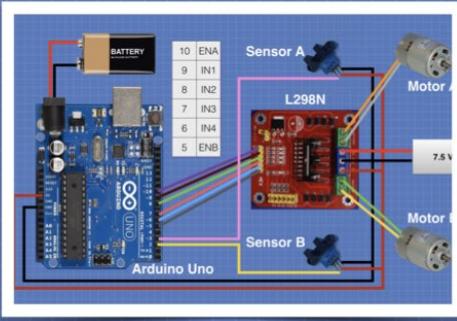
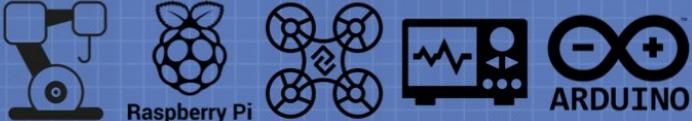


Por favor, ajude a apoiar o povo de Maui Doe para o "Maui Strong: Fire Relief Fund" da Hawai'i Community Foundation



Welcome to The Workshop!



Oficina DroneBot

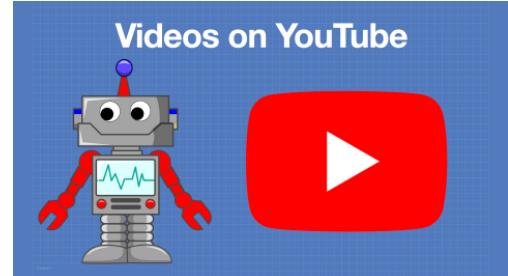
Arduino, Eletrônica, IoT, Raspberry Pi e Robôs – Bem-vindo ao Workshop!

[Lar](#) [Arduíno](#) [Raspberry Pi](#) [ESP32](#) [Eletrônicos](#) [Robôs](#) [Fórum](#) [YouTube](#) [Sobre](#)

Som com ESP32 – Protocolo I2S

[Índice \[mostrar \]](#)


O áudio encontra o ESP32 hoje, enquanto examinamos o protocolo I2S para áudio digital.


[Alterar tamanho do texto](#)
[90%](#) [100%](#) [110%](#) [120%](#)


Vamos manter contato!

Assine a newsletter e mantenha-se atualizado sobre o que está acontecendo no workshop.

Zero spam, sem vendas – apenas informações úteis!

[Inscreva-se hoje!](#)

Introdução

Se você estiver navegando pela folha de especificações de um dispositivo ESP32, poderá encontrar o termo "I2S". À primeira vista, você pode pensar que é outra forma de I2C, e "I2" de fato significa a mesma coisa, "Circuito Interintegrado". Mas é aí que a semelhança termina.

I2S é um protocolo para transferência de áudio digital. A qualidade do áudio pode variar de nível telefônico a ultra-alta fidelidade, e você pode ter um ou dois canais.



Imagen creditada a Tshirt Superstar – Música

Hoje exploraremos o uso do I2S com o ESP32 e construiremos alguns projetos que utilizam o protocolo I2S.

Vamos indo!

I2S e áudio digital

O **Inter-Integrated Circuit Sound Protocol, ou I2S**, foi desenvolvido pela Phillips Semiconductors em 1986. Como você deve se lembrar, a Phillips também desenvolveu o protocolo I2C, e ambos os protocolos foram construídos para atender a uma necessidade semelhante.

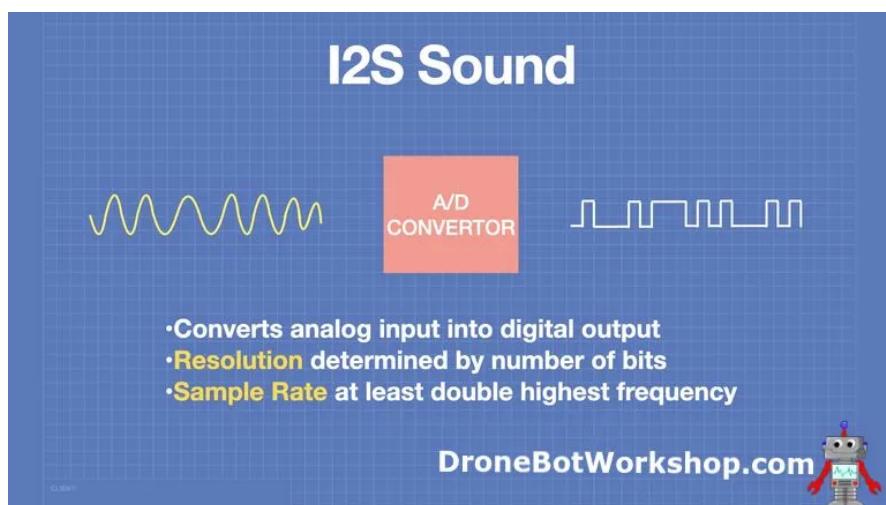
Tanto o I2C quanto o I2S atenderam à necessidade de compatibilidade entre circuitos integrados que lidavam com dados e informações sonoras. Protocolos padronizados para transferência de dados e som permitiriam projetos usando ICs de diferentes fabricantes, o que é bom para todos.

Para entender o I2S, é uma boa ideia entender também como funciona o áudio digital. Aqui está uma atualização rápida, caso você não esteja familiarizado com o conceito; se estiver, sinta-se à vontade para ignorá-lo!

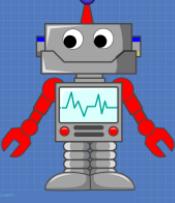
Áudio digital

O som, por sua própria natureza, é analógico e, antes do desenvolvimento do som digital, o equipamento de áudio também era analógico. As vibrações do som em um transdutor como um microfone devem ser amplificadas e depois enviadas para um alto-falante, cujo cone reproduz essas vibrações.

Levar o som do microfone para o alto-falante, especialmente se você quisesse gravá-lo e reproduzi-lo posteriormente no alto-falante, envolvia muitos componentes eletrônicos analógicos. Mesmo a melhor eletrônica analógica induzirá ruído elétrico e distorção no sinal, embora os designs modernos sejam impressionantemente limpos.



Join us on the Forum




FORUM

Artigos Mais Recentes

[Seeeduino XIAO ESP32S3 Sense Board – Uma pequena câmera ESP32](#)

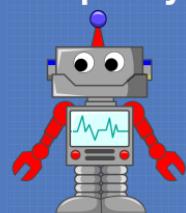
[LoRa – Primeiros passos com Arduino, ESP32 e Pico](#)

[ArduinoNano ESP32](#)

[Arduino Uno R4 – Mínimos e WiFi](#)

[Detecção de objetos ESP32-CAM com impulso de borda](#)

Frequently Asked Questions



What is your background?

How do you make those video animations?

Can I hire you?

Índice

[1 Introdução](#)

[2 I2S e áudio digital](#)

[2.1 Áudio Digital](#)

[2.2 Protocolo I2S](#)

[2.2.1 Controladores e Alvos](#)

[3 I2S e ESP32](#)

[4 periféricos I2S](#)

[5 Microfone I2S com ESP32](#)

[5.1 Módulo de Microfone INMP441](#)

[5.2 Conexão do módulo de microfone INMP441](#)

[5.3 Código do Módulo de Microfone INMP441](#)

[5.4 Testando o Microfone](#)

[6 Leitor de MP3 ESP32](#)

[6.1 Componentes do reprodutor de MP3](#)

[6.1.1 Módulo Amplificador MAX98357A I2S](#)

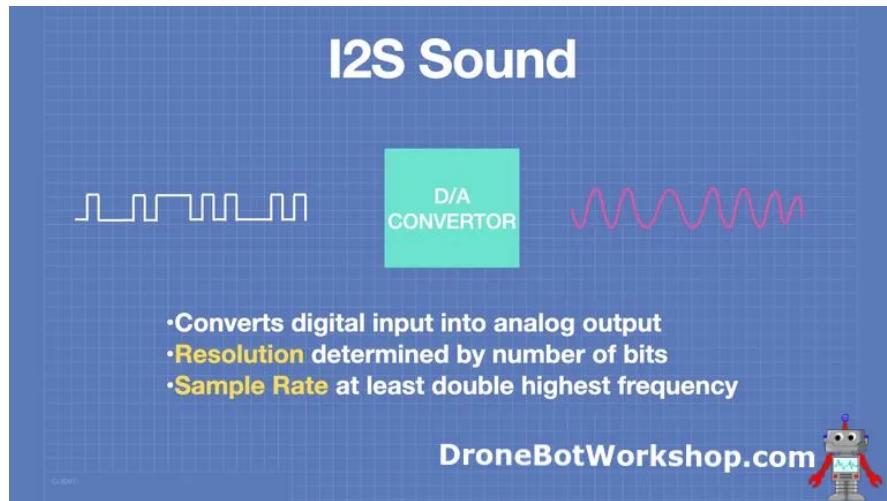
[6.1.2 Placa Breakout MicroSD Adafruit](#)

[6.2 Conexão do MP3 Player](#)

[6.3 Código do MP3 Player](#)

[6.4 Teste o MP3 Player](#)

O áudio digital foi apresentado como uma forma de eliminar essas distorções do som. Com áudio digital, o som é enviado para um conversor analógico para digital (ADC) para criar uma representação digital dele. Isto pode então ser armazenado ou transmitido sem qualquer degradação. Na outra extremidade, o sinal digital passa por um conversor digital para analógico (DAC) equivalente, que recria a entrada analógica e é então amplificado para acionar um alto-falante.



