

FireBeetle 2 ESP32-C5之火车上的MP3

作者: HonestQiao / 乔楚

日期: 2025-10-24

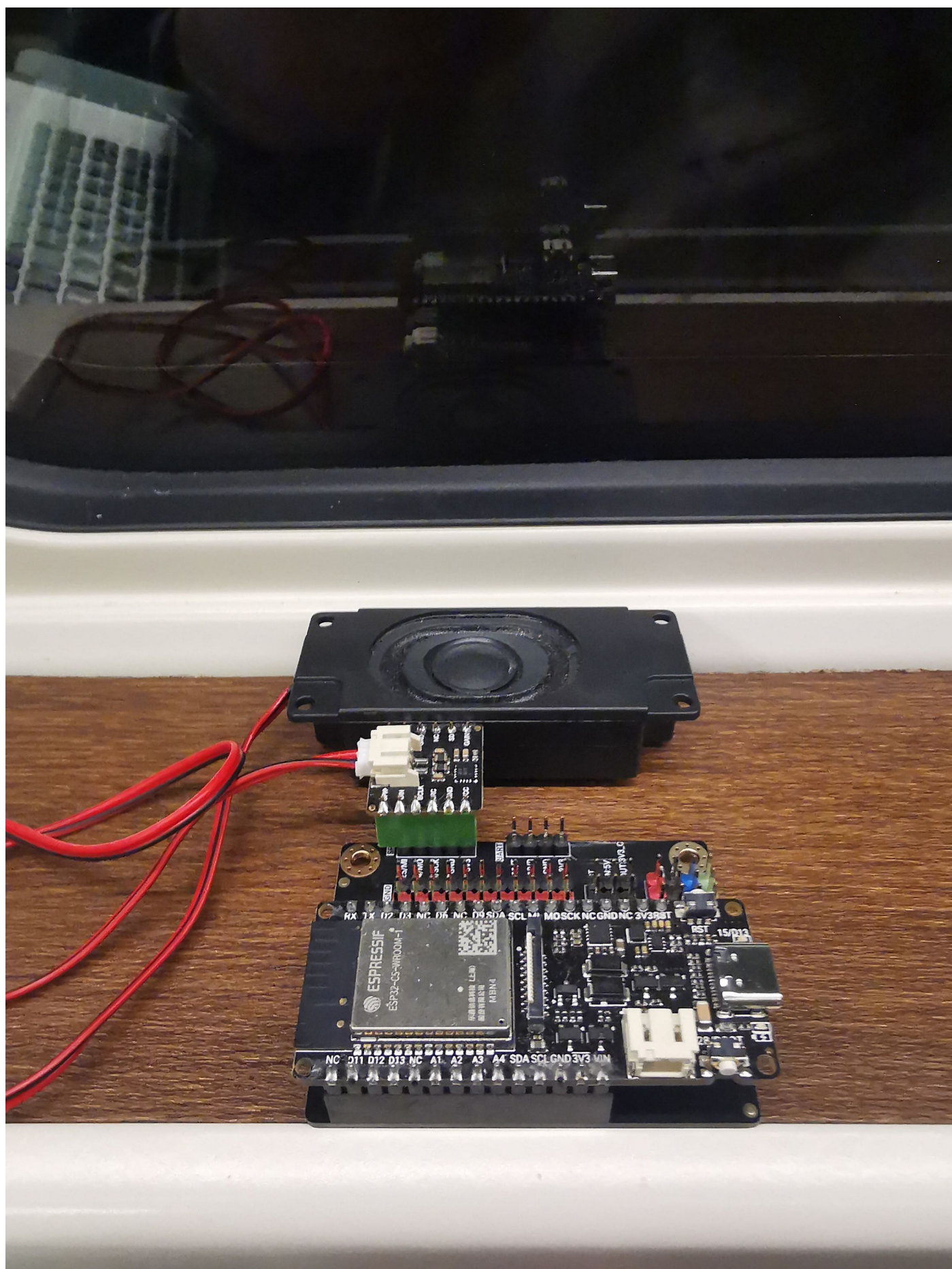
首发: <https://mc.dfrobot.com.cn/thread-398518-1-1.html>

今天又坐火车，老板居然给我订了张无座票，说上车后能帮我升级座位。结果乘务员过来后，告知没有空座了，看来得站好几个小时了。

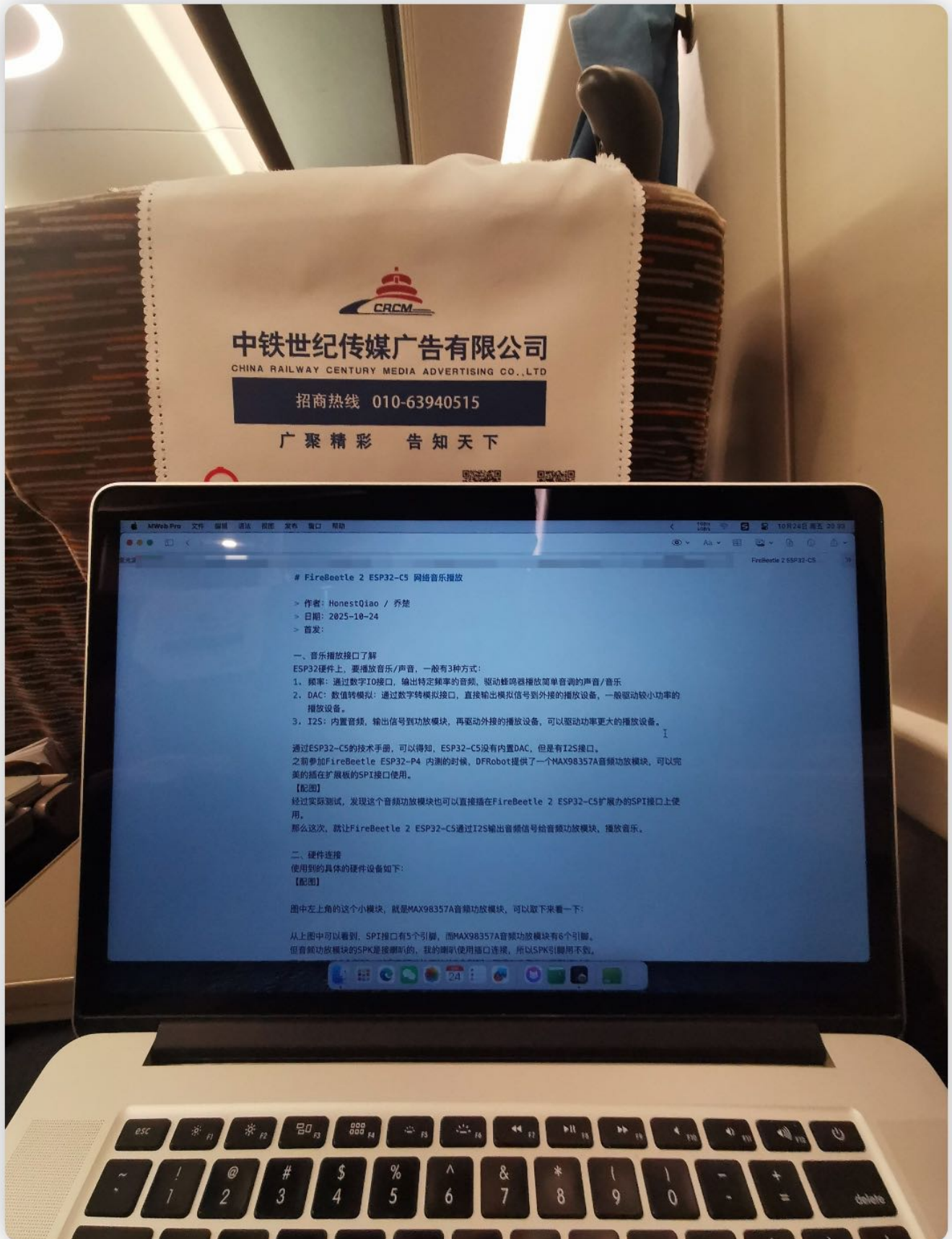


后来，陆续有人上下车，我也觅得几次坐下的机会，不过都不长久，过一两站就要让出位置来了。

再后来，快要到终点的3站，时间比较长，而且也没有人上来，于是我总算安定下来了，在包里掏东西，又掏出了装FireBeetle 2 ESP32-C5板子的盒子，那就再做点什么吧。



