busy:
          Serial.println(F("Card not found"));
          break;
        case Sleeping:
          Serial.println(F("Sleeping"));
          break;
        case SerialWrongStack:
          Serial.println(F("Get Wrong Stack"));
          break;
        case CheckSumNotMatch:
          Serial.println(F("Check Sum Not Match"));
          break;
        case FileIndexOut:
          Serial.println(F("File Index Out of Bound"));
          break;
        case FileMismatch:
          Serial.println(F("Cannot Find File"));
          break;
        case Advertise:
          Serial.println(F("In Advertise"));
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }
}
```

Sample code "FullFunction", including all the functions. Please read the comments and documents in detail

```
/***************************************************
 DFPlayer - A Mini MP3 Player For Arduino
 <https://www.dfrobot.com/index.php?route=product/product&product_id=1121>
```

```
 ***************************************************
 This example shows the all the function of library for DFPlayer.

 Created 2016-12-07
 By [Angelo qiao](Angelo.qiao@dfrobot.com)

 GNU Lesser General Public License.
 See <http://www.gnu.org/licenses/> for details.
 All above must be included in any redistribution
 ****************************************************/

/***********Notice and Trouble shooting***************
 1.Connection and Diagram can be found here
<https://www.dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299#Connection_Diagram>
 2.This code is tested on Arduino Uno, Leonardo, Mega boards.
 ****************************************************/

#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"

SoftwareSerial mySoftwareSerial(10, 11); // RX, TX
DFRobotDFPlayerMini myDFPlayer;
void printDetail(uint8_t type, int value);

void setup()
{
  mySoftwareSerial.begin(9600);
  Serial.begin(115200);

  Serial.println();
  Serial.println(F("DFRobot DFPlayer Mini Demo"));
  Serial.println(F("Initializing DFPlayer ... (May take 3~5 seconds)"));

  if (!myDFPlayer.begin(mySoftwareSerial)) {  //Use softwareSerial to communicate with
mp3.
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1.Please recheck the connection!"));
    Serial.println(F("2.Please insert the SD card!"));
    while(true);
  }
  Serial.println(F("DFPlayer Mini online."));

  myDFPlayer.setTimeOut(500); //Set serial communictaion time out 500ms

  //----Set volume----
  myDFPlayer.volume(10);  //Set volume value (0~30).
  myDFPlayer.volumeUp(); //Volume Up
  myDFPlayer.volumeDown(); //Volume Down

  //----Set different EQ----
  myDFPlayer.EQ(DFPLAYER_EQ_NORMAL);
//  myDFPlayer.EQ(DFPLAYER_EQ_POP);
//  myDFPlayer.EQ(DFPLAYER_EQ_ROCK);
//  myDFPlayer.EQ(DFPLAYER_EQ_JAZZ);
//  myDFPlayer.EQ(DFPLAYER_EQ_CLASSIC);
//  myDFPlayer.EQ(DFPLAYER_EQ_BASS);

  //----Set device we use SD as default----
//  myDFPlayer.outputDevice(DFPLAYER_DEVICE_U_DISK);
  myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);
//  myDFPlayer.outputDevice(DFPLAYER_DEVICE_AUX);
//  myDFPlayer.outputDevice(DFPLAYER_DEVICE_SLEEP);
//  myDFPlayer.outputDevice(DFPLAYER_DEVICE_FLASH);

  //----Mp3 control----
```

```
//  myDFPlayer.sleep();     //sleep
//  myDFPlayer.reset();     //Reset the module
//  myDFPlayer.enableDAC();  //Enable On-chip DAC
//  myDFPlayer.disableDAC();  //Disable On-chip DAC
//  myDFPlayer.outputSetting(true, 15); //output setting, enable the output and set the
gain to 15