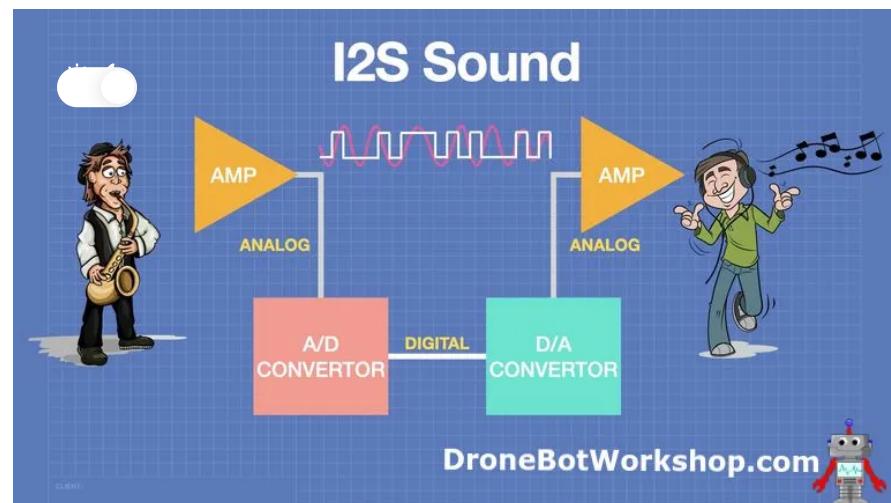
Um sinal de áudio digital não é, obviamente, uma representação perfeita do sinal original. A qualidade do sinal depende de dois parâmetros que se aplicam ao ADC e ao DAC:

- **Resolução** – O número de bits usados na amostra. Mais bits equivalem a melhor qualidade.
- **Taxa de amostragem** – quantas amostras por segundo estamos coletando. Isso precisa ser pelo menos duas vezes maior que a frequência mais alta que queremos amostrar.

Obviamente, quanto maiores as resoluções e a taxa de amostragem, maior será o arquivo de dados digitais resultante.

O áudio digital com qualidade de CD tem uma resolução de 16 bits e uma taxa de amostragem de 44,1 kHz, enquanto o áudio digital com qualidade de telefone tem 8 bits e é amostrado a 8 kHz.

Quando o áudio digital é transmitido, seja ao redor do mundo ou entre circuitos integrados, isso é feito em formato serial. Existem vários formatos, um dos mais comuns é *Pulse Code Modulation* ou PCM.



I2C trabalha com dados PCM digitais, usando qualquer resolução e taxa de amostragem. Ele pode ser usado para rotear o som da fonte ao destino, através de vários processadores de sinal e equalizadores em alguns casos.

Protocolo I2S

7 Rádio Internet ESP32

7.1 Conexão de rádio pela Internet

7.2 Código de Rádio da Internet

7.3 Testando o rádio da Internet

8 Executando em estéreo

8.1 Seleção de canal MAX98357A

8.2 Conexão de rádio estéreo

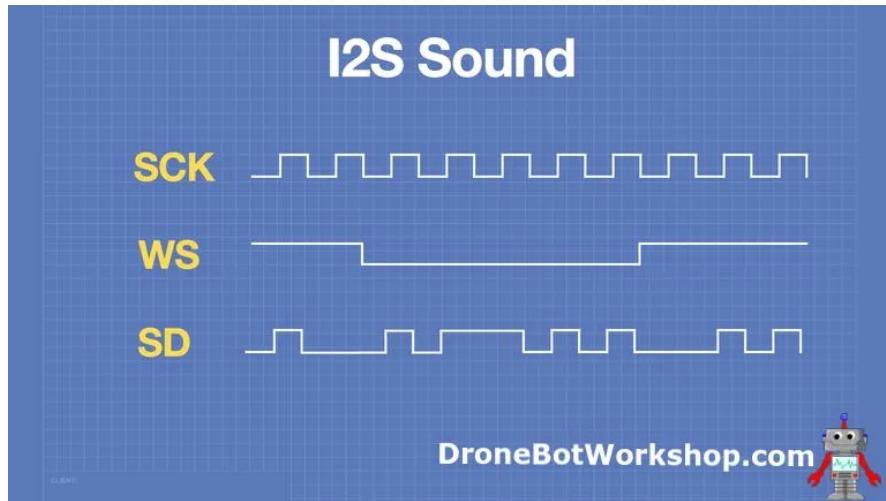
8.3 Código de Rádio Estéreo

8.4 Testando o Rádio Estéreo

9 Conclusão

9.1 Lista de Peças

9.2 Recursos

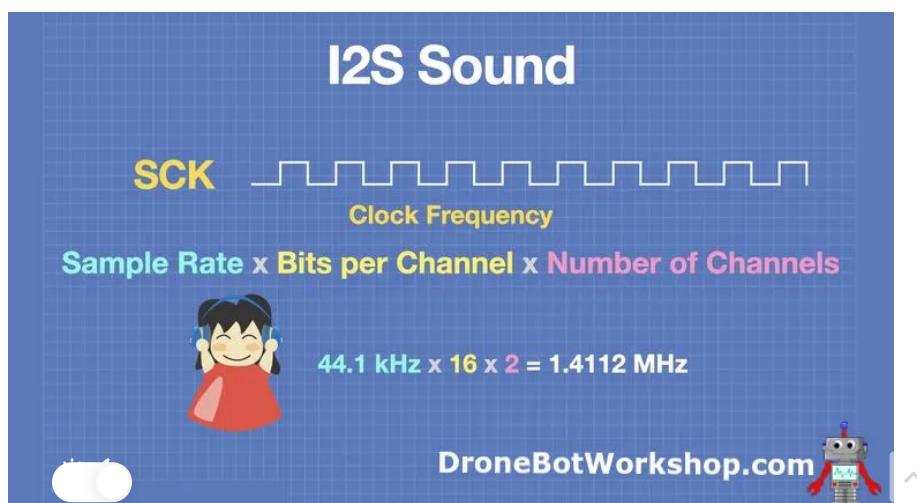


O protocolo I2S gerencia dados PCM em um barramento que consiste em pelo menos três linhas de conexão a seguir:

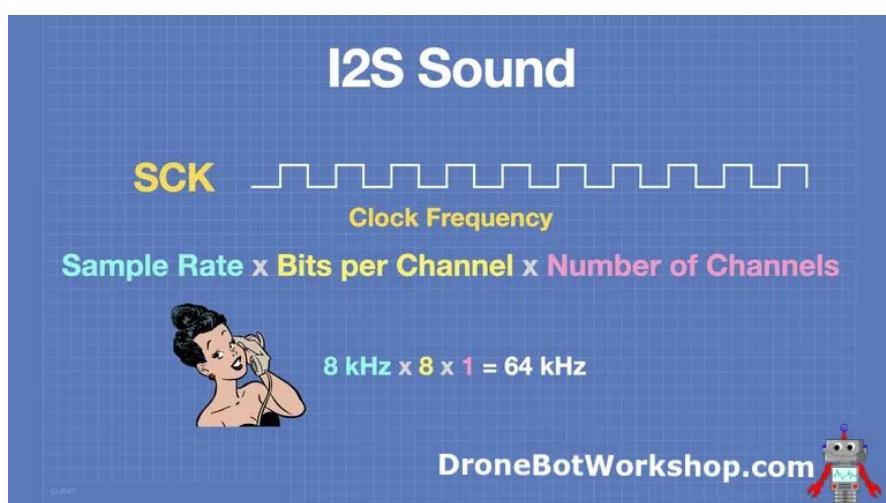
- **SCK** – A Linha Serial Clock, às vezes chamada de “linha bit clock”.
- **WS** – Word Select, que seleciona entre os canais de áudio Esquerdo e Direito.
- **SD** – Dados seriais, os dados de áudio PCM.

A qualidade do sinal de áudio determina a taxa do Serial Clock e é determinada com a seguinte fórmula:

Frequência do clock = taxa de amostragem x bits por canal x número de canais



Portanto, se quisermos enviar dois canais de áudio de alta qualidade, precisaremos de um clock de 1,4112 MHz.



O envio de um único canal de áudio com qualidade de telefone exigiria um clock de 64 kHz.

Controladores e alvos

Os dispositivos conectados a um barramento I2S podem ser divididos em duas categorias:

- **Controller** – Controls the SCK and WS signals.
- **Target** – Receives the SCK and WS signals.

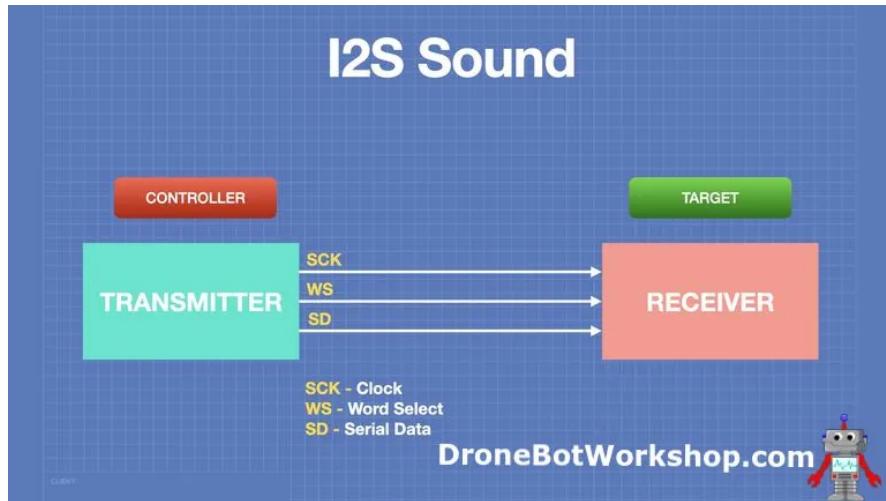
There can only be one controller on the bus, however, the bus can have multiple targets.