由于时间也不是很多了，所以整了整 FireBeetle 2 ESP32-C5 上通过网络播放MP3。



一、音乐播放接口了解

ESP32硬件上，要播放音乐/声音，一般有3种方式：

1. 频率输出：通过数字IO口，输出特定的频率，就能驱动蜂鸣器播放简单音调。
2. DAC：有的ESP32芯片内置了8位数模转换器(DAC)，可用于直接输出模拟音频信号，驱动较小功率的播放设备，适用于播放低质量的提示音等简单音频。
3. PWM：使用ESP32 的内部PWM 模块生成音频信号，通常用于对音质要求不高的应用，如播放提示音和简单的铃声。
4. I2S：I2S（Inter-IC Sound，集成电路内置音频总线）是一种数字音频接口，可以连接外部数字音频设备，如音频解码器或功放，再驱动外接的播放设备，可以驱动功率更大的播放设备，适用于播放高质量的音频。

通过ESP32-C5的技术手册，可以得知，ESP32-C5有I2S接口：

第 31 章

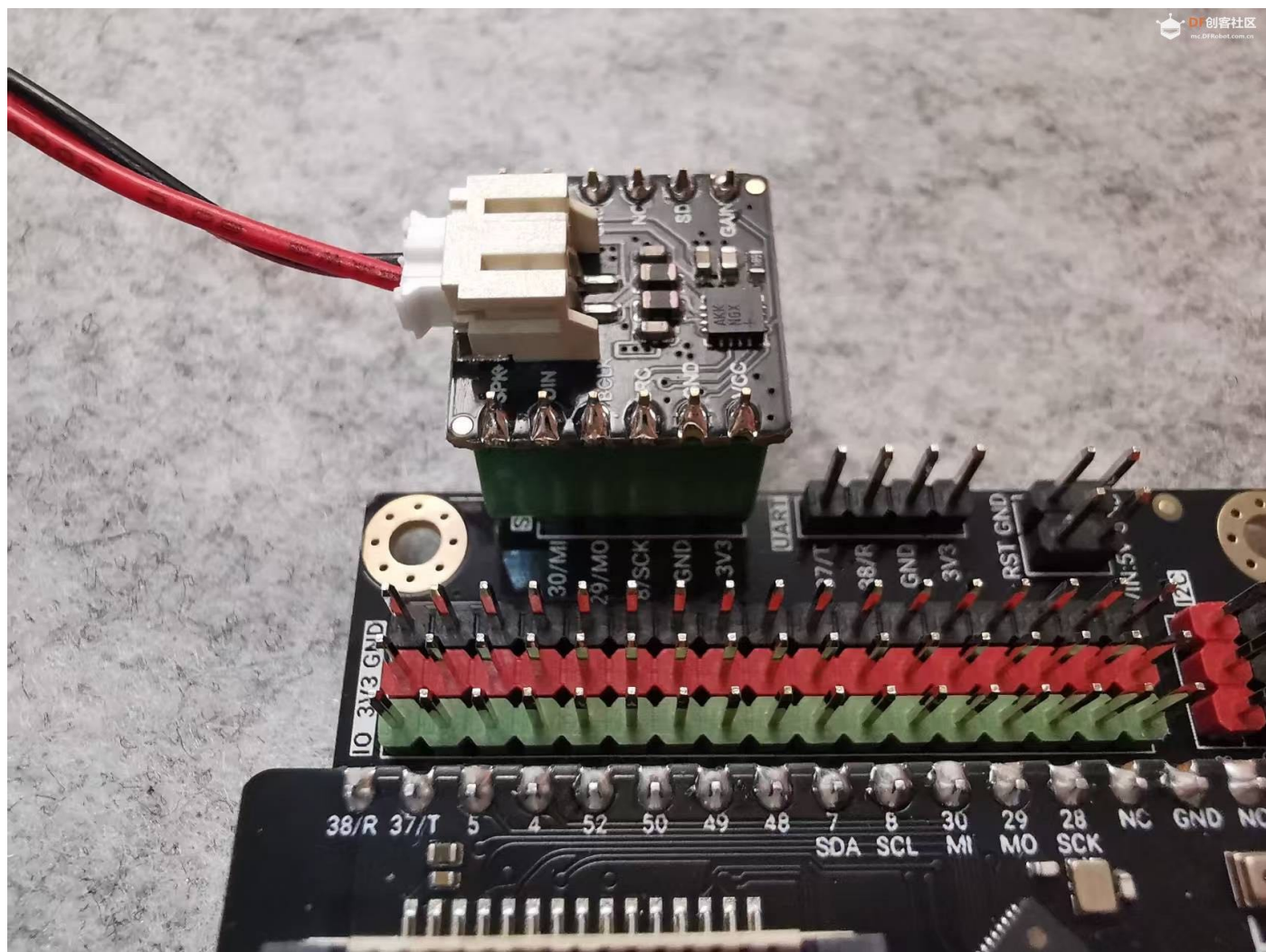
I2S 控制器 (I2S)

31.1 概述

ESP32-C5 内置一个 I2S 接口，为多媒体应用，尤其是为数字音频应用提供了灵活的数据通信接口。

I2S 标准总线定义了三种信号——串行时钟信号 BCK、字选择信号 WS 和串行数据信号 SD。一个基础的 I2S 数据总线包含一个主机和一个从机。主机和从机的角色在通信过程中保持不变。ESP32-C5 的 I2S 模块包含独立的 TX 模块和 RX 模块，能够保证优良的通信性能。

之前参加 FireBeetle ESP32-P4 内测的时候，DFRobot提供了一个MAX98357A音频功放模块，可以很方便地直接插在扩展板的I2S接口（物理上复用SPI插座）上使用。

