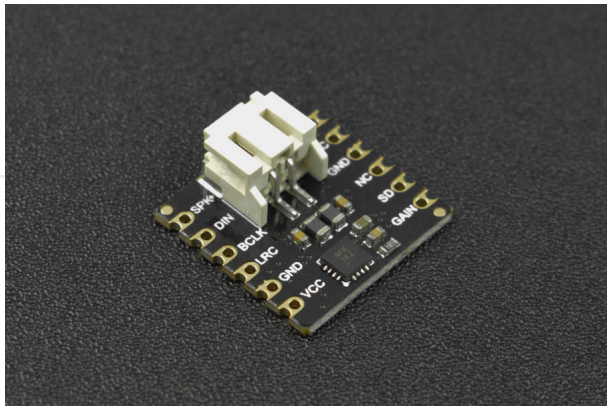
SKU:DFR0954 (https://www.dfrobot.com/product-2614.html)

(https://www.dfrobot.com/product-2614.html)



Introduction

This I2S digital amplifier based on the MAX98357 I2S Class D amplifier module works great with microcontrollers with I2S audio outputs like ESP32 and can be applied to projects like IoT nodes, smart home devices, Bluetooth audio players, and so on. The module supports 3.3V-5V power supply. And it can output over 2.5W of power when driven by 5V and a 4Ω speaker. Besides, it allows users to set different modes (off, left channel, right channel, and mixed) by changing the resistance on the SD port. The product defaults to be mixed mode. Featuring a small size, this I2S amplifier comes with stamp holes that enable it to be directly plugged into or mounted on a PCB and breadboard. Also, it is designed with PH2.0 and stamp holes for connecting speakers, flexible and convenient to use.

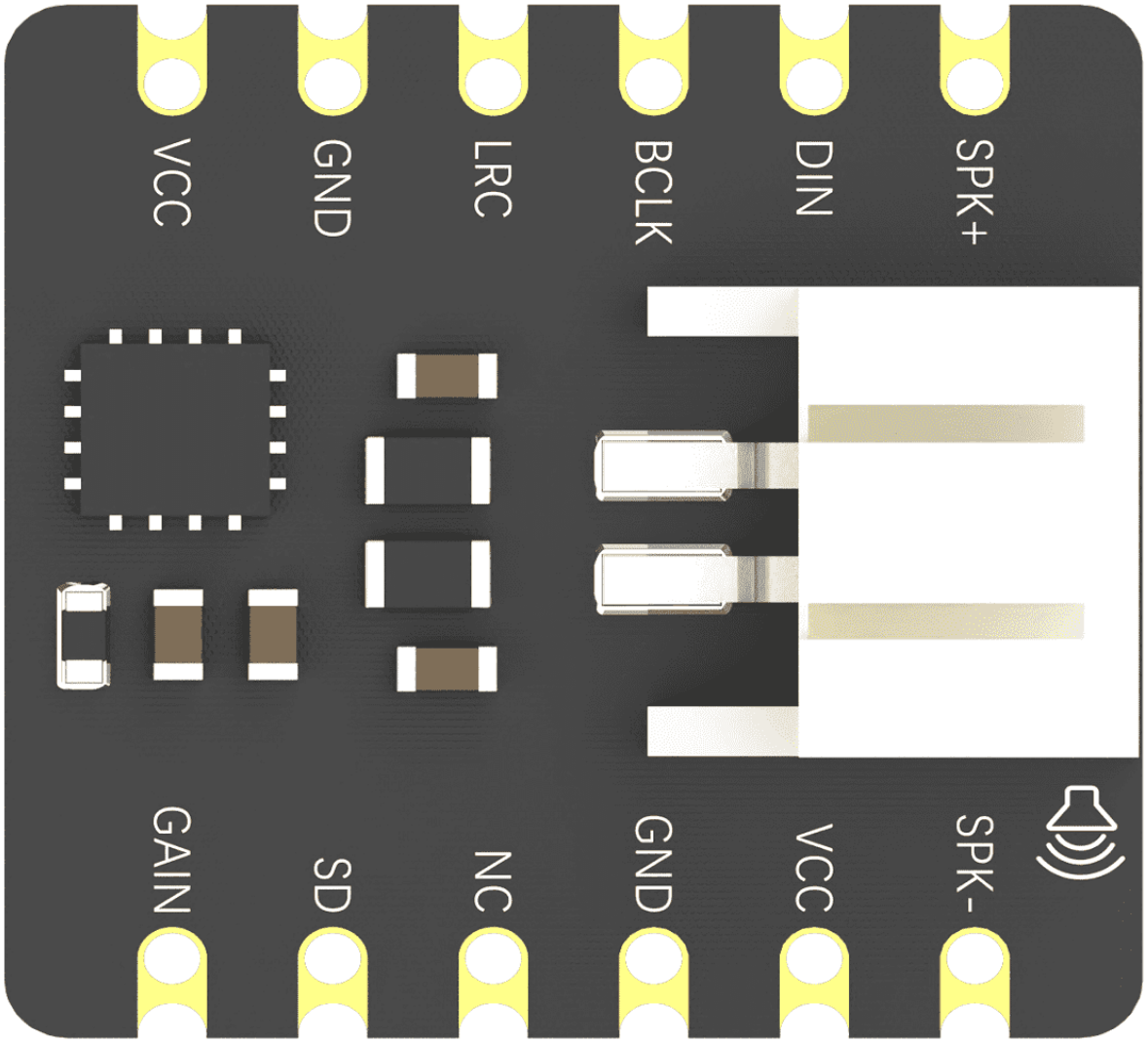
Features

- MAX98357 I2S digital Class D amplifier, power >2.5W
- 3.3V~5V voltage power supply
- Switchable sound channel
- Adjustable gain
- Stamp holes design, support SMT and DIP
- Two ways to connect a speaker: PH2.0 and stamp holes

Application

- Bluetooth Audio Player
- IoT Node Audio Player
- Smart Home Voice Interaction
- Robotic Device Audio Player

Board Overview



Pin	Function	Remark
SPK-	Speaker negative	4Ω/8Ω ＜3W

Pin	Function	Remark
VCC	Power input	DC 3.3V-5V
GND	Ground	
NC	Not connected pin	
SD	Turn off amplifier, adjust sound channel	Refer to 4.1.2(SD Mode Description)
GAIN	Adjust amplifier gain	Default to 9dB/Refer to 4.1.1 Gain Control Description
SPK+	Speaker positive	4Ω/8Ω ＜3W
DIN	Digital input signal	
BCLK	Bit clock input	Bit Clock Input
LRC	Frame clock	Left/right clock of I2S and LJ mode. Synchronized clock in TDM mode