As for audio devices, they can be divided into three categories:

- **Transmitters** – Send audio signals.
- **Receivers** – Receive audio signals.
- **Controllers** – Control the audio signals

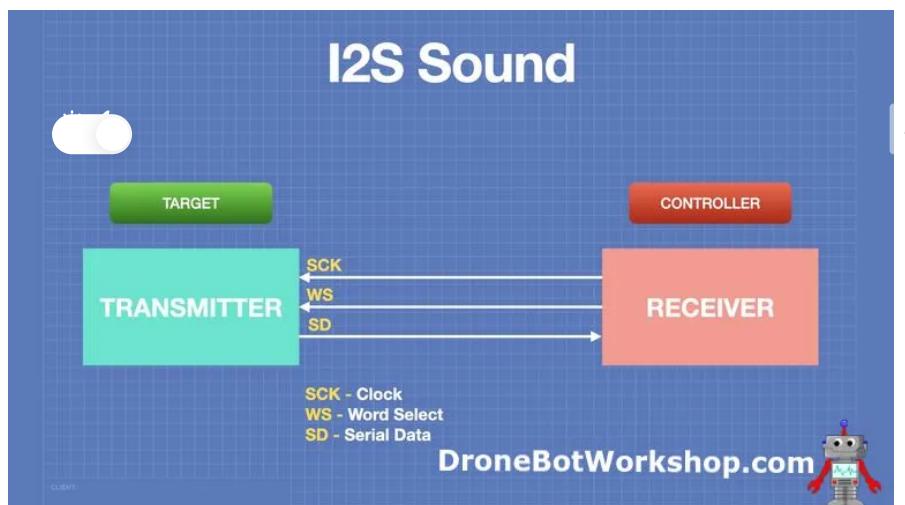
At a minimum, we need a Transmitter and Receiver, the Controller is optional.

A simple topology is illustrated here:

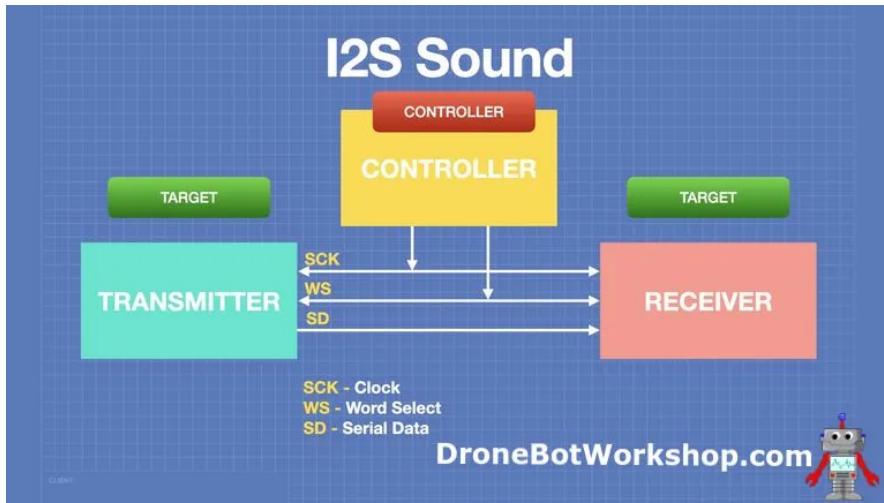


In this layout, the Audio Transmitter is also the I2S Controller, and it provides the SCK, WS, and SD signals. The Receiver is the I2S Target.

While that is a pretty standard arrangement it doesn't have to be the only one, the following layout is just as valid:



Aqui, os papéis do I2S são invertidos, com o receptor de áudio também atuando como controlador I2S. Ele fornece os sinais SCK e WS. Porém, o Transmissor, que é o Alvo neste arranjo, ainda fornece o SD (Dados Seriais). O Transmissor de áudio sempre fornece o SD, o que faz sentido se você pensar bem!



E aqui está um acordo com o Transmissor e o Receptor sendo ambos alvos I2S. Um dispositivo controlador separado é um controlador I2S e fornece SCK e WS para ambos os destinos. Mais uma vez, o Transmissor fornece o sinal SD.

I2S e ESP32

O ESP32 possui dois periféricos I2S, I2S0 e I2S1. Cada um pode ser configurado como Controlador ou Alvo, e cada um pode ser um Transmissor ou Receptor de áudio.

Cada controlador I2S pode operar no modo de comunicação half-duplex. Assim, os dois controladores podem ser combinados para estabelecer comunicação full-duplex.

Os dispositivos também possuem um modo DMA (Direct Memory Access), este modo permite o streaming de dados de amostra sem exigir que a CPU copie cada amostra de dados e pode ser útil ao transmitir áudio de alta qualidade.

Há também um modo que permite que a saída do I2S0 seja roteada internamente para a entrada do DAC ESP32 para produzir saída analógica direta sem envolver quaisquer codecs I2S externos.

Os periféricos I2S também suportam um modo avançado denominado “modo LCD” para comunicação de dados através de um barramento paralelo. Isso é usado por alguns LCDs e módulos de câmera. O modo LCD pode ser operado nos seguintes modos:

- Modo de transmissão mestre LCD
- Modo de recepção escravo da câmera
- Modo ADC/DAC

O Espressif tem sua excelente [documentação usual para I2S no ESP32](#), que pode fornecer links para mais informações.

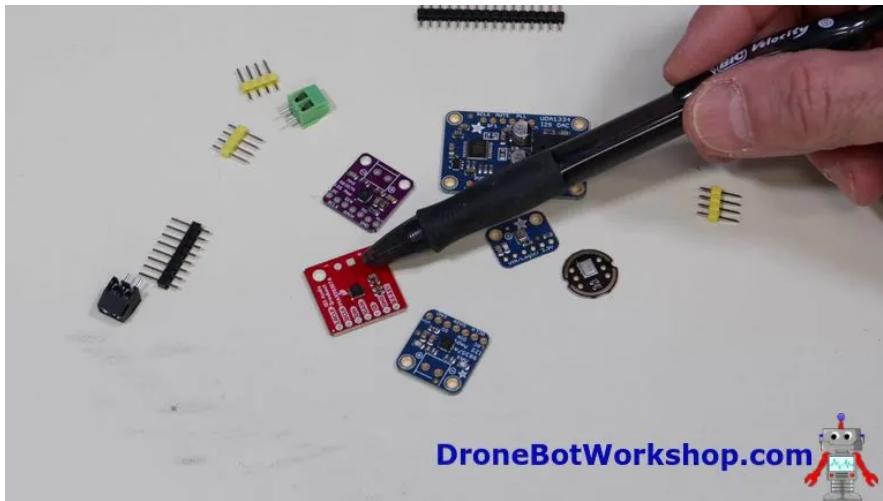
Periféricos I2S

I2S é um padrão e os periféricos variam desde pequenos módulos de amplificador e microfone até sistemas de áudio completos com conectividade I2S.

Em nossos experimentos, usaremos alguns periféricos I2S:

- Módulo de microfone INMP441
- Módulo Amplificador MAX98357A I2S

No vídeo também mostro alguns outros microfones e módulos amplificadores I2S



O microfone e o amplificador estão disponíveis em vários fornecedores, incluindo Amazon e eBay. Sparkfun e Adafruit também têm vários painéis I2S para você brincar.

Microfone I2S com ESP32

Começaremos nossos experimentos I2S com um módulo de microfone I2S.

Existem vários desses módulos disponíveis, usei um módulo INMP441 comum, mas você poderia substituir por outro módulo de microfone I2S.

Em nosso experimento, exibiremos as formas de onda de áudio do microfone usando o Serial Plotter no Arduino IDE.

Módulo de microfone INMP441

O INMP441 é um módulo de microfone I2S comum e barato. Ele usa um microfone MEMS (*Micro-ElectroMechanical Systems*) e possui um conversor A/D interno de 24 bits e interface I2S.

O Módulo de Microfone INMP441 possui as seguintes especificações:

- Resposta omnidirecional.
- Interface I2S de 24 bits.
- Relação sinal-ruído de 61 dBA.
- Resposta de frequência de 60 Hz – 15 kHz.

Possui as seguintes conexões:

- **S^D** – ^ conexão de dados seriais I2S.
- **V_{DD}** – tensão de entrada, de 3 a 6 volts.
- **GND** – Terra.
- **L/R** – Seleção de canal.
- **WS** – Seleção de palavras.
- **SCK** – Relógio serial.

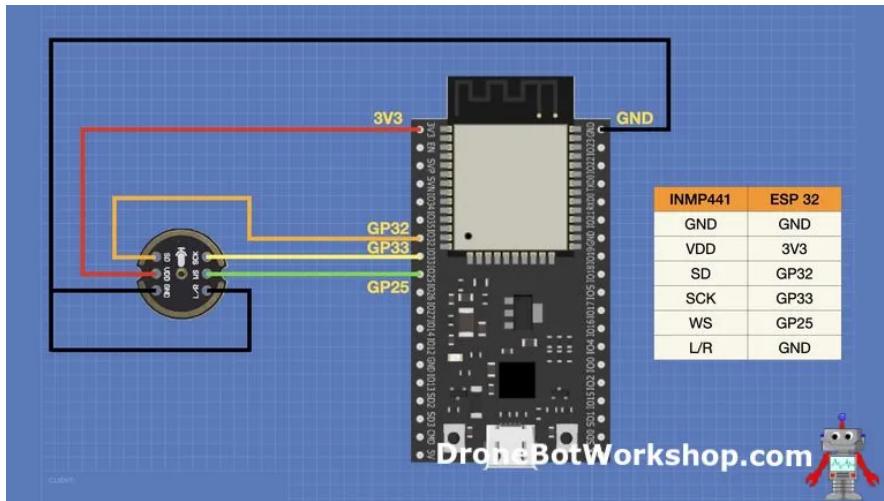
A seleção de canal (pino L/R) funciona da seguinte forma:

- **ESQUERDA** – L/R conectado ao GND.
- **DIREITA** – E/D conectado ao VDD.