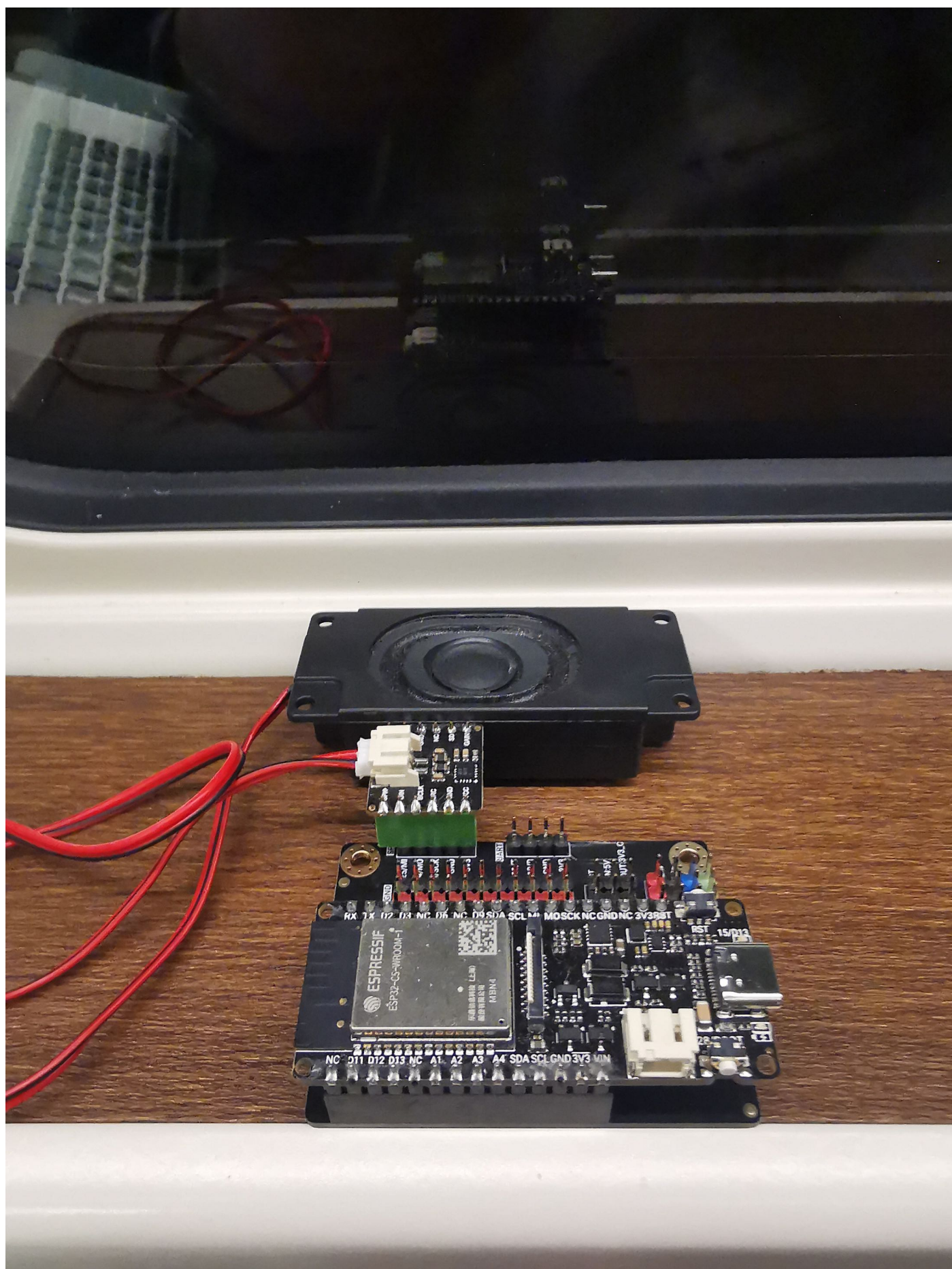


经过实际测试，发现这个音频功放模块也可以直接插在FireBeetle 2 ESP32-C5扩展板的SPI接口上使用。

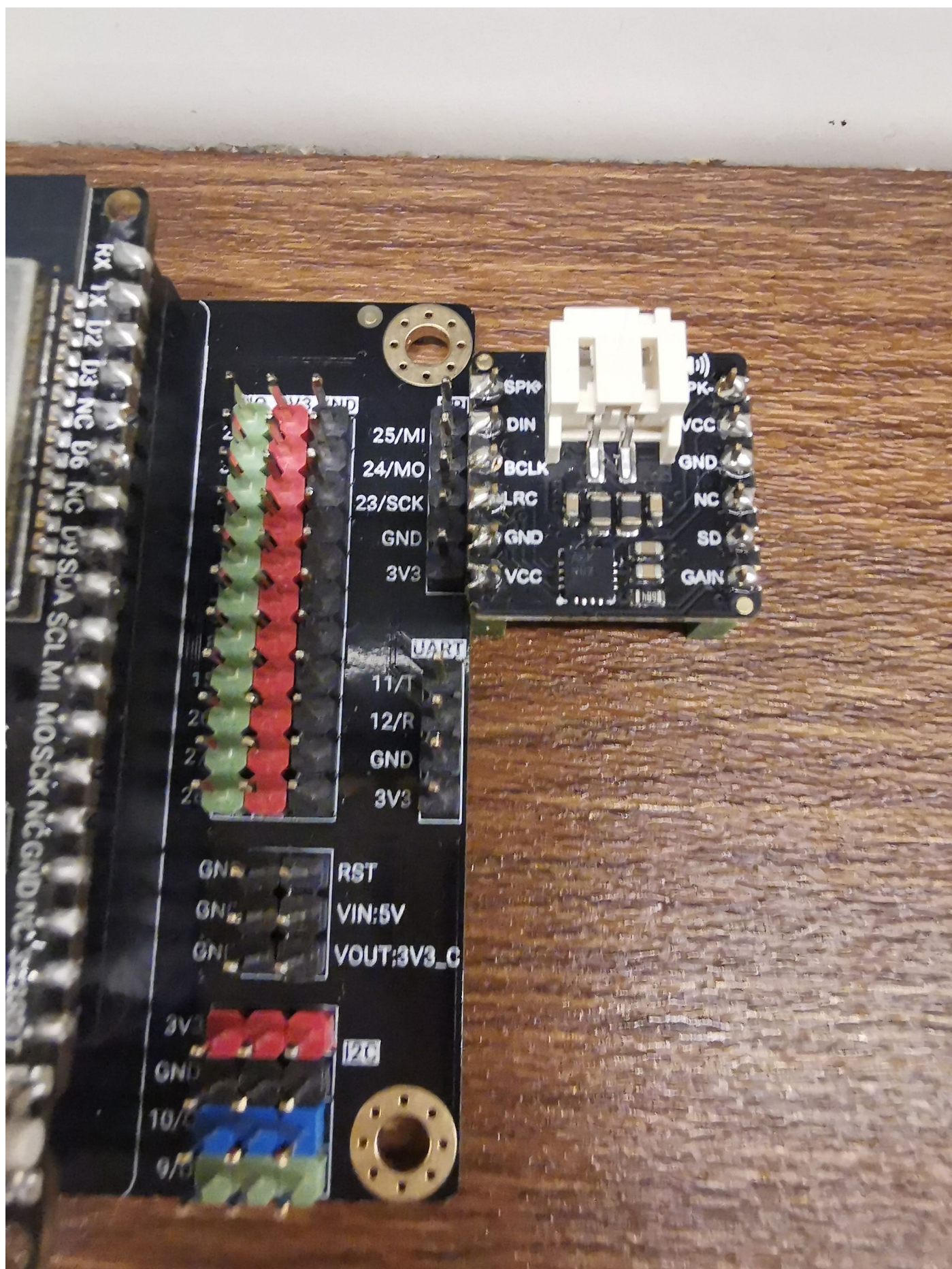
那么这次，就让FireBeetle 2 ESP32-C5通过I2S输出音频信号给音频功放模块，播放音乐。

二、硬件连接

使用到的具体的硬件设备如下：



图中的这个小模块，就是之前用FireBeetle ESP32-P4上的MAX98357A音频功放模块，可以取下来看一下：



扩展板上的SPI插座（用于I2S）有5个引脚，而MAX98357A音频功放模块有6个引脚。其中，音频功放模块的SPK引脚用于连接喇叭；由于我的喇叭使用独立插口，因此其SPK引脚在此次连接中并未使用。