Special Pin Function

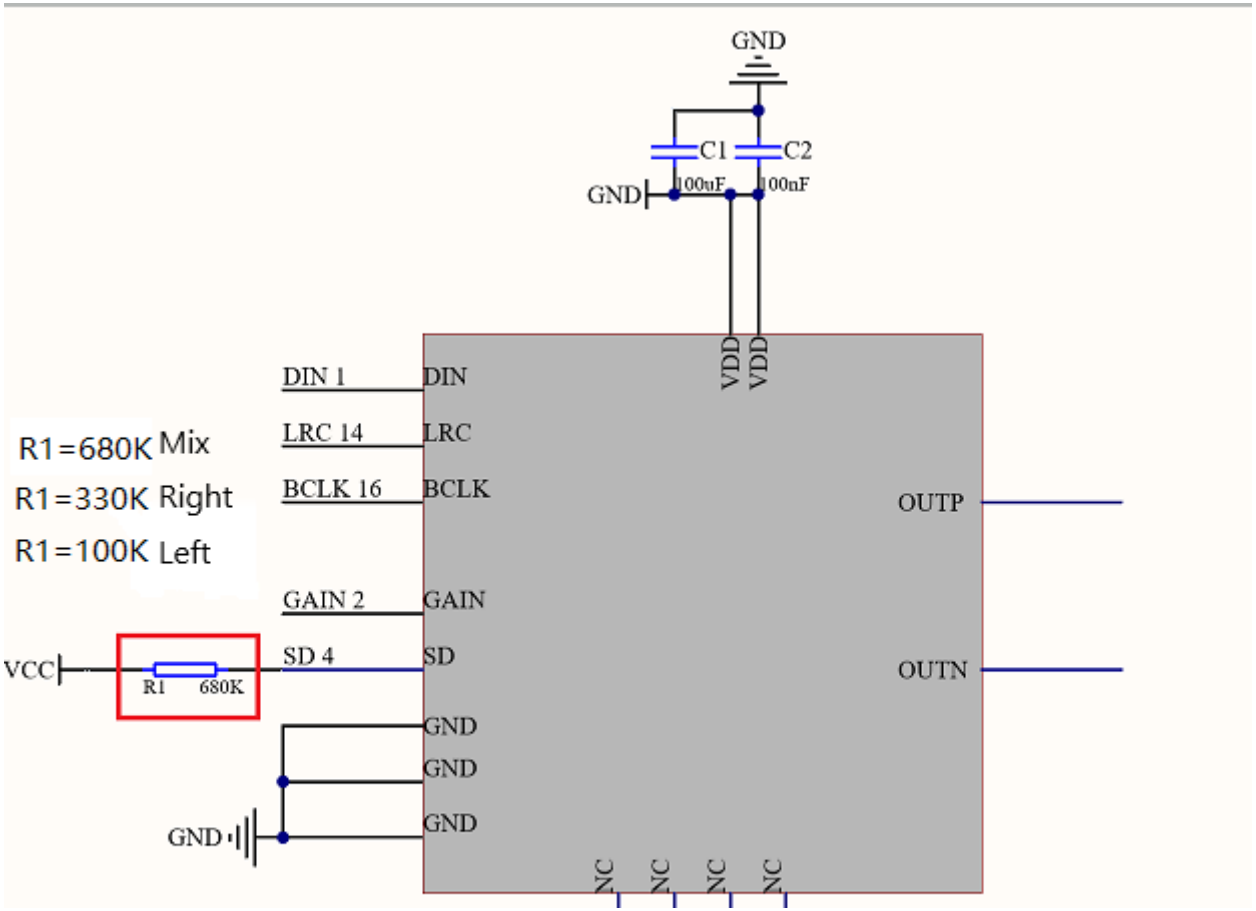
4.1.1 Gain Control(Gain)

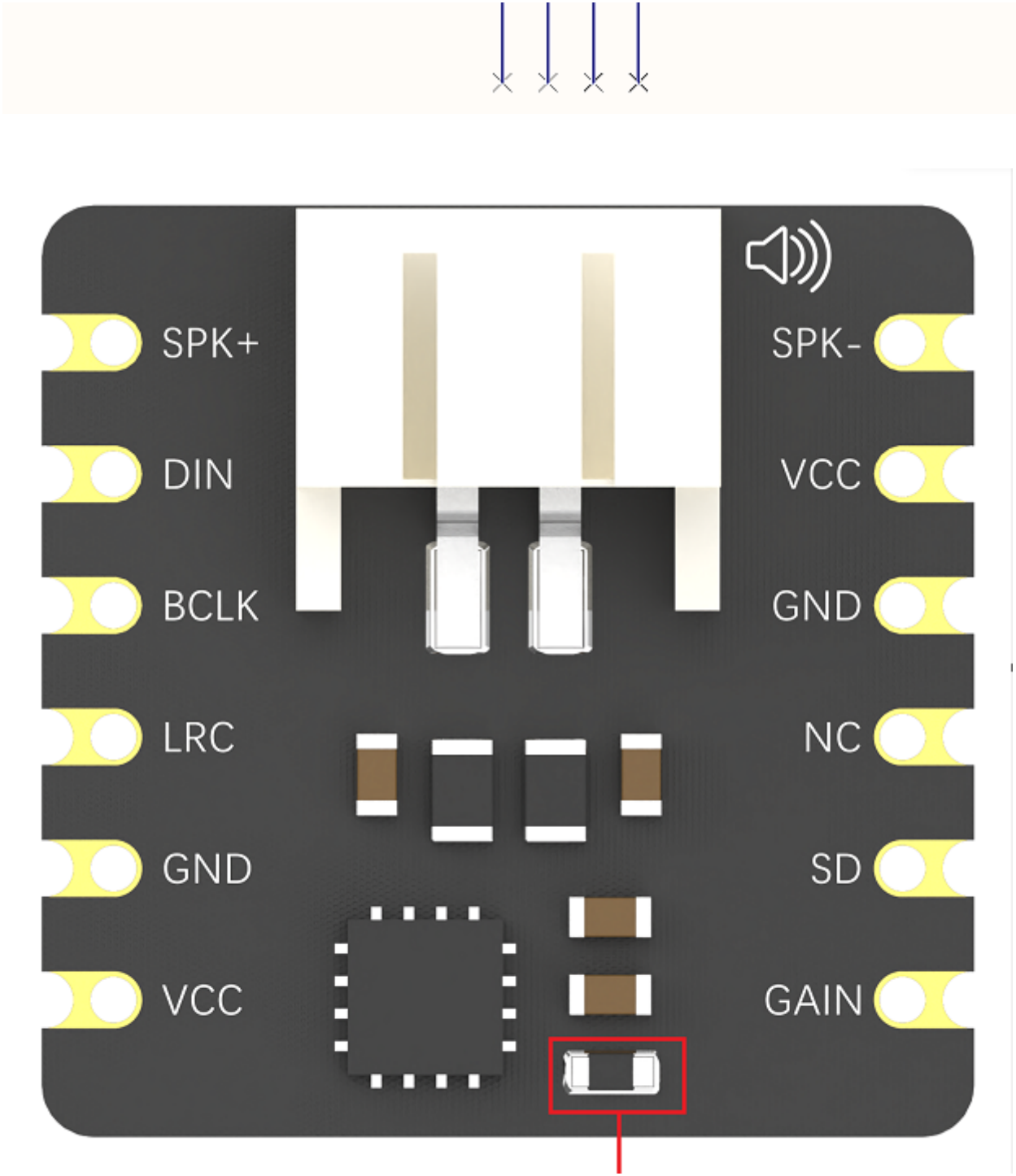
- Gain is 15dB when there is a 100K resistor connected between GAIN and GND
- Gain is 12dB when GAIN is directly connected to GND
- Gain is 9dB(Default status) when nothing is connected to GAIN
- Gain is 6dB when GAIN is directly connected to VCC