Uma coisa sobre isso, e sobre a maioria dos módulos de microfone I2S MEMS, é que o som entra no microfone pela parte inferior da placa de circuito. Você verá um pequeno ícone de microfone ao lado de um pequeno orifício, é onde o som entra no microfone. Certifique-se de montar seu módulo de forma que o som possa entrar no microfone, isso geralmente envolve soldar os pinos “de cabeça para baixo”.

Conexão do módulo de microfone INMP441

Veja como conectaremos nosso módulo de microfone e ESP32. Observe que seu ESP32 pode ter uma pinagem diferente da ilustrada aqui, use os números GPIO em vez de pinos físicos para conectar seu módulo.



Você notará que o pino L/R do módulo do microfone está aterrado, pois o usaremos como canal ESQUERDO.

Código do módulo de microfone INMP441

Em nosso primeiro experimento, usaremos a Biblioteca I2S que está instalada em seu Arduino IDE quando você instalar os arquivos do ESP32 Boards Manager. Aqui está a aparência do código:

```
ESP32 I2S Microphone Sample
1 /*
2   ESP32 I2S Microphone Sample
3   esp32-i2s-mic-sample.ino
4   Sample sound from I2S microphone, display on Serial Plotter
5   Requires INMP441 I2S microphone
6
7   DroneBot Workshop 2022
8   https://dronebotworkshop.com
9 */
10
11 // Include I2S driver
12 #include <driver/i2s.h>
13
14 // Connections to INMP441 I2S microphone
15 #define I2S_WS 25
16 #define I2S_SD 33
17 #define I2S_SCK 32
18
19 // Use I2S Processor 0
20 #define I2S_PORT I2S_NUM_0
21
22 // Define input buffer length
23 #define bufferLen 64
24 int16_t sBuffer[bufferLen];
25
26 void i2s_install() {
27     // Set up I2S Processor configuration
28     i2s_config_t i2s_config = {
29         .mode = I2S_MODE_MASTER | I2S_MODE_RX,
30         .sample_rate = 44100,
31         .bits_per_sample = i2s_bits_per_sample_t(16),
32         .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
33         .communication_format = i2s_comm_format_t(I2S_COMM_FORMAT_STAND_I2S),
34         .intr_alloc_flags = 0,
35         .dma_buf_count = 8,
36         .dma_buf_len = bufferLen,
37         .use_apll = false
38     };
39
40     i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
41 }
42
43 void i2s_setpin() {
44     // Set I2S pin configuration
45     const i2s_pin_config_t pin_config = {
46         .bck_io_num = I2S_SCK,
47         .ws_io_num = I2S_WS,
48         .data_out_num = -1,
49         .data_in_num = I2S_SD
50     };
51
52     i2s_set_pin(I2S_PORT, &pin_config);
53 }
54
55 void setup() {
56
57     // Set up Serial Monitor
58     Serial.begin(115200);
59     Serial.println(" ");
60
61     delay(1000);
62
63     // Set up I2S
64     i2s_install();
}
```

```

55     i2s_setpin();
56     i2s_start(I2S_PORT);
57
58
59     delay(500);
60 }
61
62 void loop() {
63
64     // False print statements to "lock range" on serial plotter display
65     // Change rangelimit value to adjust "sensitivity"
66     int rangelimit = 3000;
67     Serial.print(rangelimit * -1);
68     Serial.print(" ");
69     Serial.print(rangelimit);
70     Serial.print(" ");
71
72     // Get I2S data and place in data buffer
73     size_t bytesIn = 0;
74     esp_err_t result = i2s_read(I2S_PORT, &sBuffer, bufferLen, &bytesIn, portMAX_DELAY);
75
76     if (result == ESP_OK) {
77
78         // Read I2S data buffer
79         int16_t samples_read = bytesIn / 8;
80         if (samples_read > 0) {
81             float mean = 0;
82             for (int16_t i = 0; i < samples_read; ++i) {
83                 mean += (sBuffer[i]);
84             }
85
86             // Average the data reading
87             mean /= samples_read;
88
89             // Print to serial plotter
90             Serial.println(mean);
91         }
92     }
93 }
94
95
96
97
98
99
100
101
102
103 }
```

Começamos incluindo o driver ESP32 I2S.

Em seguida, definimos as conexões com nosso microfone. Se desejar, você pode religar o microfone e alterar o código aqui.

O ESP32 possui dois processadores I2S internos. Usaremos a primeira, Porta I2S 0. Também definimos o comprimento de um buffer de dados de entrada.

Next, we have a function called *i2s_install*, which sets up the I2S port parameters.

A second function, *i2s_setpin*, sets up the physical connection to the I2S device, which in our case is the microphone module.

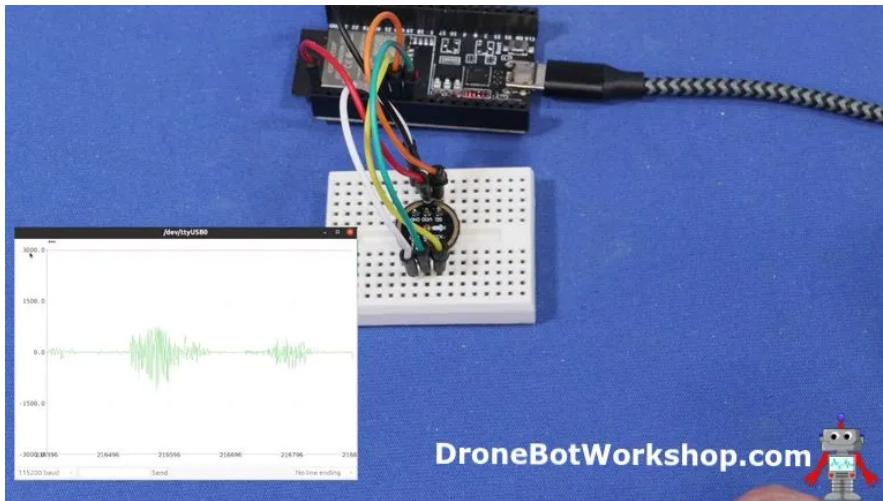
In the Setup, we set up our serial connection, as we will be using the Serial Plotter to display our audio waveforms. We then call our two functions to set up the I2S port, and then start it with a third built-in function.

Our Loop starts with a “false” print statement, this just causes two constants to be printed to steady the reading on the Serial Plotter, which otherwise will dynamically change its Y-axis scale.

We then read data from the module and place it in our data buffer. If the data is good, we read it out and display it on the Serial Plotter

Testing the Microphone

Hook everything up, load the sketch and open the Serial Plotter.



You should see a representation of the sound that the microphone is getting. You can adjust the sensitivity by altering the *rangelimit* variable in the Loop.

ESP32 MP3 Player

For our next experiment, we will be using an I2S amplifier module. The sound source will be an MP3 file that is stored on a MicroSD card.

This is an extremely basic MP3 player, for practical use you would need to make a system for navigating the MicroSD to play more than one selection. It's just to illustrate how to use the I2S amplifier, as well as a library that makes working with I2S audio applications a bit easier.

MP3 Player Components

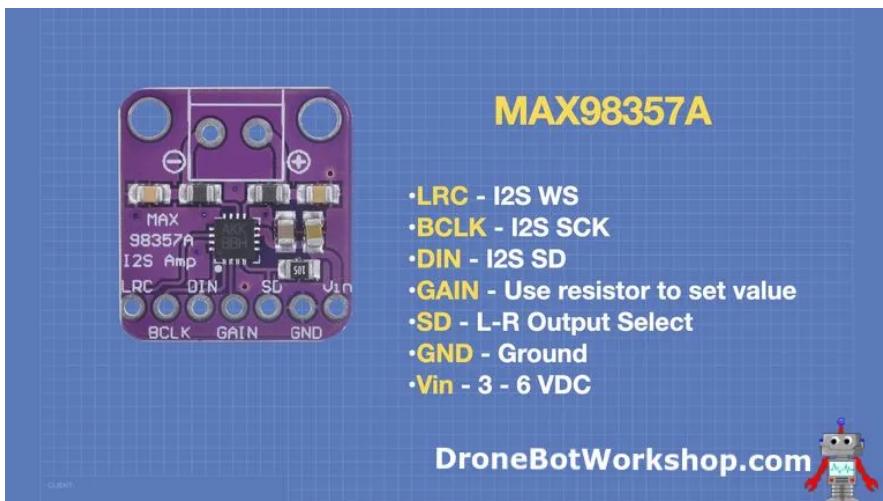
We will be adding two components to our ESP32 to construct our MP3 player, specifically an I2S amplifier and a MicroSD card breakout box. Of course, we will also need a MicroSD card, a small one formatted with FAT32. Put an MP3 file onto the card, in our experiment we will only be playing one selection.

MAX98357A I2S Amplifier Module

The MAX98357A I2S amplifier module is an inexpensive yet surprisingly powerful audio amplifier module with an I2S input.

The device can output up to 3 watts into a 4-ohm load. Note that you must use a speaker with a 4 to 8-ohm voice coils, and you can't use a dynamic speaker, as the voice coil is actually part of the output driving circuit in the module.

The device has the following pinout:



Note that the data is input on the DIN line, not the SD line, which is used for output channel selection. We will go into detail about output channel selection in a bit.