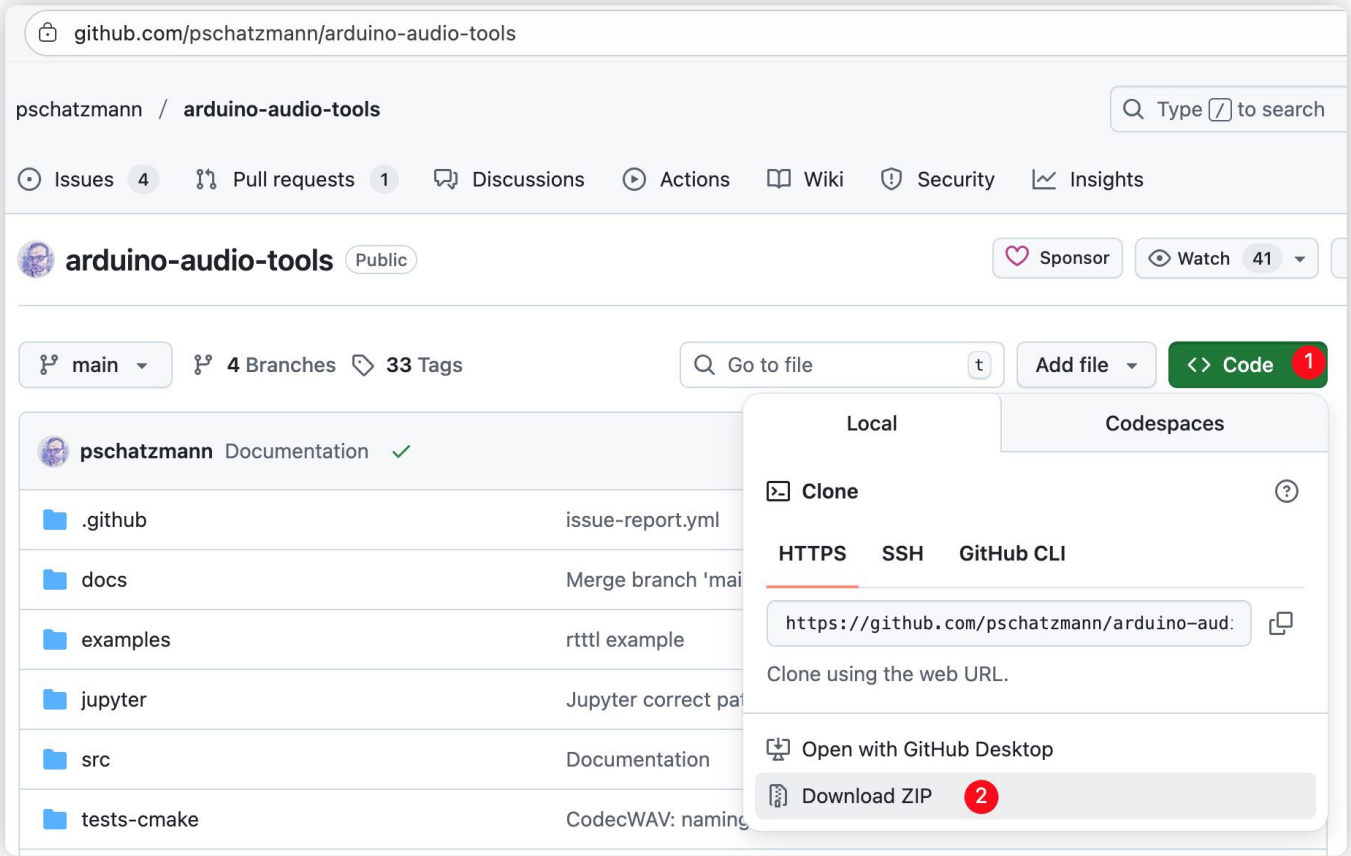
因此，扩展板的SPI插座的5个引脚，对应音频功放模块的5个引脚，而且3V3/VCC和GND都刚好对应。

这样从3V3、VCC、GND开始，直接插上去，就直接可以使用了。
最终的引脚对应关系如下：

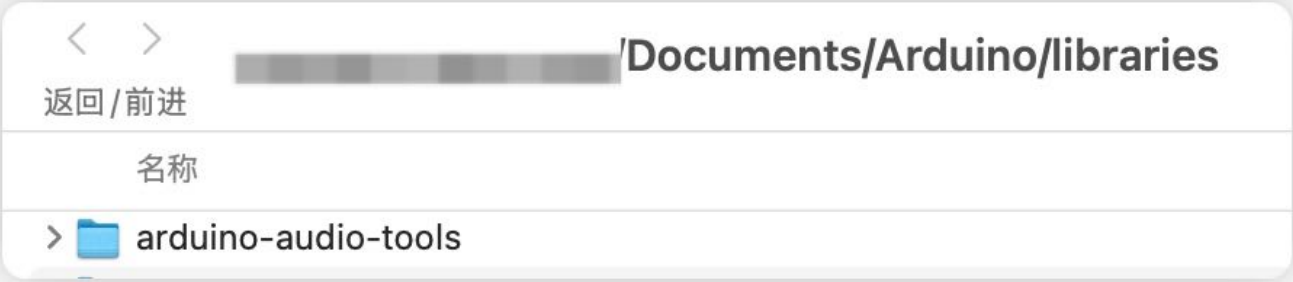
FireBeetle 2 ESP32-C5扩展板	MAX98357A音频功放模块
MISO	DIN
MOSI	BCLK
SCK	LRC
GND	GND
3V3	VCC

三、音频扩展库

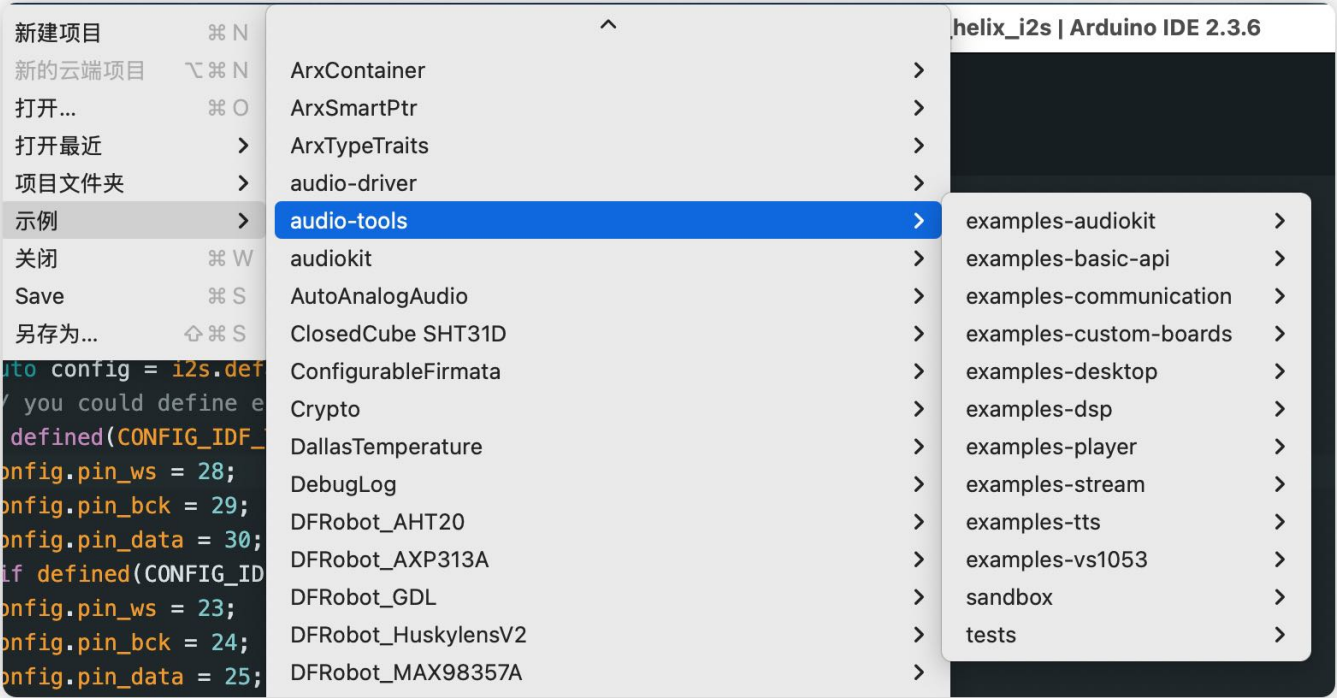
在这次的网络音乐播放中，我使用了arduino-audio-tools扩展库。
这个库是由pschatzmann开发的一个超级强大的支持Arduino环境的音频库，其功能强大，我所用到的不过是冰山一角。
但这个库，在Arduino IDE中不能直接找到，需要按照如下方式下载：



下载完成后，放置到Arduino库目录下面：



再次打开Arduino IDE，就可以看到arduino-audio-tools的示例：



另外，要支持MP3文件，还需 `arduino-libhelix` 这个库，按照如下方式下载：

github.com/pschatzmann/arduino-libhelix

pschatzmann / **arduino-libhelix** Type / to search

Issues Pull requests Discussions Actions Security Insights

arduino-libhelix Public Sponsor Watch 2

main 2 Branches 16 Tags Go to file Add file Code 1

pschatzmann Merge pull request #36 from A-KL/aac-enable-codec-t

.github	github config
docs/html	idf logging support
examples	compile error for p
src	Enabling library bu
.gitignore	Compile warnings
CMakeLists.txt	HELIX_LOGGING_A
Doxyfile	doxygen

last year

Local

Codespaces

Clone

HTTPS SSH GitHub CLI

https://github.com/pschatzmann/arduino-libhelix

Clone using the web URL.