  //----Mp3 play----
  myDFPlayer.next();  //Play next mp3
  delay(1000);
  myDFPlayer.previous();  //Play previous mp3
  delay(1000);
  myDFPlayer.play(1);  //Play the first mp3
  delay(1000);
  myDFPlayer.loop(1);  //Loop the first mp3
  delay(1000);
  myDFPlayer.pause();  //pause the mp3
  delay(1000);
  myDFPlayer.start();  //start the mp3 from the pause
  delay(1000);
  myDFPlayer.playFolder(15, 4);  //play specific mp3 in SD:/15/004.mp3; Folder
Name(1~99); File Name(1~255)
  delay(1000);
  myDFPlayer.enableLoopAll(); //loop all mp3 files.
  delay(1000);
  myDFPlayer.disableLoopAll(); //stop loop all mp3 files.
  delay(1000);
  myDFPlayer.playMp3Folder(4); //play specific mp3 in SD:/MP3/0004.mp3; File
Name(0~65535)
  delay(1000);
  myDFPlayer.advertise(3); //advertise specific mp3 in SD:/ADVERT/0003.mp3; File
Name(0~65535)
  delay(1000);
  myDFPlayer.stopAdvertise(); //stop advertise
  delay(1000);
  myDFPlayer.playLargeFolder(2, 999); //play specific mp3 in SD:/02/004.mp3; Folder
Name(1~10); File Name(1~1000)
  delay(1000);
  myDFPlayer.loopFolder(5); //loop all mp3 files in folder SD:/05.
  delay(1000);
  myDFPlayer.randomAll(); //Random play all the mp3.
  delay(1000);
  myDFPlayer.enableLoop(); //enable loop.
  delay(1000);
  myDFPlayer.disableLoop(); //disable loop.
  delay(1000);

  //----Read imformation----
  Serial.println(myDFPlayer.readState()); //read mp3 state
  Serial.println(myDFPlayer.readVolume()); //read current volume
  Serial.println(myDFPlayer.readEQ()); //read EQ setting
  Serial.println(myDFPlayer.readFileCounts()); //read all file counts in SD card
  Serial.println(myDFPlayer.readCurrentFileNumber()); //read current play file number
  Serial.println(myDFPlayer.readFileCountsInFolder(3)); //read fill counts in folder
SD:/03
}

void loop()
{
  static unsigned long timer = millis();

  if (millis() - timer > 3000) {
    timer = millis();
    myDFPlayer.next();  //Play next mp3 every 3 second.
  }

  if (myDFPlayer.available()) {
```

```
    printDetail(myDFPlayer.readType(), myDFPlayer.read()); //Print the detail message
from DFPlayer to handle different errors and states.
  }
}

void printDetail(uint8_t type, int value){
  switch (type) {
    case TimeOut:
      Serial.println(F("Time Out!"));
      break;
    case WrongStack:
      Serial.println(F("Stack Wrong!"));
      break;
    case DFPlayerCardInserted:
      Serial.println(F("Card Inserted!"));
      break;
    case DFPlayerCardRemoved:
      Serial.println(F("Card Removed!"));
      break;
    case DFPlayerCardOnline:
      Serial.println(F("Card Online!"));
      break;
    case DFPlayerPlayFinished:
      Serial.print(F("Number:"));
      Serial.print(value);
      Serial.println(F(" Play Finished!"));
      break;
    case DFPlayerError:
      Serial.print(F("DFPlayerError:"));
      switch (value) {
        case Busy:
          Serial.println(F("Card not found"));
          break;
        case Sleeping:
          Serial.println(F("Sleeping"));
          break;
        case SerialWrongStack:
          Serial.println(F("Get Wrong Stack"));
          break;
        case CheckSumNotMatch:
          Serial.println(F("Check Sum Not Match"));
          break;
        case FileIndexOut:
          Serial.println(F("File Index Out of Bound"));
          break;
        case FileMismatch:
          Serial.println(F("Cannot Find File"));
          break;
        case Advertise:
          Serial.println(F("In Advertise"));
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }
}
```