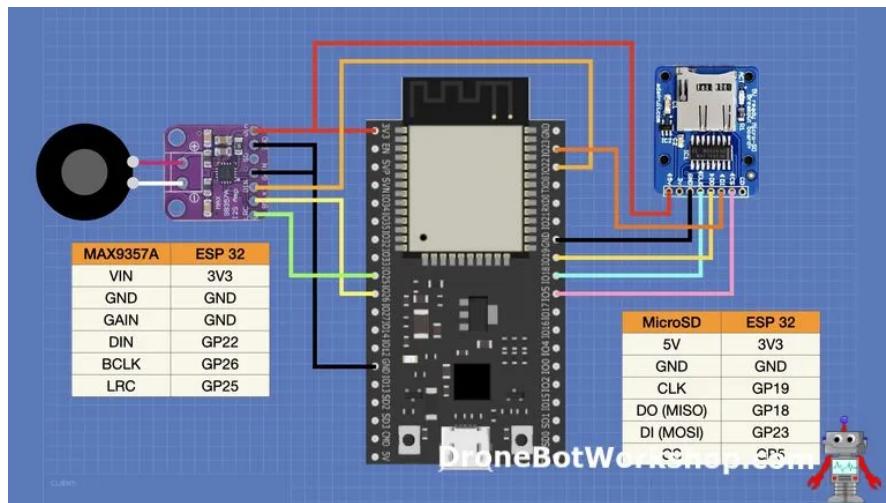
Adafruit MicroSD Breakout Board

We need a MicroSD card breakout board and I selected one from Adafruit. The reason for getting this module was its ability to work with both 3.3-volt and 5-volt logic. If you use another board, make sure it can operate on 3.3-volts, as many popular modules are for 5-volt logic and would damage the ESP32.

The MicroSD card module uses the SPI bus for communications.

MP3 Player Hookup

Here is the hookup for our MP3 player:



Se desejar, você pode mover o amplificador I2S para diferentes pinos GPIO, pois não há nada de especial neles. Apenas certifique-se de fazer as alterações apropriadas no esboço. O cartão MicroSD requer conexões de barramento SPI, existem vários no ESP32 para você escolher.

Código do reprodutor de MP3

Estaremos usando uma biblioteca que facilita muito o trabalho com I2S. Não está no seu Gerenciador de Biblioteca, então você precisará baixá-lo.

[A biblioteca ESP32-AudioI2S pode ser encontrada no GitHub](#), você pode cloná-la na pasta *Bibliotecas* do Arduino ou apenas baixá-la como um arquivo ZIP. Se você pegar o arquivo ZIP, poderá adicioná-lo ao seu Arduino IDE usando o item *Adicionar biblioteca ZIP* no menu *Sketch*.

Esta biblioteca irá simplificar o trabalho com I2S, você cria um objeto “áudio” que pode então manir o código.

Aqui está o esboço que usaremos:

```
ESP32 SD I2S Music Player
1 /*
2  * ESP32 SD I2S Music Player
3  * esp32-i2s-sd-player.ino
4  * Plays MP3 file from microSD card
5  * Uses MAX98357 I2S Amplifier Module
6  * Uses ESP32-audioI2S Library - https://github.com/schreibfaul1/ESP32-audioI2S
7  *
8  * DroneBot Workshop 2022
9  * https://dronebotworkshop.com
10 */
11
12 // Include required libraries
13 #include "Arduino.h"
14 #include "Audio.h"
15 #include "SD.h"
16 #include "FS.h"
17
18 // microSD Card Reader connections
19 #define SD_CS      5
20 #define SPI_MOSI   23
21 #define SPI_MISO   19
22 #define SPI_SCK    18
23
24 // I2S Connections
25 #define I2S_DOUT   22
26 #define I2S_BCLK   26
27 #define I2S_LRC    25
28
29 // Create Audio object
30 Audio audio;
31
32 void setup() {
```

```

33 // Set microSD Card CS as OUTPUT and set HIGH
34 pinMode(SD_CS, OUTPUT);
35 digitalWrite(SD_CS, HIGH);
36
37
38 // Initialize SPI bus for microSD Card
39 SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
40
41 // Start Serial Port
42 Serial.begin(115200);
43
44 // Start microSD Card
45 if(!SD.begin(SD_CS))
46 {
47   Serial.println("Error accessing microSD card!");
48   while(true);
49 }
50
51 // Setup I2S
52 audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
53
54 // Set Volume
55 audio.setVolume(5);
56
57 // Open music file
58 audio.connecttoFS(SD, "/MYMUSIC.mp3");
59
60 }
61
62 void loop()
63 {
64   audio.loop();
65 }
```

Começamos carregando a nova biblioteca que acabamos de instalar, junto com as bibliotecas SD e SPI que precisaremos para trabalhar com o breakout do cartão MicroSD.

Em seguida, definimos a conexão ao módulo MicroSD e ao módulo amplificador I2S.

Depois disso, usamos nossa nova biblioteca para criar um objeto “áudio”.

No Setup, configuramos o terminal CS (Chip Select) no MicroSD para que esteja sempre selecionado. Iniciamos a porta serial e também iniciamos o cartão MicroSD.

Supondo que o MicroSD esteja OK, configuramos nossa porta I2C. Observe como a biblioteca torna isso muito mais fácil em comparação ao uso apenas da biblioteca ESP32 I2S (da qual esta biblioteca depende).

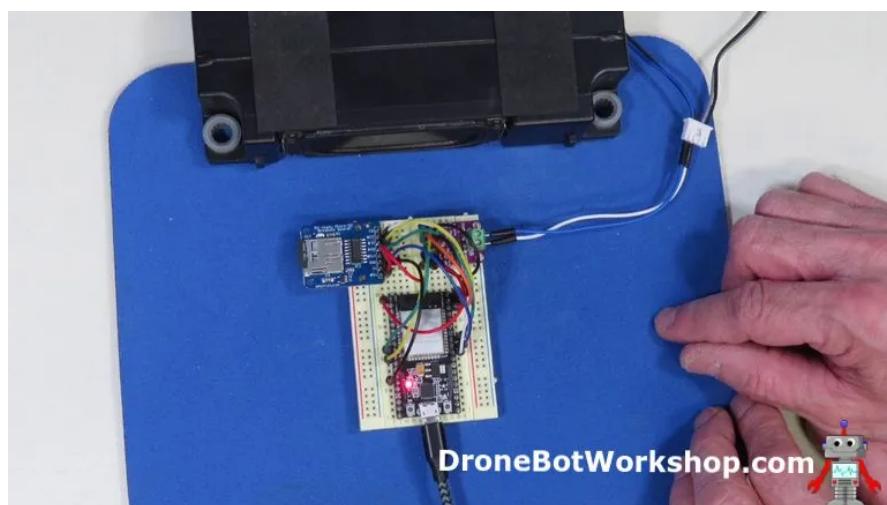
Também definimos o nível de áudio, qualquer número de 0 (sem áudio) a 21 funcionará aqui. É melhor começar com um número pequeno (escolhi 5), pois o amplificador é bastante eficiente.

Finally, we open up the MP3 file, using a “connect to FS (connect to File System) property of the audio object. Again, the library makes grabbing an audio source very easy.

The Loop couldn't be simpler. We just call the Loop method of the audio object repeatedly.

Test the MP3 Player

Place a MicroSD card that has the file you wish to play into the breakout board. Load the code onto the ESP32.



When it has done uploading, press the Reset button on the ESP32. You should be greeted by the contents of your MP3 file.

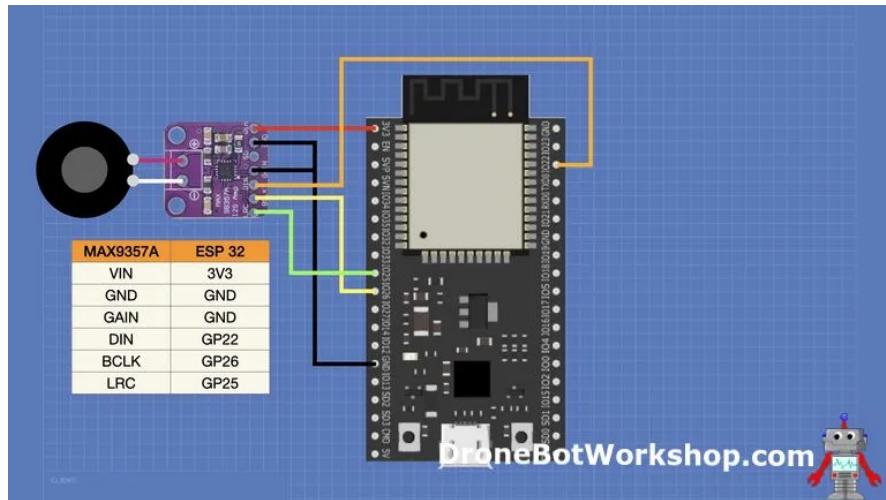
Make sure you chose something you like to listen to, as you have no volume control and can only play one selection once. But, as far as demo code goes, it shows you how simple it is to build the foundation of an I2S audio player.

ESP32 Internet Radio

In this experiment, we will take advantage of the WiFi capabilities of the ESP32 and build an internet radio, which will use the same I2S amplifier module that we previously used.

Internet Radio Hookup

The hookup of our Internet Radio is essentially the same as the hookup of the MP3 player, minus the MicroSD module. If you have already wired up the MP3 player, then you can just leave it as it is, as the MicroSD module will just be ignored.