Open with GitHub Desktop

Download ZIP 2

下载完成后，同样放置到Arduino库目录下面：

github.com/pschatzmann/arduino-libhelix

pschatzmann / **arduino-libhelix** Type / to search

Issues Pull requests Discussions Actions Security Insights

arduino-libhelix Public Sponsor Watch 2

main 2 Branches 16 Tags Go to file Add file Code

pschatzmann Merge pull request #36 from A-KL/aac-enable-codec-t

.github	github config
docs/html	idf logging support
examples	compile error for p
src	Enabling library bu
.gitignore	Compile warnings
CMakeLists.txt	HELIX_LOGGING_A
Doxyfile	doxygen

last year

Local

Codespaces

Clone

HTTPS SSH GitHub CLI

https://github.com/pschatzmann/arduino-libhelix

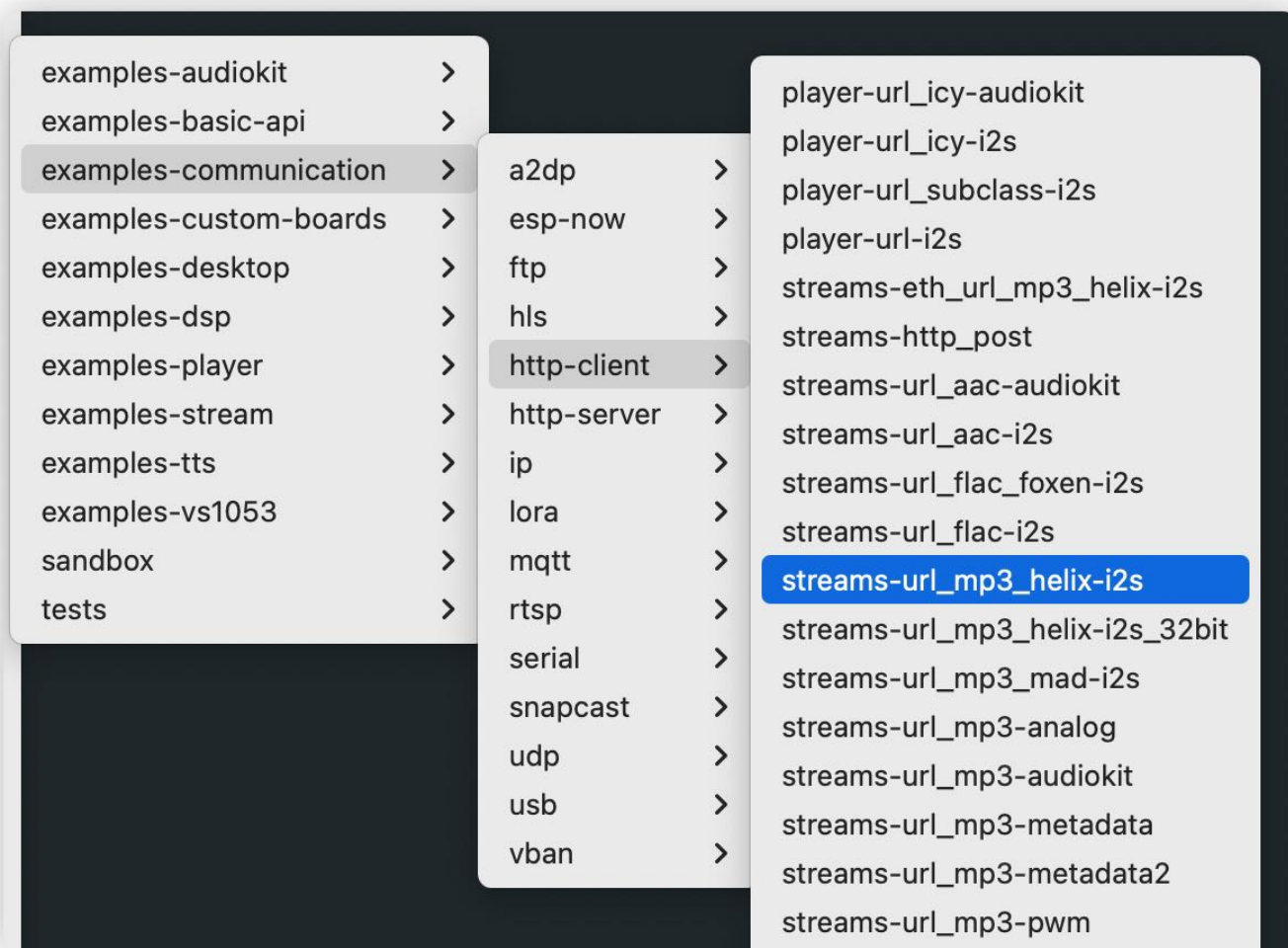
Clone using the web URL.