- Gain is 3dB when there is a 100k resistor connected between Gain and VCC

SD Mode

- Turn off amplifier when SD is connected to GND(Voltage below 0.16V)
- Output mix channel(Default) when the voltage on SD is between 0.16V to 0.77V, R1 resistance is 680k.
- Output right channel when the voltage on SD is between 0.77V to 1.4V, R1 resistance is 330K.
- Output left channel when the voltage on SD is over 1.4V, R1 resistance is 100K




SPK+
DIN
BCLK
LRC
GND
VCC
SPK-
VCC
GND
NC
SD
GAIN

Specification

Module Parameter

- MCU: MAX98357A
- Operating Voltage: DC 3.3V~5V
- Output Power: 8Ω 1.8W / 4Ω 2.5W (at 12db)
- Output Channel: single channel
- Communication Interface: I2S interface
- Operating Temperature: -40°C - ＋80°C
- Storage Temperature: -40°C - ＋80°C
- Dimension (Without Packaging): 18mm × 18mm/ 0.059 × 0.059

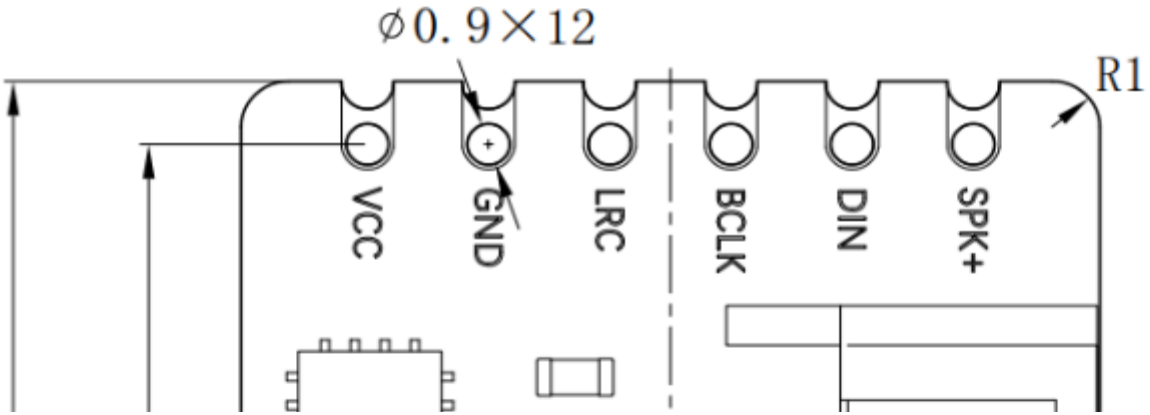
Chip Audio-relevant Parameters

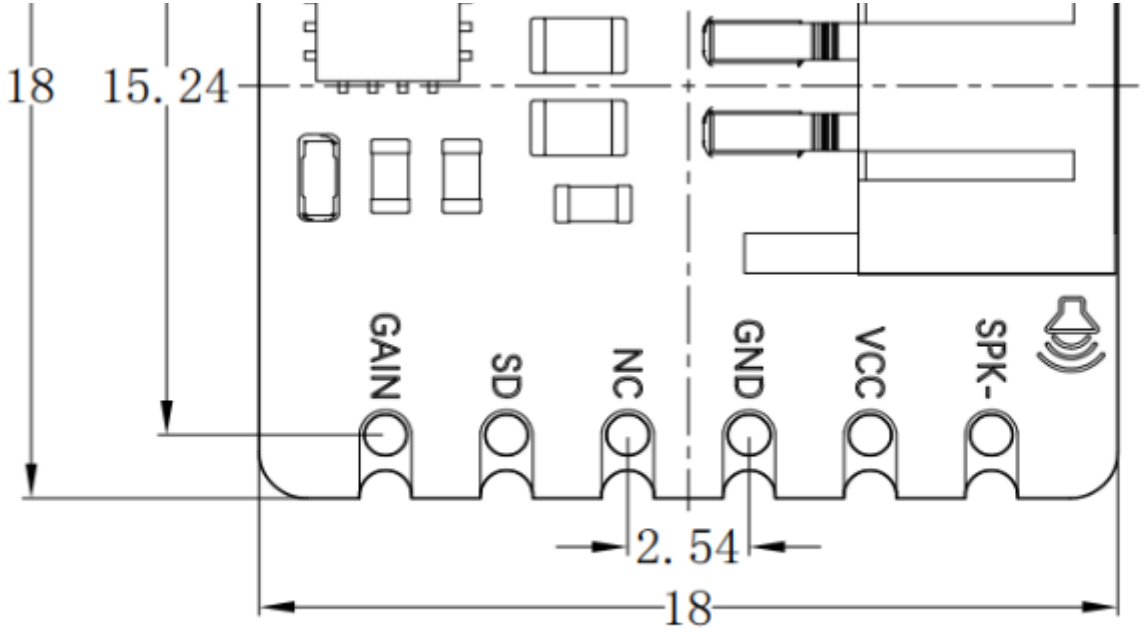
- VDD = 5V, VGND = 0V, GAIN_SLOT = VDD. BCLK = 3.072MHz, LRCLK = 48kHz
- speaker loads (ZSPK) connected between OUTP and OUTN, ZSPK = J, TA = TMIN to TMAX, unless otherwise noted. Typical values are at TA = +25NC.
- Sample Rate: 8kHz~96kHz
- THD:
  - 0.02% (f = 1kHz, POUT = 1W, TA = +25NC, ZSPK = 4I + 33FH, TQFN)
  - 0.013% (f = 1kHz, POUT = 0.5W, TA = +25NC, ZSPK = 8I + 68FH)
- Dynamic Range: 105DB (A-weighted, VRMS = 2.54V, 24- or 32-bit data)
- Output Noise: 25FVRMS (A-weighted, 24- or 32-bit data (Note 4))