Internet Radio Code

We will be using the same library we used for the MP3 player, as it makes it very easy to build an Internet Radio since you can just specify a URL as a sound source.

```
Simple Internet Radio Demo
1 /*
2   Simple Internet Radio Demo
3   esp32-i2s-simple-radio.ino
4   Simple ESP32 I2S radio
5   Uses MAX98357 I2S Amplifier Module
6   Uses ESP32-audioI2S Library - https://github.com/schreibfaul1/ESP32-audioI2S
7
8   DroneBot Workshop 2022
9     ://dronebotworkshop.com
10
11
12 // Include required libraries
13 #include "Arduino.h"
14 #include "WiFi.h"
15 #include "Audio.h"
16
17 // Define I2S connections
18 #define I2S_DOUT 22
19 #define I2S_BCLK 26
20 #define I2S_LRC 25
21
22 // Create audio object
23 Audio audio;
24
25 // Wifi Credentials
26 String ssid = "YOURSSID";
27 String password = "YOURPASSWORD";
28
29 void setup() {
30
31   // Start Serial Monitor
32   Serial.begin(115200);
33
34   // Setup WiFi in Station mode
35   WiFi.disconnect();
36   WiFi.mode(WIFI_STA);
37   WiFi.begin(ssid.c_str(), password.c_str());
38
39   while (WiFi.status() != WL_CONNECTED) {
40     delay(500);
41     Serial.print(".");
42   }
43
44   // WiFi Connected, print IP to serial monitor
45   Serial.println("");
46   Serial.println("WiFi connected");
```

```

47   Serial.println("IP address: ");
48   Serial.println(WiFi.localIP());
49   Serial.println("");
50
51 // Connect MAX98357 I2S Amplifier Module
52 audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
53
54 // Set the volume (0-100)
55 audio.setVolume(10);
56
57 // Connect to an Internet radio station (select one as desired)
58 //audio.connecttohost("http://vis.media-ice.musicradio.com/CapitalMP3");
59 //audio.connecttohost("mediaserv30.live-nect MAX98357 I2S Amplifier Module
60 //audio.connecttohost("www.surfmusic.de/m3u/100-5-das-hitradio,4529.m3u");
61 //audio.connecttohost("stream.1a-webradio.de/deutsch/mp3-128/vtuner-1a");
62 //audio.connecttohost("www.antenne.de/webradio/antenne.m3u");
63 audio.connecttohost("0n-80s.radionetz.de:8000/0n-70s.mp3");
64
65 }
66
67 void loop()
68 {
69   // Run audio player
70   audio.loop();
71
72 }
73
74 // Audio status functions
75
76 void audio_info(const char *info) {
77   Serial.print("info      "); Serial.println(info);
78 }
79
80 void audio_id3data(const char *info) { //id3 metadata
81   Serial.print("id3data    "); Serial.println(info);
82 }
83
84 void audio_eof_mp3(const char *info) { //end of file
85   Serial.print("eof_mp3    "); Serial.println(info);
86 }
87
88 void audio_showstation(const char *info) {
89   Serial.print("station    "); Serial.println(info);
90 }
91
92 void audio_showstreaminfo(const char *info) {
93   Serial.print("streaminfo "); Serial.println(info);
94 }
95
96 void audio_showstreamtitle(const char *info) {
97   Serial.print("streamtitle "); Serial.println(info);
98 }
99
100 void audio_bitrate(const char *info) {
101   Serial.print("bitrate    "); Serial.println(info);
102 }
103
104 void audio_commercial(const char *info) { //duration in sec
105   Serial.print("commercial "); Serial.println(info);
106 }
107
108 void audio_icyurl(const char *info) { //homepage
109   Serial.print("icyurl    "); Serial.println(info);
110 }
111
112 void audio_lasthost(const char *info) { //stream URL played
113   Serial.print("lasthost   "); Serial.println(info);
114 }
115
116 void audio_eof_speech(const char *info) {
117   Serial.print("eof_speech "); Serial.println(info);
118 }
119 }
```

Our code is quite similar to the MP3 player, except we don't have the MicroSD breakout board to deal with.

After defining the I2S amplifier connections, we create an audio object, just as we did before.

Then we grab the WiFi credentials, as our Internet Radio will (obviously) need to connect to the Internet!

Setup starts the Serial Monitor, which we will use to display program information, and then connects to the WiFi.

We then define the connections to the amplifier module and set the volume, again as we did in the last sketch.

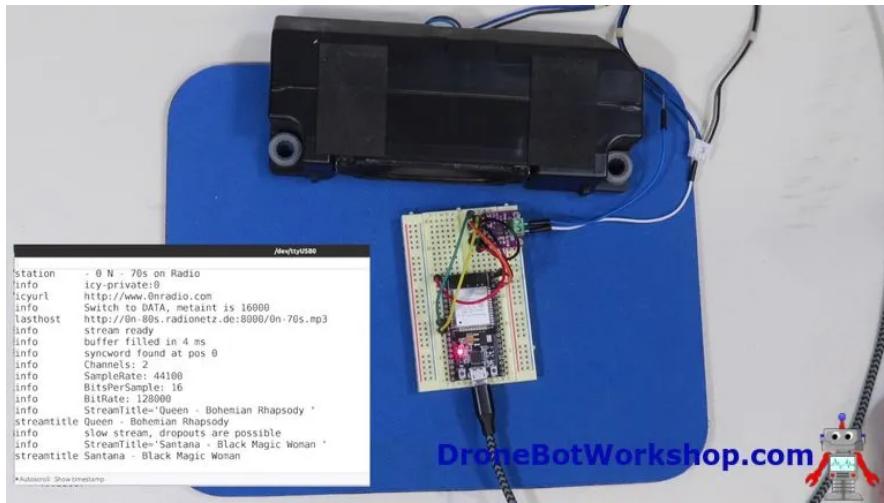
Finally, we use the libraries `connecttohost` function to connect to a URL, I have listed a few here and there are many more you can use.

Once again, the Loop just uses the audio object `Loop` method to play music.

We also have a number of other functions below the Loop. These functions will display status information on the serial monitor.

Testing the Internet Radio

Load the code onto the ESP32 and press reset. Observe the serial monitor.



You should first see the WiFi connection information, which hopefully will show that you have connected to a network. Assuming you have, you will then start to load the URL for the radio station.

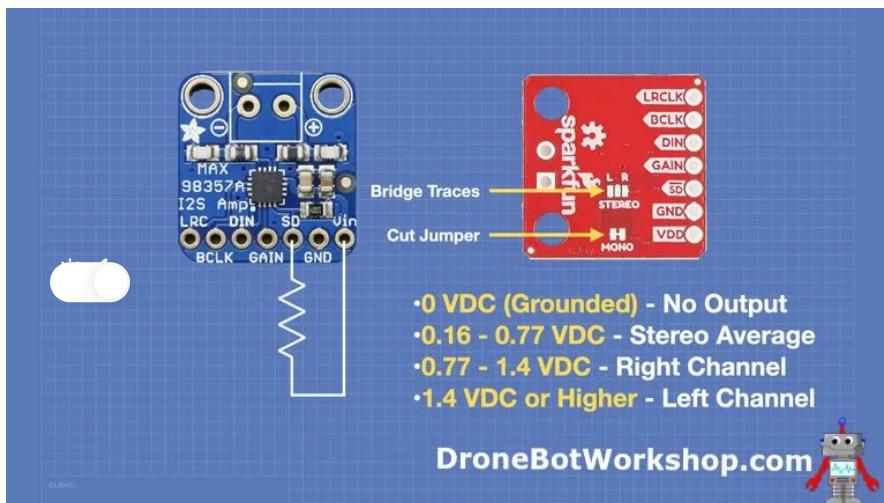
If you connect successfully, you will see the audio stream name and status displayed in the serial monitor.

Running in Stereo

We can modify our Internet Radio to produce a stereo output by simply adding another I2S amplifier module. We will also add a potentiometer to use as a volume control, a basic but essential control for any audio device.

MAX98357A Channel Selection

The MAX98357A I2S amplifier module that we have been using in our experiments can output either the left audio channel, the right audio channel, or a mix of both.



By default, the device outputs a mix, which is essentially monophonic sound. But we can modify it to output just the left or just the right channel, so we can use two devices for stereo output.

The SD pin on the module is the key to selecting the channel output. Despite its confusing label, this is not the Serial Data input, instead, it is the channel selection pin.

The pin works with a control voltage, the voltage on this pin determines which channel the module will output.

By default, an internal pull-down resistor keeps the level between 0.16 and 0.77 volts, so the amplifier module will output a mixture of both channels. Note that you can also mute the amplifier by grounding the SD pin.

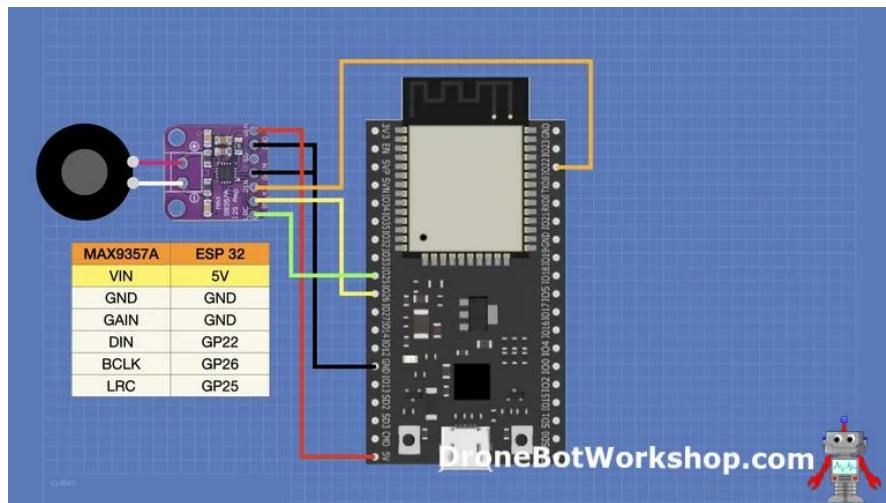
You can change the voltage on the pin using a pull-up resistor. The value will need to be determined by experimentation, as it is affected by the supply voltage.