Open with GitHub Desktop

Download ZIP

四、网络播放代码

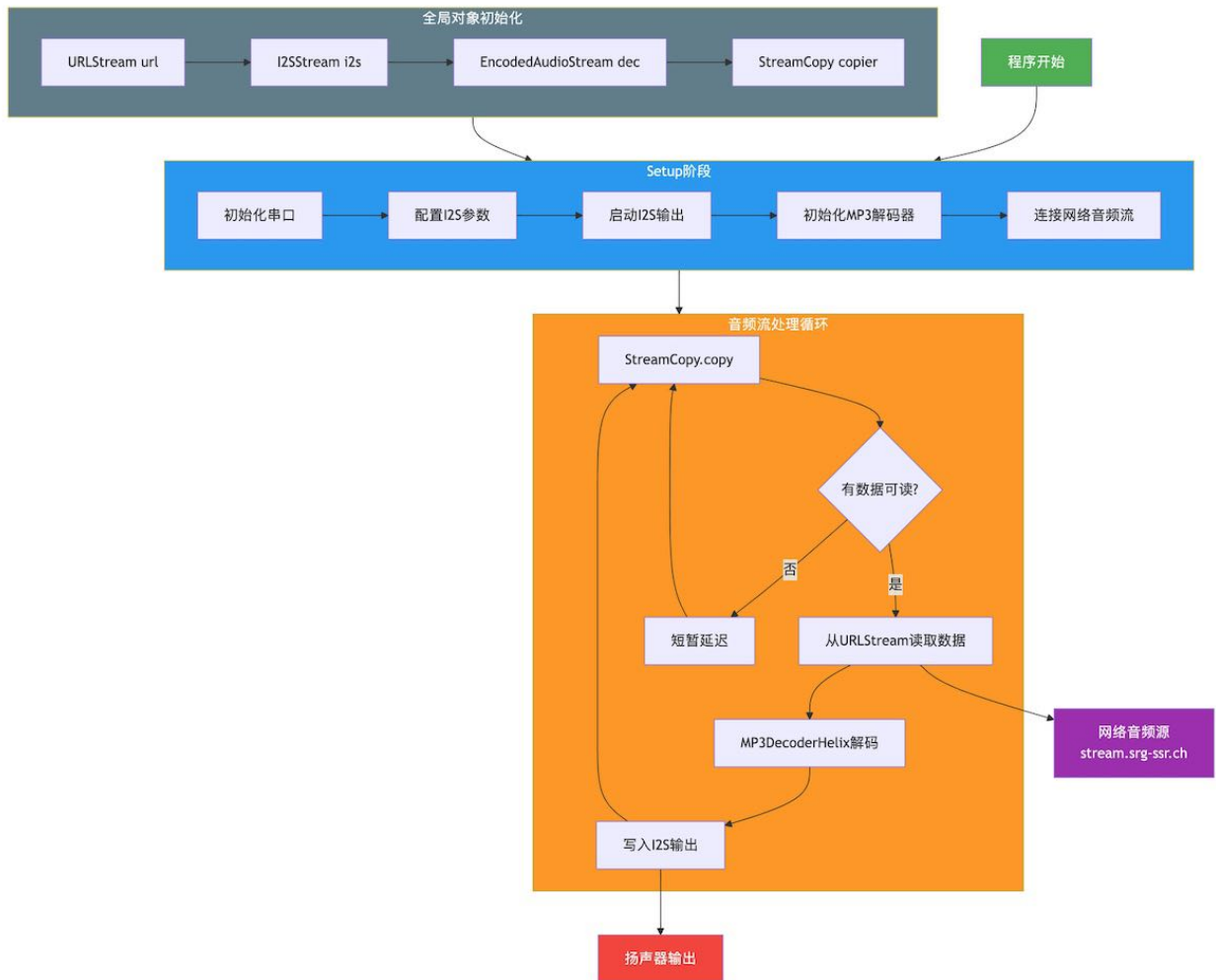
在arduino-audio-tools的示例中，我选择了这一个：



选择该例子后，具体的代码如下：

```
1  #include "AudioTools.h"
2  #include "AudioTools/AudioCodecs/CodecMP3Helix.h"
3  #include "AudioTools/Communication/AudioHttp.h"
4
5
6  URLStream url("ssid","password");
7  I2SStream i2s; // final output of decoded stream
8  EncodedAudioStream dec(&i2s, new MP3DecoderHelix()); // Decoding stream
9  StreamCopy copier(dec, url); // copy url to decoder
10
11
12 void setup(){
13     Serial.begin(115200);
14     AudioToolsLogger.begin(Serial, AudioToolsLogLevel::Info);
15
16     // setup i2s
17     auto config = i2s.defaultConfig(TX_MODE);
18     // you could define e.g your pins and change other settings
19     //config.pin_ws = 10;
20     //config.pin_bck = 11;
21     //config.pin_data = 12;
22     //config.mode = I2S_STD_FORMAT;
23     i2s.begin(config);
24
25     // setup I2S based on sampling rate provided by decoder
26     dec.begin();
27
28     // mp3 radio
29     url.begin("http://stream.srg-ssr.ch/m/rsj/mp3_128","audio/mp3");
30
31 }
32
33 void loop(){
34     copier.copy();
35 }
36
```

代码的主要逻辑如下：



其中，一般需要修改的部分如下：

- 连接WiFi的SSID和密码：

```

5
6   URLStream url("ssid","password");
7

```

- 要播放的音频网址：

```

28
29  // mp3 radio
30  url.begin("http://stream.srg-ssr.ch/m/rsj/mp3_128","audio/mp3");
31

```

为了适配 FireBeetle ESP32-P4 扩展板，我之前做过一些修改，包括：

1. I2S引脚设置

```

47
48 // setup i2s
49 auto config = i2s.defaultConfig(TX_MODE);
50 // you could define e.g your pins and change other settings
51 #if defined(CONFIG_IDF_TARGET_ESP32P4)
52     config.pin_ws = 28;
53     config.pin_bck = 29;
54     config.pin_data = 30;
55 #elif defined(CONFIG_IDF_TARGET_ESP32C5)
56     config.pin_ws = 23;
57     config.pin_bck = 24;
58     config.pin_data = 25;
59 #endif
60 //config.mode = I2S_STD_FORMAT;
61 i2s.begin(config);
62

```

这个地方，主要是配置I2S使用的引脚。因为我将音频功放模块，接到了SPI接口，所以按照前面的引脚对应关系，填写即可。注意 FireBeetle ESP32-P4 扩展板和 FireBeetle ESP32-C5 扩展板上的SPI引脚编号是不同的。

2. 音频网址：

```

65
66 // mp3 radio
67 // url.begin("http://stream.srg-ssr.ch/m/rsj/mp3_128","audio/mp3");
68 url.begin("http://192.168.1.15:8088/music/hjj-yty.mp3","audio/mp3"); // 河静静-摇太阳
69

```

音频网址我修改为本地的地址了，方便测试。我用手机开了热点，让我的电脑和FireBeetle ESP32-C5连上。

另外，需要注意的是，音乐文件请尽量避免使用中文名称，以防因URL编码问题导致请求失败。

五、烧录运行

编译上面的代码，Arduino IDE配置如下：

开发板 : "ESP32C5 Dev Module" >
端口 : "/dev/cu.usbmodem14201" >
Reload Board Data

获得开发板信息

USB CDC On Boot: "Enabled" >
CPU Frequency: "240MHz (WiFi)" >
Core Debug Level: "Info" >
Erase All Flash Before Sketch Upload: "Disabled" >
Flash Frequency: "80MHz" >
Flash Mode: "QIO" >
Flash Size: "4MB (32Mb)" >
JTAG Adapter: "Disabled" >
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)" >
PSRAM: "Disabled" >
Upload Speed: "921600" >
Zigbee Mode: "Disabled" >

编程器 >

烧录引导程序

注意修改上面红框的部分，确保串口可以输出调试信息。

代码调整完成后，编译烧录到开发板，串口输出信息如下：

输出 串口监视器 ×

消息 (按回车将消息发送到"/dev/cu.usbmodem14201"上的"ESP32C5 Dev Module")

```
...[ 1915][W][STA.cpp:137] _onStaArduinoEvent(): Reason: 2 - AUTH_EXPIRE
[ 1921][W][STA.cpp:543] disconnect(): STA already disconnected.
..
[I] URLStream.h : 365 - WiFiClient
[I] HttpRequest.h : 240 - process connecting to host 192.168.1.15 port 8088
[ 3453][E][NetworkClient.cpp:280] connect(): socket error on fd 48, errno: 104, "Connection reset by peer"
[I] HttpRequest.h : 367 - is connected false with timeout 60000
[E] HttpRequest.h : 243 - Connect failed
[I] HttpRequest.h : 316 - Request written ... waiting for reply
[I] URLStream.h : 95 - ==> http status: 0
```