- Output Noise: 25μVRMS (A-weighted, 24- or 32-bit data (Note 4))
- Gain (relative to reference level of 2.1dBV):
  - GAIN_SLOT = GND through 100kΩ: Min 14.4dB, Typical 15dB, Max 15.6dB
  - GAIN_SLOT = GND: Min 11.4dB, Typical 12dB, Max 12.6dB
  - GAIN_SLOT = unconnected: Min 8.4dB, Typical 9dB, Max 9.6dB
  - GAIN_SLOT = VDD: Min 5.4dB, Typical 6dB, Max 6.6dB
  - GAIN_SLOT = VDD through 100kΩ: Min 2.4dB, Typical 3dB, Max 3.6dB
- Efficiency: 92% (ZSPK = 8I + 68FH, THD+N = 10%, f = 1kHz, gain = 12dB)
- DAC Gain Error: 1%
- Frequency Response: ±0.2dB
- Class D Switching Frequency: 330kHz
- Spread Spectrum Bandwidth: ±12.5kHz

Product Dimension

Unit:mm





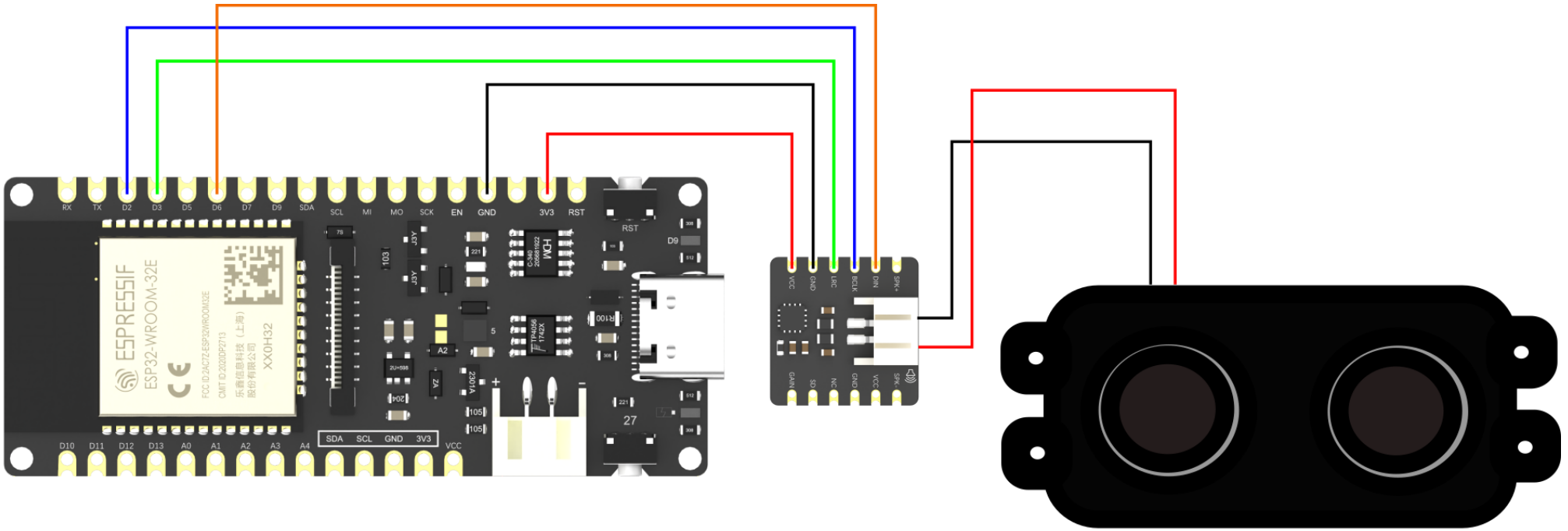
Tutorial 1 - Play music via Bluetooth

Requirements

- **Hardware**
  - Firebeetle ESP32-E(DFR0654) (https://www.dfrobot.com/product-2195.html) x 1
  - Speaker x1
  - Phone with Bluetooth function
  - M-M/F-M/F-F Jumper wires
- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Click the FireBeetle 2 ESP32-E Wikipage (https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654#target_6) to find SDK installtion tutorial
  - Download and install the **I2S Amplifer Library** (https://github.com/cdjq/DFRobot_MAX98357A) (About how to install the library?

(https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

Connection Diagram



Description:

I2S Module VCC to ESP32-E 3V3
I2S Module GND to ESP32-E GND
I2S Module LRC to ESP32-E 26/D3
I2S Module BCLK to ESP32-E 25/D2
I2S Module DIN to ESP32-E 14/D6
I2S Module SPK+ to Speaker power +
I2S Module SPK- to Speaker power -