I suggest using a 100k trimpot to set the voltage. You can then remove the trimpot, measure its resistance, and replace it with the standard resistor value that is closest.

If you purchase the Sparkfun version of the MAX98357A I2S amplifier module it is much easier, as the resistors are already there for you. You set the channel configuration by cutting the MONO jumper and bridging the STEREO pads to either L or R, depending upon which channel you want the module to output.

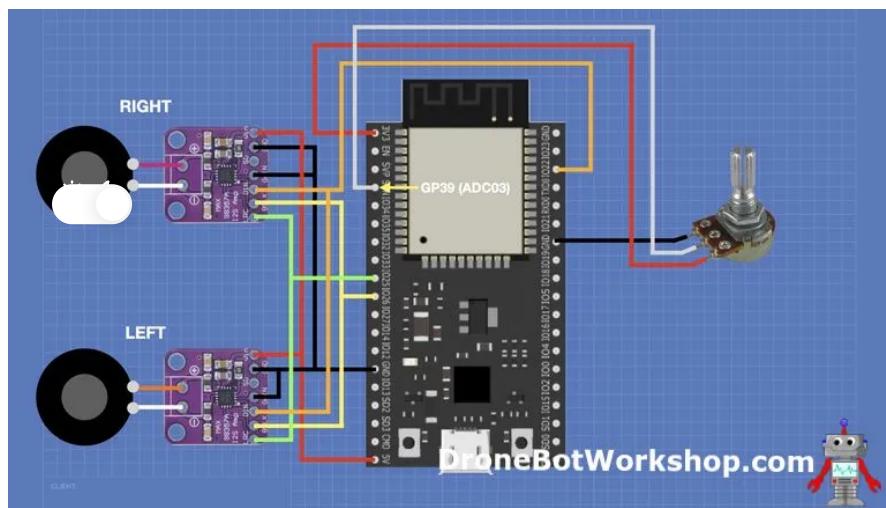
Stereo Radio Hookup

The hookup starts out using the same GPIO pins as our earlier Internet Radio, so you can just expand the wiring on that if you have built it already.



Uma diferença é que agora estou usando a linha de 5 volts do ESP32 para alimentar os amplificadores. Estou preocupado que alimentar dois amplificadores de 3 watts com o regulador interno ESP32 possa sobrecarregá-lo. 5 volts também fornecerão mais potência de saída.

Caso contrário, a fiação é idêntica para ambos os módulos amplificadores, basta conectá-los em paralelo.



Temos também um potenciômetro conectado ao pino 39 do GPIO, que também é ADC03. Precisamos usar um conversor analógico para digital no grupo inferior, pois estamos usando WiFi e quando você usa, não pode usar o grupo superior.

Além disso, certifique-se de conectar os alto-falantes estéreo em fase!

Código de rádio estéreo

Não há nada de especial no código para estéreo, já que o código anterior produzia saída estéreo. Estábamos apenas usando um amplificador mono para ouvi-lo.

As diferenças de código aqui são apenas para o controle de volume.

```

1  /*
2   Internet Radio with Volume Demo
3   esp32-i2s-radio-volume.ino
4   ESP32 I2S radio with volume control
5   Uses two MAX98357 I2S Amplifier Modules, strapped for Left and Right channel
6   Uses ESP32-audioI2S Library - https://github.com/schreibfaul1/ESP32-audioI2S
7
8   DroneBot Workshop 2022
9   https://dronebotworkshop.com
10 */
11
12 // Include required libraries
13 #include "Arduino.h"
14 #include "WiFi.h"
15 #include "Audio.h"
16
17 // Define I2S connections
18 #define I2S_DOUT 22
19 #define I2S_BCLK 26
20 #define I2S_LRC 25
21
22 // Define volume control pot connection
23 // ADC3 is GPIO 39
24 const int volControl = 39;
25
26 // Integer for volume level
27 int volume = 10;
28
29 // Create audio object
30 Audio audio;
31
32 // Wifi Credentials
33 String ssid = "YOUR_SSID";
34 String password = "YOUR_PASSWORD";
35
36 void setup() {
37
38   // Start Serial Monitor
39   Serial.begin(115200);
40
41   // Setup WiFi in Station mode
42   WiFi.disconnect();
43   WiFi.mode(WIFI_STA);
44   WiFi.begin(ssid.c_str(), password.c_str());
45
46   while (WiFi.status() != WL_CONNECTED) {
47     delay(500);
48     Serial.print(".");
49   }
50
51   // WiFi Connected, print IP to serial monitor
52   Serial.println("");
53   Serial.println("WiFi connected");
54   Serial.println("IP address: ");
55   Serial.println(WiFi.localIP());
56   Serial.println("");
57
58   // Connect MAX98357 I2S Amplifier Module
59   audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
60
61   // Set the volume
62   audio.setVolume(volume);
63
64   // Connect to an Internet radio station (select one as desired)
65   //audio.connecttohost("http://vis.media-ice.musicradio.com/CapitalMP3");
66   //audio.connecttohost("mediaserv30.live-nect MAX98357 I2S Amplifier Module
67   //audio.connecttohost("www.surfmusic.de/m3u/100-5-das-hitradio,4529.m3u");
68   //audio.connecttohost("stream.1a-webradio.de/deutsch/mp3-128/vtuner-1a");
69   //audio.connecttohost("www.antenne.de/webradio/antenne.m3u");
70   audio.connecttohost("0n-80s.radionetz.de:8000/0n-70s.mp3");
71
72 }
73
74 void loop()
75 {
76   // Run audio player
77   audio.loop();
78
79   // Get the volume level
80   volume = map ((analogRead(volControl)), 0, 4095, 0, 20);
81
82   // Set the volume
83   audio.setVolume(volume);
84
85 }
86
87 // Audio status functions
88
89 void audio_info(const char *info) {
90   Serial.print("info      "); Serial.println(info);
91 }
92
93 void audio_id3data(const char *info) { //id3 metadata
94   Serial.print("id3data    "); Serial.println(info);
95 }
96
97 void audio_eof_mp3(const char *info) { //end of file
98   Serial.print("eof_mp3    "); Serial.println(info);
99 }
100
101 void audio_showstation(const char *info) {
102   Serial.print("station    "); Serial.println(info);
103 }
104
105 void audio_showstreaminfo(const char *info) {
106   Serial.print("streaminfo "); Serial.println(info);
107 }
```

```

105 void audio_showstreamtitle(const char *info) {
106   Serial.print("streamtitle "); Serial.println(info);
107 }
108 void audio_bitrate(const char *info) {
109   Serial.print("bitrate "); Serial.println(info);
110 }
111 void audio_commercial(const char *info) { //duration in sec
112   Serial.print("commercial "); Serial.println(info);
113 }
114 void audio_icyurl(const char *info) { //homepage
115   Serial.print("icyurl "); Serial.println(info);
116 }
117 void audio_lasthost(const char *info) { //stream URL played
118   Serial.print("lasthost "); Serial.println(info);
119 }
120 void audio_eof_speech(const char *info) {
121   Serial.print("eof_speech "); Serial.println(info);
122 }

```

Observe como mais uma vez o uso da biblioteca torna isso muito fácil.

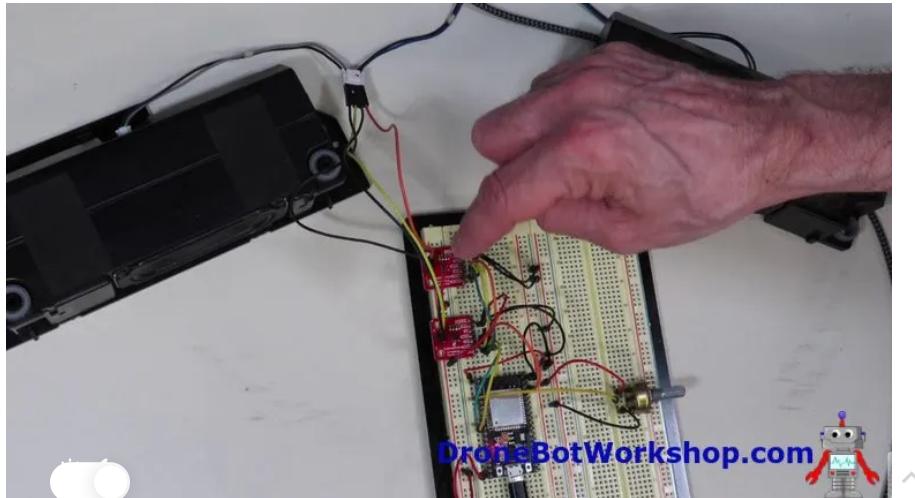
Adicionamos alguns números inteiros para lidar com a conexão do potenciômetro e o nível de volume.

A maior parte do código na configuração é idêntico ao do rádio na Internet anterior.

A única mudança real está no Loop. Ainda executamos o método Loop do objeto de áudio, mas também pegamos a entrada do potenciômetro e a mapeamos para um intervalo de 0 a 20. Em seguida, usamos isso para controlar o volume.

Testando o rádio estéreo

Testar o rádio é essencialmente igual ao da versão anterior, carregue o código, pressione reset e observe o monitor serial. Depois de conectar-se a uma estação, você deverá ouvir o programa, desta vez em um maravilhoso estéreo.



Conclusão

Ser capaz de manipular áudio digital aumenta a lista crescente de aplicações para o ESP32. Você provavelmente pode imaginar vários projetos interessantes usando os recursos I2S das placas, e as placas periféricas são baratas e muito fáceis de trabalhar.

Então não, isso não é um erro ortográfico de "I2C" na folha de especificações do ESP32. I2S é apenas mais um excelente recurso do que está rapidamente se tornando o microcontrolador mais versátil do mundo.