因为前面代码中，我给了本地的音乐文件地址，所以这里的日志显示请求不到。

我需要在本地启动一个网络服务，来让开发板可以从网络访问音频文件，使用Python就可以简单方便的做到这一步。

将音乐文件放置到www目录的子目录music中：

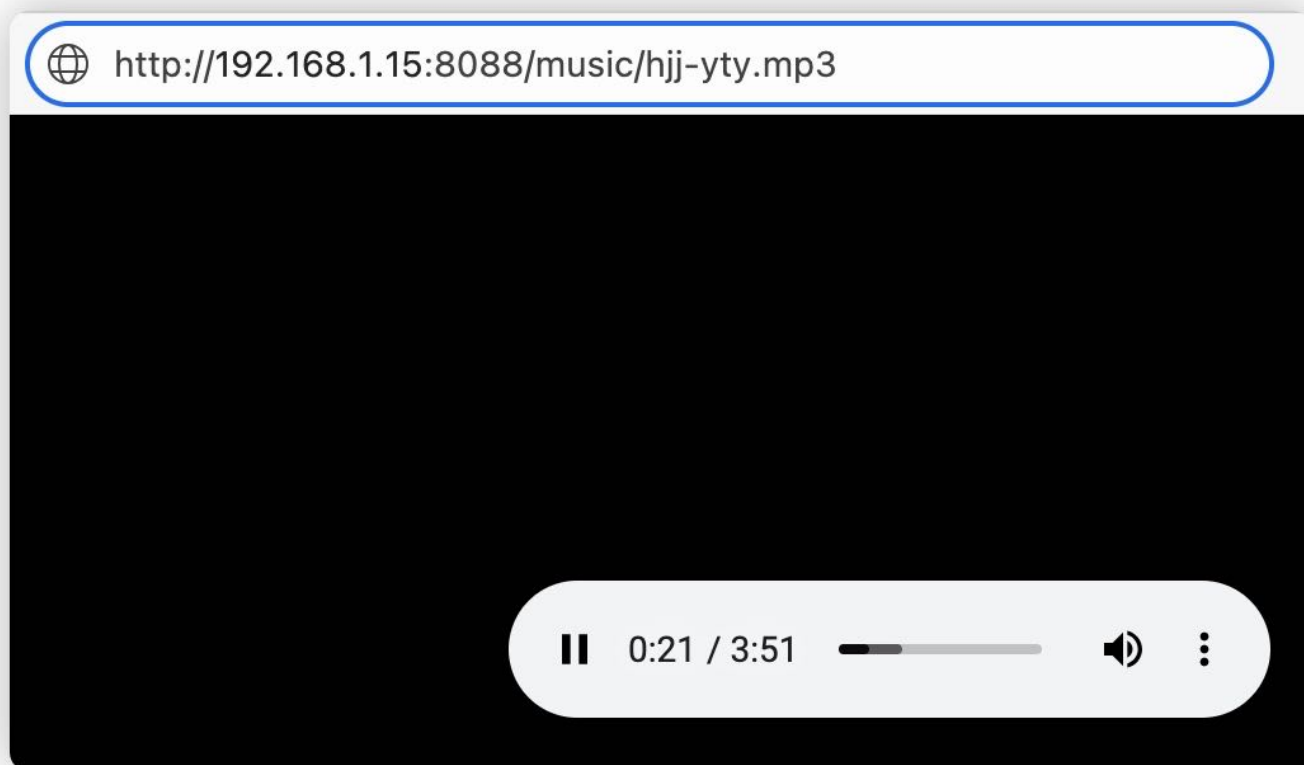


然后在该www目录下，使用Python启动一个HTTP服务器即可，命令如下：

```
cd www
```

```
python -m http.server 8088 -d .
```

启动网络服务完成后，可以打开网络浏览器测试一下，看看能否正常播放：



确认可以通过网络正常播放音频文件后，重启开发板，串口输出中会输出如下信息：

```
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
```

然后，开发板开始不断自动重启，无法正常工作：

```
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
Guru Meditation Error: Core 0 panic'ed (Store access fault). Exception was unhandled.

Core 0 register dump:
MEPC   : 0x4200cf20  RA      : 0x4200cf20  SP      : 0x408276f0  GP      : 0x40813684
TP     : 0x40827900  T0     : 0x00000002  T1     : 0x00000001  T2     : 0x00000001
S0/FP  : 0x00000003  S1     : 0x00000011  A0     : 0x0000010f  A1     : 0x4082f4b6
A2     : 0x0000010f  A3     : 0x00000017  A4     : 0xb9fc3e00  A5     : 0x408276f0
A6     : 0x00000009  A7     : 0xffffffff  S2     : 0x40829d48  S3     : 0x02063d88
S4     : 0x00000000  S5     : 0x00000000  S6     : 0x4081e000  S7     : 0x420e2000
S8     : 0x00000001  S9     : 0x420e1000  S10    : 0x00000000  S11    : 0x00000000
T3     : 0x00000007  T4     : 0x000003e8  T5     : 0x00000000  T6     : 0x0000000f
MSTATUS : 0x00001880  MTVEC   : 0x40800003  MCAUSE : 0x00000007  MTVAL  : 0x02063d88
MHARTID : 0x00000000

Stack memory:
408276f0: 0x4082f4b6 0xb9fc3e00 0x00000017 0x0000000d 0x00000000 0x420e1000 0x00000001 0x420e2000
40827710: 0x4081e000 0x4082779c 0x00000000 0x4082fcbc 0x408277a0 0x00000006 0x40829d48 0x4200f642
40827730: 0x00000000 0x00000000 0x00000000 0x0000059c 0x00000000 0x00000000 0x00000000 0x00000000
```

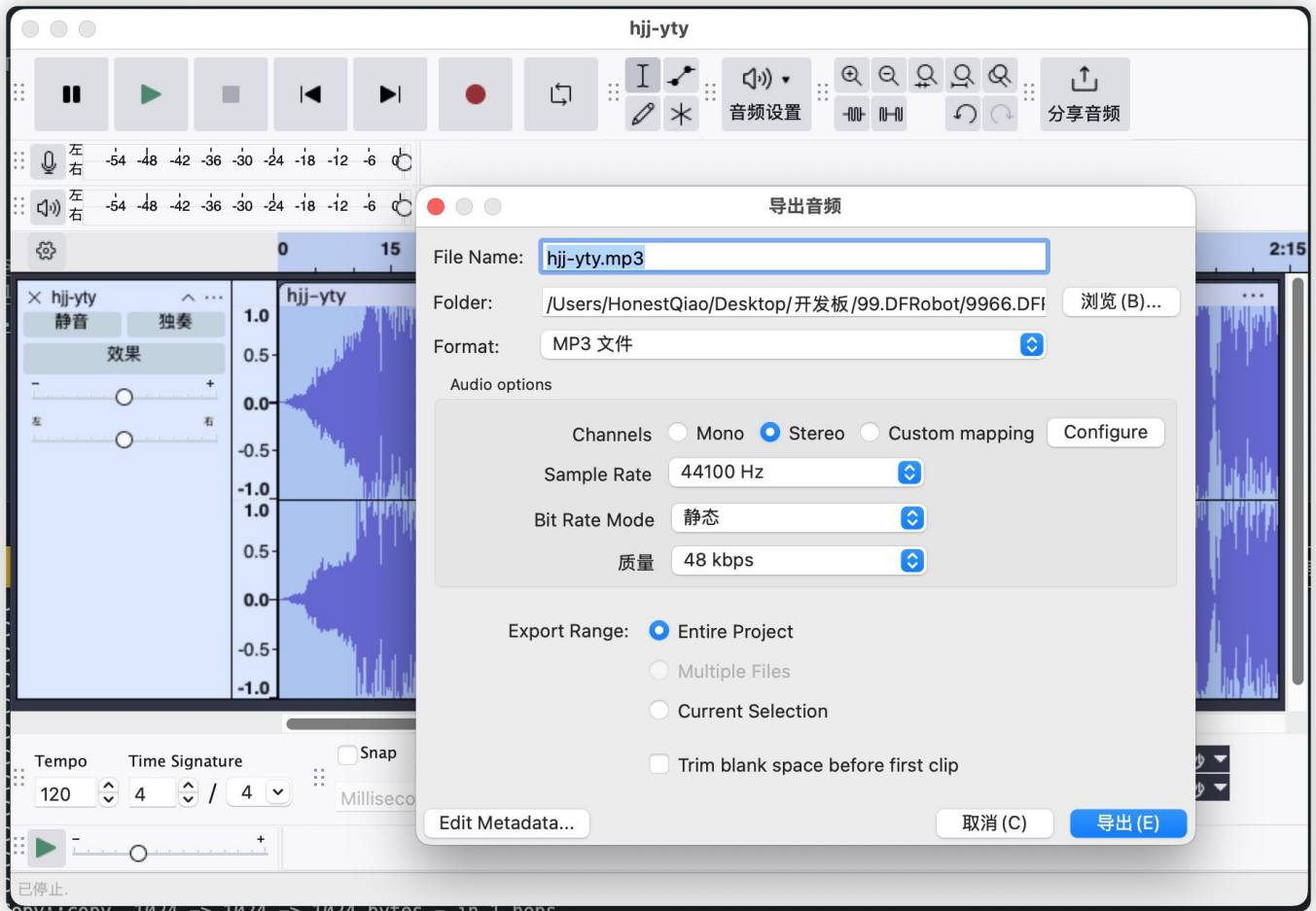
这是因为FireBeetle 2 ESP32-C5未配置PSRAM，其自身内存有限（仅384KB），在进行音频解码时极易发生内存溢出。