I2S Module SPK- to Speaker power -

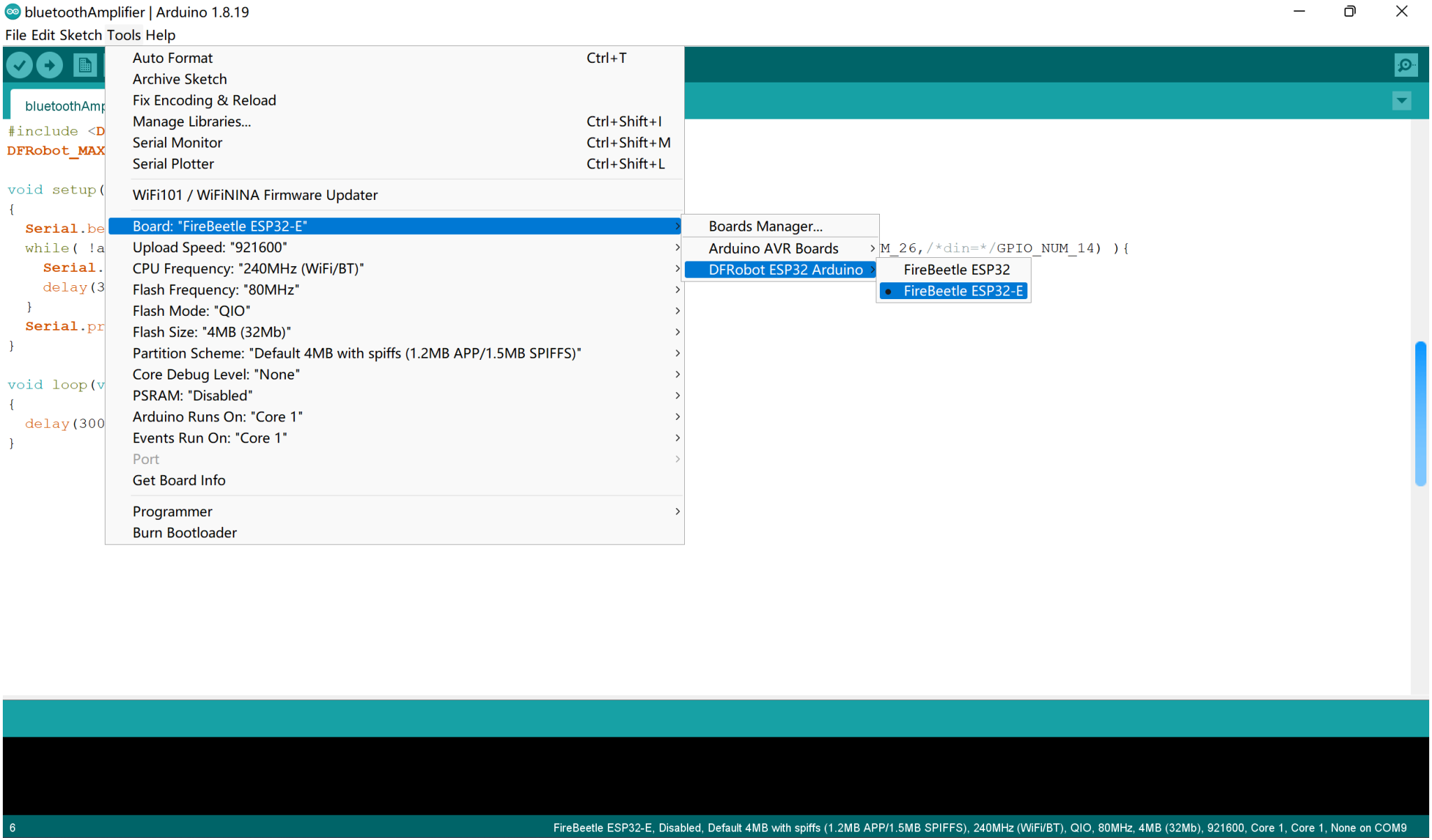
Sample Code

Enable ESP32-E Bluetooth function, create a Bluetooth node named bluetoothAmplifier, and use a phone with Bluetooth function to connect that node to play music.

```
#include <DFRobot_MAX98357A.h>
DFRobot_MAX98357A amplifier;

void setup(void)
{
  Serial.begin(115200);//Set serial rate to 115200
  while( !amplifier.begin(/*btName=*/"bluetoothAmplifier", /*bclk=*/GPIO_NUM_25, /*lrclk=*/GPIO_NUM_26,/*din=*/GPIO_NUM_14
    Serial.println("Initialize failed !");//No I2S pin signal detected, init failed
    delay(3000);
  }
  Serial.println("Initialize succeed!");//I2S pin signal detected, init succeeded
}

void loop(void)
{
  delay(3000);
}
```

Result

Open Arduino IDE, select board as "FireBeetle2 ESP32-E" and set upload rate to 921600


bluetoothAmplifier | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "FireBeetle ESP32-E"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino AVR Boards
DFRobot ESP32 Arduino
FireBeetle ESP32
FireBeetle ESP32-E
M_26,/*din=*/GPIO_NUM_14)){
6
FireBeetle ESP32-E, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None on COM9

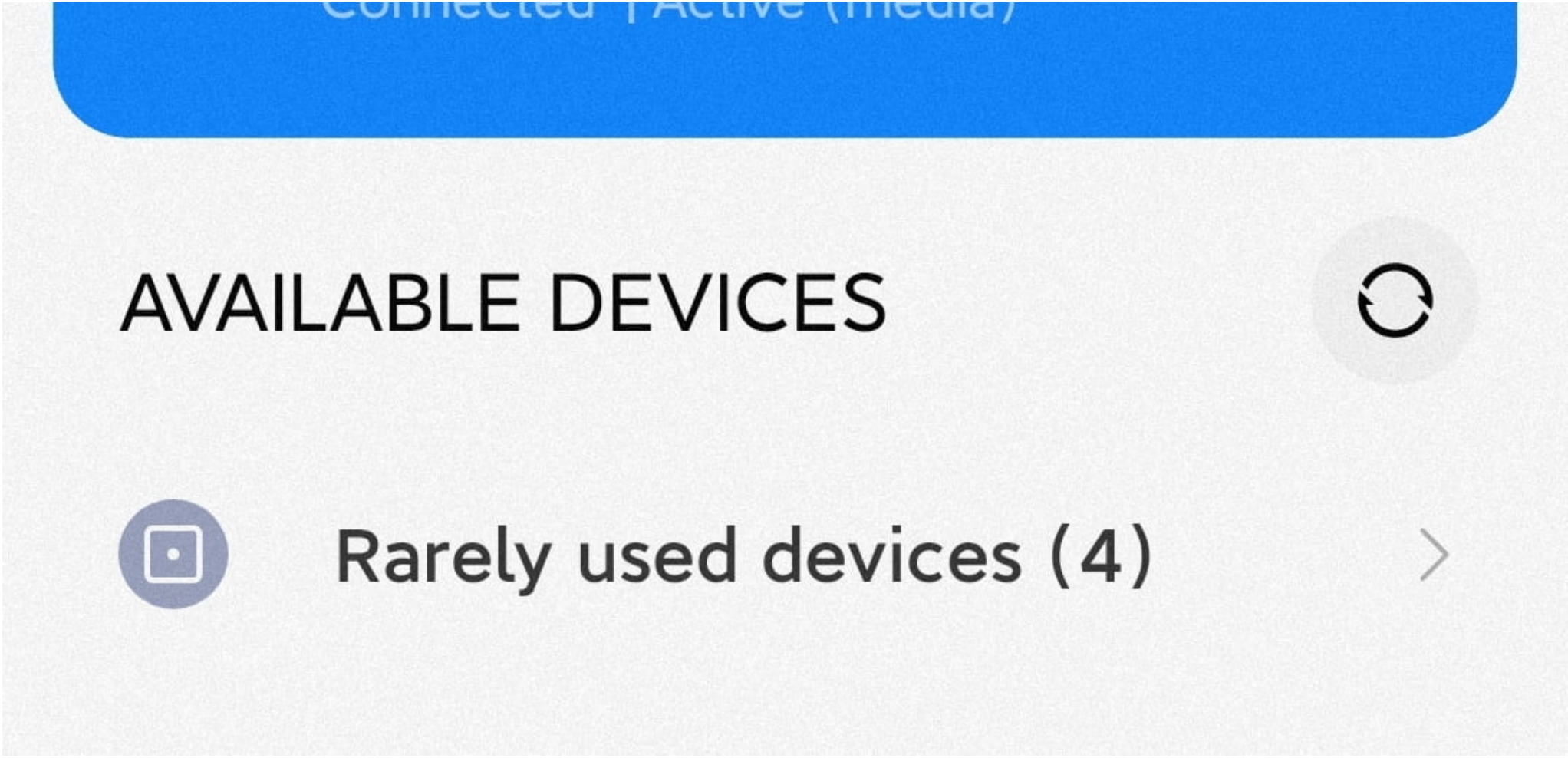
Copy the codes to Arduino IDE, and burn into FireBeetle2 ESP32-E.

bluetoothAmplifier §

```
#include <DFRobot_MAX98357A.h>
DFRobot_MAX98357A amplifier;

void setup(void)
{
  Serial.begin(115200);
  while( !amplifier.begin(/*btName=*/"bluetoothAmplifier", /*bclk=*/GPIO_NUM_25, /*lrclk=*/GPIO_NUM_26,/*din=*/GPIO_NUM_14) ){
    Serial.println("Initialize failed !");
    delay(3000);
  }
  Serial.println("Initialize succeed!");

void loop(void)
{
  delay(3000);
}
```

Turn on the mobile phone Bluetooth and connet to the bluetoothAmplifier node, then play muisc.


Bluetooth
Bluetooth
Device name
KFC
bluetoothAmplifier
SBC



Tutorial 2 - Play music from SD card

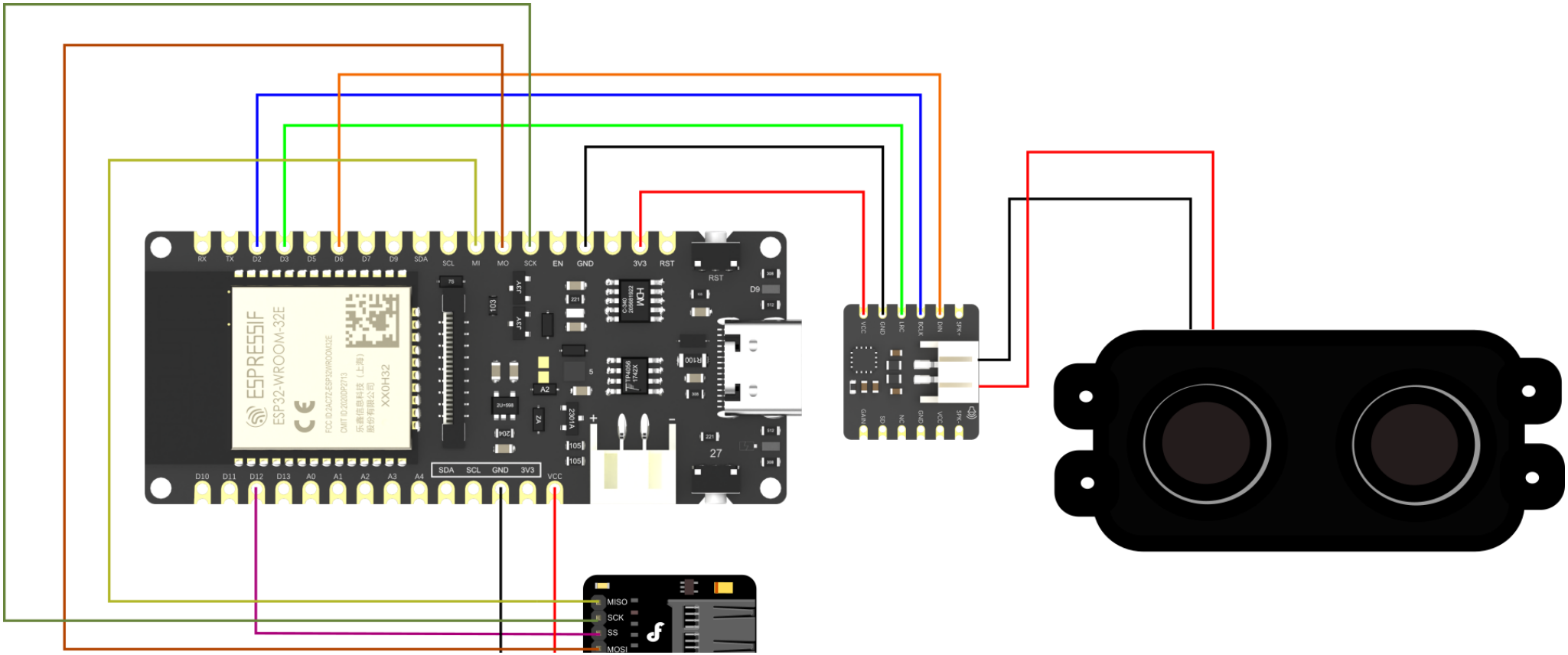
Requirements

- **Hardware**
  - Firebeetle ESP32-E(DFR0654) (https://www.dfrobot.com/product-2195.html) x 1
  - Speaker x1
  - MicroSD Module (DFR0229) (https://www.dfrobot.com/product-875.html) x 1
  - SD/MicroSD Memory Card (SKU: FIT0394) (https://www.dfrobot.com/product-1191.html) x1

- Card Reader ×1
- M-M/F-M/F-F Jumper wires

- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Click the FireBeetle 2 ESP32-E Wikipage (https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654#target_6) to find SDK installtion tutorial
  - Download and install the **I2S Ampilifer Library** (https://github.com/cdjq/DFRobot_MAX98357A) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

Connection Diagram



Description:

I2S Module VCC to ESP32-E 3V3
I2S Module GND to ESP32-E GND
I2S Module LRC to ESP32-E 26/D3
I2S Module BCLK to ESP32-E 25/D2
I2S Module DIN to ESP32-E 14/D6
SD card module 5V to ESP32-E VCC
SD card module GND to ESP32-E GND
SD card module MOSI to ESP32-E 23/MOSI
SD card module SS to ESP32-E 4/D12
SD card module SCL to ESP32-E 18/SCK
SD card module MISO to ESP32-E 19/MISO
I2S Module SPK+ to Speaker power +
I2S Module SPK- to Speaker power -

Sample Code

Connect to SD card module, then the music in the SD card can be played. Press RST button to play from the beginning,

```
#include <DFRobot_MAX98357A.h>

DFRobot_MAX98357A amplifier;

String musicList[100];   // SD card music playlist
void setup(void)
{
  Serial.begin(115200);
  while ( !amplifier.initI2S(/*_bclk=*/GPIO_NUM_25, /*_lrclk=*/GPIO_NUM_26, /*_din=*/GPIO_NUM_14) ){
    Serial.println("Initialize I2S failed !");
    delay(3000);
  }
  while (!amplifier.initSDCard(/*csPin=*/GPIO_NUM_4)){
    Serial.println("Initialize SD card failed !");
    delay(3000);
  }
  Serial.println("Initialize succeed!");
  amplifier.scanSDMusic(musicList);
  printMusicList();
  amplifier.setVolume(5);
  amplifier.closeFilter();
  amplifier.openFilter(bq_type_highpass, 500);
  amplifier.SDPlayerControl(SD_AMPLIFIER_PLAY);
  delay(5000);
  if(musicList[1].length()){
    Serial.println("Changing Music...\n");
    amplifier.playSDMusic(musicList[1].c_str());
  }else{
    Serial.println("The currently selected music file is incorrect!\n");
  }
}
```