• 搭载ESP32-C5-WROOM-1-N4模组

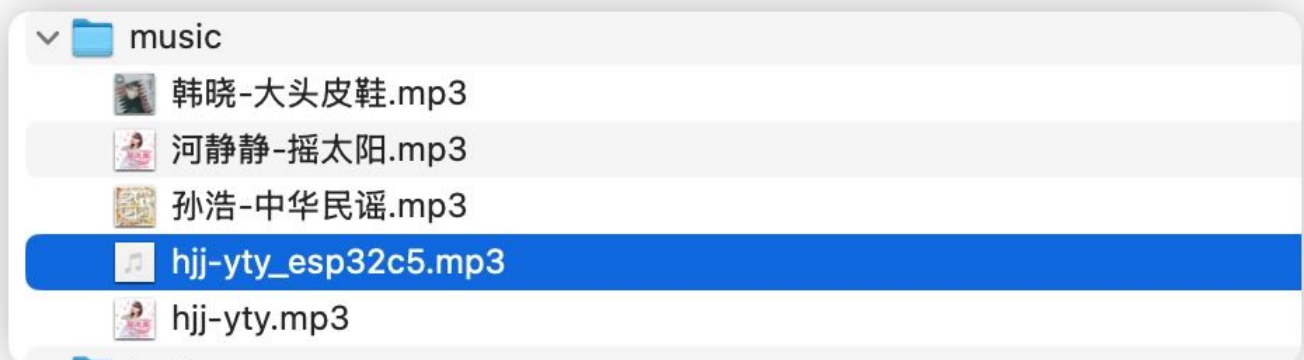
- 主频高达 240 MHz，内置384 KB SRAM、320 KB ROM、4 MB FLASH
- 支持 2.4 G 和5 G 双频 Wi-Fi 6，更低延迟，更低功耗
- 支持其他BLE、Zigbee、Thread通讯协议

我们需要将音频文件的码率降低，这可以使用Audacity来进行处理。

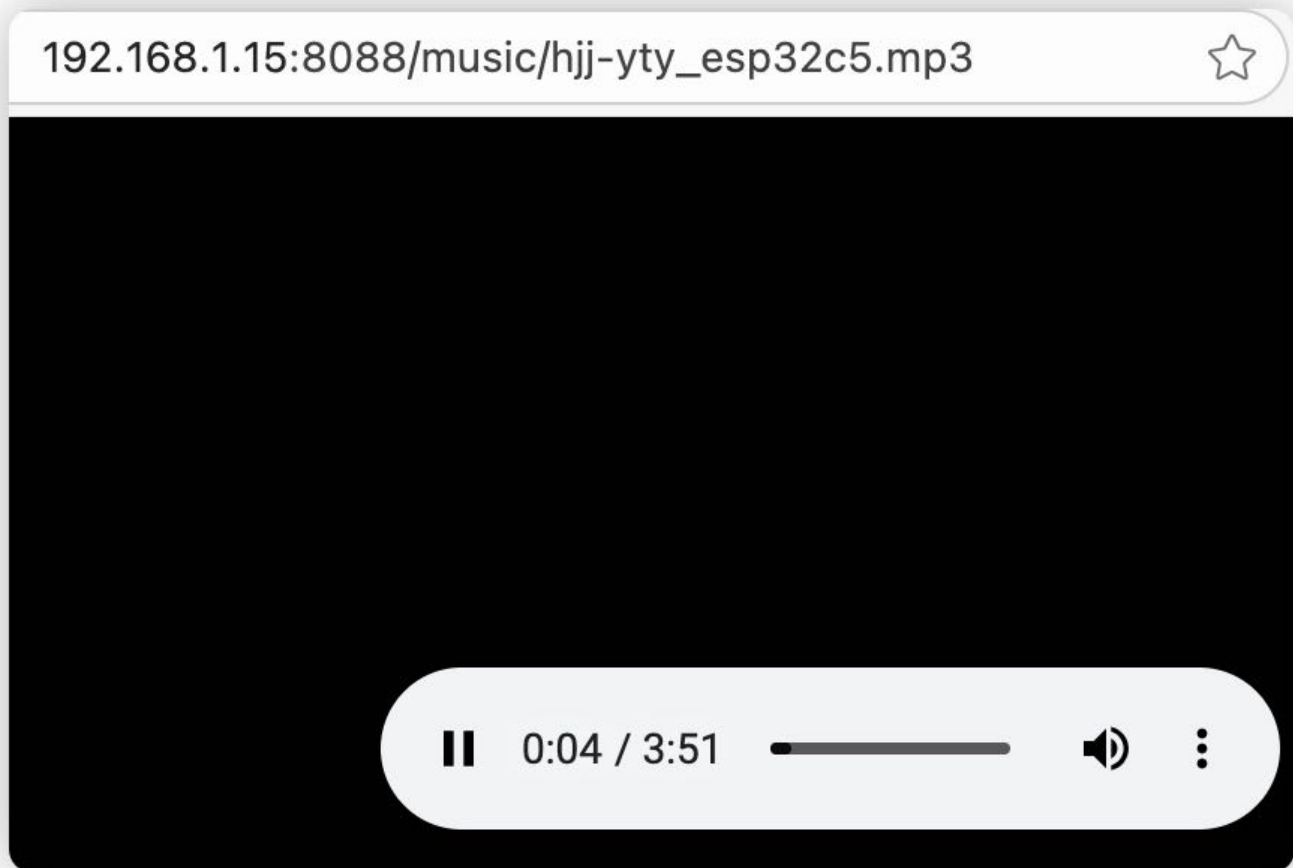
用Audacity打开音频文件，然后按照下图导出：



导出后，将其存放到音频文件目录中：



并再次通过网络播放测试：



然后修改代码中的音频网址：

```
66 // mp3 radio
67 // url.begin("http://stream.srg-ssr.ch/m/rsj/mp3_128","audio/mp3");
68 // url.begin("http://192.168.1.15:8088/music/hjj-yty.mp3","audio/mp3"); // 河静静-摇太阳
69 url.begin("http://192.168.1.15:8088/music/hjj-yty_esp32c5.mp3","audio/mp3"); // 河静静-摇太阳 ESP32-C5专用版
70
```

修改后，编译烧录到开发板，就可以看到日志输出信息：

```
输出  串口监视器  ×
消息 (按回车将消息发送到"/dev/cu.usbmodem14201"上的"ESP32C5 Dev Module")
[I] StreamCopy.n : 1/6 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
[I] StreamCopy.h : 176 - StreamCopy::copy 1024 -> 1024 -> 1024 bytes - in 1 hops
```

串口监视器持续滚动输出音频数据流日志，即表明播放正常。

六、音乐播放效果

实际的播放演示如下：

<https://www.bilibili.com/video/BV1C1spz7EBw/>