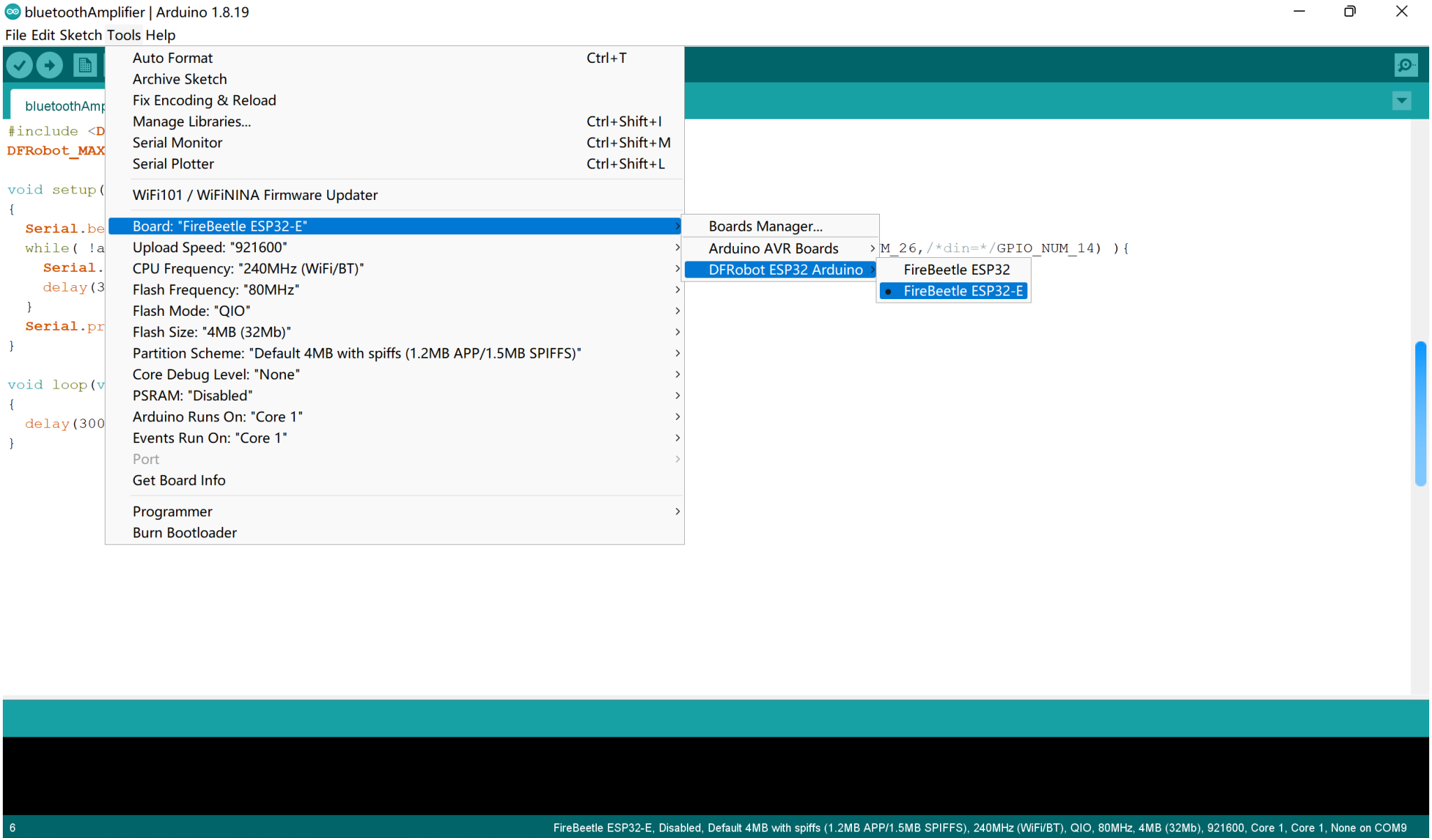
```
}
void loop()
{
  parseSerialCommand();
  delay(500);

}
void printMusicList(void)
{
  uint8_t i = 0;
  if(musicList[i].length()){
    Serial.println("\nMusic List: ");
  }else{
    Serial.println("The SD card audio file scan is empty, please check whether there are audio files in the SD card that m
  }

  while(musicList[i].length()){
    Serial.print("\t");
    Serial.print(i);
    Serial.print("  -  ");
    Serial.println(musicList[i]);
    i++;
  }
}
void parseSerialCommand(void)
{
  String cmd;
  float value;
  if(Serial.available()){
    cmd = Serial.readStringUntil('-');

    if(cmd.equals("hp")){
      Serial.println("Setting a High-Pass filter...\n");
      value =Serial.parseFloat();
      amplifier.openFilter(bq_type_highpass, value);
    }else if(cmd.equals("lp")){
      Serial.println("Setting a Low-Pass filter...\n");
```

```
    value =Serial.parseFloat();
    amplifier.openFilter(bq_type_lowpass, value);
  }else if(cmd.equals("closeFilter")){
    Serial.println("Closing filter...\n");
    amplifier.closeFilter();

  }else if(cmd.equals("vol")){
    Serial.println("Setting volume...\n");
    value =Serial.parseFloat();
    amplifier.setVolume(value);
  }else if(cmd.equals("start")){
    Serial.println("starting amplifier...\n");
    amplifier.SDPlayerControl(SD_AMPLIFIER_PLAY);
  }else if(cmd.equals("pause")){
    Serial.println("Pause amplifier...\n");
    amplifier.SDPlayerControl(SD_AMPLIFIER_PAUSE);
  }else if(cmd.equals("stop")){
    Serial.println("Stopping amplifier...\n");
    amplifier.SDPlayerControl(SD_AMPLIFIER_STOP);
  }else if(cmd.equals("musicList")){
    Serial.println("Scanning music list...\n");
    amplifier.scanSDMusic(musicList);
    printMusicList();
  }else if(cmd.equals("changeMusic")){
    cmd = musicList[Serial.parseInt()];
    if(cmd.length()){
      Serial.println("Changing Music...\n");
      amplifier.playSDMusic(cmd.c_str());
    }else{
      Serial.println("The currently selected music file is incorrect!\n");
    }

  }else{
    Serial.println("Help : \n \
    Currently available commands (format: cmd-value):\n \
      Start playback: e.g. start-\n \
      Pause playback: e.g. pause-\n \
```

```
        Stop playback: e.g. stop-\n \
        Print music list: e.g. musicList-\n \
        Change songs according to the music list: e.g. changeMusic-1\n \
        Set and open high-pass filter: e.g. hp-500\n \
        Set and open low-pass filter: e.g. lp-15000\n \
        Close filter: e.g. closeFilter-\n \
        Set volume: e.g. vol-5.0\n \
      For the detailed meaning, please refer to the code comments of this demo.\n");
    }
    while(Serial.read() >= 0);
  }
}
```

Result

Open Arduino IDE, select board as "FireBeetle2 ESP32-E" and set upload rate to 921600


bluetoothAmplifier | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "FireBeetle ESP32-E"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino AVR Boards
DFRobot ESP32 Arduino
FireBeetle ESP32
FireBeetle ESP32-E
6
FireBeetle ESP32-E, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None on COM9

Copy the codes into Arduino IDE, burn into FireBeetle2 ESP32-E, the music files the SD card will be played automatically.

API Function

```
/**
   * @fn begin
   * @brief Init function
   * @param btName - Created Bluetooth device name
   * @param bclk - I2S communication pin number, serial clock (SCK), aka bit clock (BCK)
   * @param lrclk - I2S communication pin number, word select (WS), i.e. command (channel) select, used to switch between
   * @param din - I2S communication pin number, serial data signal (SD), used to transmit audio data in two's complement f
   * @return true on success, false on error
   */
  bool begin(const char *btName="bluetoothAmplifier",
             int bclk=GPIO_NUM_25,
             int lrclk=GPIO_NUM_26,
             int din=GPIO_NUM_27);

  /**
   * @fn initI2S
   * @brief Initialize I2S
   * @param _bclk - I2S communication pin number, serial clock (SCK), aka bit clock (BCK)
   * @param _lrclk - I2S communication pin number, word select (WS), i.e. command (channel) select, used to switch between
   * @param _din - I2S communication pin number, serial data signal (SD), used to transmit audio data in two's complement
   * @return true on success, false on error
   */
  bool initI2S(int _bclk, int _lrclk, int _din);

  /**
   * @fn initBluetooth
   * @brief Initialize bluetooth
   * @param  btName - The created Bluetooth device name
```

```
   * @return true on success, false on error
   */
  bool initBluetooth(const char * _btName);


  /**
   * @fn initSDCard
   * @brief Initialize SD card
   * @param csPin The number of the cs pin for spi communication of SD card module
   * @return true on success, false on error
   */
  bool initSDCard(uint8_t csPin=GPIO_NUM_5);

/*************************** Function ******************************/

  /**
   * @fn scanSDMusic
   * @brief Scan the music files in WAV format in the SD card
   * @param musicList - The music files in WAV format scanned from the SD card. Type: character string array.
   * @return None
   * @note Only support English for path name of music files and WAV for their format currently.
   */
  void scanSDMusic(String * musicList);

  /**
   * @fn playSDMusic
   * @brief Play music files in the SD card
   * @param Filename - music file name, only support the music files in .wav format currently
   * @note Music file name must be an absolute path like /musicDir/music.wav
   * @return None
   * @note Only support English for path name of music files and WAV for their format currently.
   */
  void playSDMusic(const char *Filename);

  /**
   * @fn SDPlayerControl
```

```
     * @brief SD card music playback control interface
     * @param CMD - playback control command:
     * @n SD_AMPLIFIER_PLAY: start to play music, which can be played from the position where you paused before
     * @n   If no music file is selected through playSDMusic(), the first one in the playlist will be played by default.
     * @n   Playback error may occur if music files are not scanned from SD card in the correct format (only support English
     * @n SD_AMPLIFIER_PAUSE: pause playback, retain the playback position of the current music file
     * @n SD_AMPLIFIER_STOP: stop playback, end the current music playback
     * @return None
     */
    void SDPlayerControl(uint8_t CMD);

    /**
     * @fn getMetadata
     * @brief Get "metadata" through AVRC command
     * @param type - The type of metadata to be obtained, and the parameters currently supported:
     * @n     ESP_AVRC_MD_ATTR_TITLE   ESP_AVRC_MD_ATTR_ARTIST   ESP_AVRC_MD_ATTR_ALBUM
     * @return The corresponding type of "metadata"
     */
    String getMetadata(uint8_t type);

    /**
     * @fn getRemoteAddress
     * @brief Get address of Bluetooth device remotely
     * @note The address will be obtained after the module is paired with the remote Bluetooth device and successfully commu
     * @return Return the array pointer storing the address of the remote Bluetooth device
     * @n Return None when the module does not connect to the remote device or failed to communicate with it based on the Blu
     * @n AVRCP(Audio Video Remote Control Profile)
     */
    uint8_t * getRemoteAddress(void);

    /**
     * @fn setVolume
     * @brief Set volume
     * @param vol - Set volume, the range can be set to 0-9
     * @note 5 for the original volume of audio data, no increase or decrease
     * @return None
```

```
     */
    void setVolume(float vol);

    /**
     * @fn openFilter
     * @brief Enable audio filter
     * @param type - bq_type_highpass: enable high-pass filtering; bq_type_lowpass: enable low-pass filtering
     * @param fc - Threshold of filtering, range: 2-20000
     * @note For example, setting high-pass filter mode and the threshold of 500 indicates to filter out the audio signal be
     * @return None
     */
    void openFilter(int type, float fc);

    /**
     * @fn closeFilter
     * @brief Disable the audio filter
     * @return None
     */
    void closeFilter(void);

    /**
     * @fn reverseLeftRightChannels
     * @brief Reverse left and right channels, When you find that the left
     * @n  and right channels play opposite, you can call this interface to adjust
     * @return None
     */
    void reverseLeftRightChannels(void);
```

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/).

More Documents

- DFR0954 STP 3D Model.rar (https://dfimg.dfrobot.com/nobody/wiki/e7582a3da5a86fdbd2a7528805bde2cf.rar)
- DFR0954 2D CAD Dimension.rar (https://dfimg.dfrobot.com/nobody/wiki/ce0fdc8cab28133ccf25c561e94e697a.rar)
- DFR0954 2D PDF Dimension.pdf (https://dfimg.dfrobot.com/nobody/wiki/7697624de663548e6e41a8d6aa79f04c.pdf)
- DFR0954 Schematics.pdf (https://dfimg.dfrobot.com/nobody/wiki/9d4e5e3d16530ea5b6ba8f1385fffc4d.pdf)
- MAX98357A Datasheet.pdf (https://dfimg.dfrobot.com/nobody/wiki/d75ded846393befb317bcf0dcce001e7.pdf)

Get **MAX98357 I2S Amplifier Module** (https://www.dfrobot.com/product-2614.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/distributor)

Turn to the Top