Lista de peças

Aqui estão alguns componentes que você pode precisar para concluir os experimentos deste artigo. Observe que alguns desses links podem ser links afiliados e o DroneBot Workshop pode receber uma comissão sobre suas compras. Isso não aumenta o custo para você e é um método de suporte a este site sem anúncios.

EM BREVE!

Recursos

[Código usado aqui](#) – Todo o código usado neste artigo, compactado em um arquivo ZIP.

[Biblioteca ESP32-AudioI2S](#) – A biblioteca ESP32-AudioI2S no GitHub

Resumo



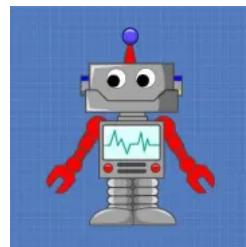
Nome do artigo Som com ESP32 - Protocolo I2S

Descrição Aprenda a usar o protocolo de áudio I2S com o microcontrolador ESP32. Aprenderemos como funciona o I2S e depois construiremos um MP3 player e um Internet Radio.

Autor Oficina DroneBot

Nome do editor Oficina DroneBot

Logotipo do editor



Marcado em: [Tutorial ESP32](#)

Oficina DroneBot 22 de maio de 2022 ESP32, tutorial 58 comentários

← Usando interrupções do Arduino – Hardware, mudança de pinos e temporizador

Usando módulos LCD redondos GC9A01 →

Se inscrever ▾

Participe da discussão

58 COMENTÁRIOS

Mais antigo ▾

Gudu 1 ano atrás

Este tutorial é uma dádiva de Deus. Tenho me esforçado para projetar um player assim, completo com display BT, e essa aula realmente abriu meus olhos e iluminou os erros em meu projeto. Muito obrigado e seja sempre abençoado.

Responder

Duro Dethe ⏱ 1 ano atrás

Tenho tentado transmitir um link de rádio, mas não está funcionando. Você poderia experimentar e me informar qual poderia ser o problema?

link: <https://us.everestcast.com:1140/zhzokcgz/1/winamp.m3u>

Obrigado

👉 Responder

Paarth ⏱ 1 ano atrás

Senhor, uma pergunta, no vídeo você disse que o Grupo ADC2 não pode ser usado enquanto o WiFi estiver em uso. Então, por que os pinos i2s estão conectados ao ADC2 e como isso funciona?

Obrigado, Paarth

👉 Responder

Alex Grutter ⏱ 7 meses atrás

| 🎧 Resposta a [Paarth](#)

Ainda não tenho certeza sobre esse chip específico, mas normalmente é uma questão de “pin muxing”. Na maioria dos microcontroladores, um pino externo pode ser conectado a uma variedade de periféricos internos do chip, como ADC, I2S e SPI/UART. Vou assumir que o WiFi usa um canal SPI ou UART que está conectado apenas a um conjunto de pinos, e esses pinos são compartilhados com as opções de pinos externos dos periféricos ADC. Claro como lama?

🕒 Última edição feita 7 meses atrás por Alex Grutter

👉 Responder

Wayne ⏱ 1 ano atrás

Bill, obrigado por este artigo! Meu pai adorava sua rádio Grace Digital na Internet, mas infelizmente ela saiu do ar quando o agregador de estações de rádio Reciva fechou permanentemente no ano passado. A rádio na Internet que você criou aqui é muito simples. A placa MakerHawk MAX98357 I2S chegou hoje. Com seu excelente esboço, em uma hora eu tinha rádio na Internet saindo de um alto-falante. A alegria que meu pai terá quando eu trouxer para ele um substituto baseado em seu tutorial será inexprimível. Obrigado novamente!

👉 Responder



Hovhannes ⏱ 1 ano atrás

Olá, como conectar a porta i2c0 à entrada DAC nativa

👉 Responder

Jailton Bittencourt ⏱ 1 ano atrás

Olá Bill, bom dia.

Estou tendo o seguinte problema ao executar o código no meu ESP-32 WROOM.

solicitação de informações <http://stream.1a-webradio.de/deutsch/mp3-128/vtuner-1a> falhou!

Tentei uma estação web diferente com o mesmo problema.

Alguma idéia de como resolvê-lo?

Obrigado

👉 Responder

Tomás ⏱ 1 ano atrás

| Resposta a [Jailton Bittencourt](#)

Obrigado pelo tutorial!

Há um erro: Na foto da conexão do INMP441 as linhas 32 e 33 estão misturadas. Tirando isso funciona muito bem!

Atenciosamente

, Tomás

Responder

Tomás 1 ano atrás

Obrigado pelo tutorial!

Há um erro: Na foto da conexão do INMP441 as linhas 32 e 33 estão misturadas.

Tirando isso funciona muito bem!

Atenciosamente

, Tomás

Responder

Mike P. 1 ano atrás

Olá,

excelente artigo e vídeo, como sempre!

Alguém entende as funções declaradas da linha 88 em diante no esboço acima? Eles estão sendo chamados, pois geram suas informações quando o esboço é executado, mas o que os está chamando?

Além disso, uma string char constante é declarada entre colchetes de parâmetro, o que não faz sentido para mim.

Estou tentando extrair a taxa de bits do áudio para exibição em um LCD da minha rádio de internet, o que fiz, mas a taxa de bits está mostrando a taxa de bits do canal anterior.?!

Responder

Glyn 8 meses atrás

| Resposta a [Mike P](#)

<https://github.com/schreibfaul1/ESP32-audioI2S/blob/master/src/Audio.cpp>

As funções estão sendo chamadas se existirem no código audioI2S real.

Responder



Peter 1 ano atrás

Descobri que recebi apenas ruído do meu microfone até mudar de ESQUERDA para DIREITA ou inverter o sentido da linha E/D.

Responder

Paulo M. 7 meses atrás

| Resposta a [Pedro](#)

Eu tive a mesma experiência – alguém já descobriu isso?

Responder

Peter 1 ano atrás

Agora mudei para a rádio na Internet. Falha com:

solicitação de informações <http://www.surfmusic.de/m3u/100-5-das-hitradio.4529.m3u> falhou!

para cada um dos canais de origem que experimentei. E cada um deles funciona bem no meu

Logitech Media Server.

Alguma ideia?

 Responder

Peter  1 ano atrás

|  Resposta a [Pedro](#)

O texto acima foi compilado com VS Code. Tentei novamente com o Arduino IDE e funcionou perfeitamente.
Não faço ideia do que está acontecendo aqui.....
Mas os exercícios estão muito bem explicados.

 Responder

Alfredo  1 ano atrás

Pergunta: É possível dispensar os acessórios externos e utilizar as saídas DAC (pinos 25 e 26) do ESP32 para obter o áudio em modo estéreo?

 Responder

James Holbrook  1 ano atrás

Agora, mostre-nos como emitir o áudio através de Bluetooth para uso em alto-falantes ou fones de ouvido Bluetooth.

 Responder

JeonLab  1 ano atrás

Ótimo tutorial sobre o I2S do ESP32. Usei o ESP32 para meus próprios projetos, mas não usei a parte I2S que estou interessado em experimentar. Obrigado por compartilhar este artigo.

 Responder

MVARAGAO  1 ano atrás

Olá!
Parabéns!

Quando executo o "Código do módulo de microfone INMP441", recebo o erro:
'I2S_COMM_FORMAT_STAND_I2S' não foi declarado neste escopo

Como resolver isso?

Obrigado.

 Última edição feita há 1 ano por MVARAGAO

 Responder

Pepetrick  1 ano atrás

|  Resposta a [MVARAGAO](#)

tive o mesmo problema, parece funcionar com
I2S_COMM_FORMAT_I2S_MSB, sugestão encontrada na internet

 Responder

wagner  7 meses atrás

|  Resposta ao [Pepetrick](#)

atualize o esp32 para a versão 2.0.6, aqui resolveu.

 Responder

URSO DE PELÚCIA  1 ano atrás

Olá, não vi em lugar nenhum a não ser qual ESP32 você usou para isso. Estou muito preocupado em comprar o errado e não consigo encontrar um com boa

documentação. Ajudaria muito se eu soubesse disso por alguém que tem muito mais conhecimento sobre essas coisas do que eu.

→ Responder

bubu ① 11 meses atrás

| ↗ Respondendo ao [TEDDY](#)

Você não pode errar, todos os chips ESP32 suportam I2S. Mas compre um “módulo de desenvolvimento” (chip ESP32 + mais alguns componentes), não apenas o chip ESP32 em si, para que você possa interagir com ele com mais facilidade. Por exemplo, nesta página https://www.aliexpress.com/item/32905750373.html?spm=a2g0o.order_detail.0.0.41b5f19cSExWa7 obtenha aquele com os pinos saíndo do PCB.

→ Responder

Bob H. ① 11 meses atrás

Não consegui compilar com a biblioteca mencionada no código de exemplo. No Github, acabo com “ESP32-audiol2s-2.0.5” em vez de “Audio.h”.

→ Responder

Prospector ① 11 meses atrás

1 Your radio works fine for me, but unfortunately the transmission is interrupted

→ Responder

Weiguang Chen ① 11 meses atrás

Ótimo artigo, posso perguntar se a biblioteca ESP32-2.0 é compatível com este código? Além disso, quero usar esp32 para receber 4 sinais de microfone do inmp441 ao mesmo tempo

→ Responder

Nilesh ① 11 meses atrás

Podemos usar o recurso Blue tooth do ESP32 para conectar o alto-falante Bluetooth em vez de usar qualquer módulo amplificador?

→ Responder

bubu ① 11 meses atrás

| ↗ Resposta a [Nilesh](#)

Talvez, se você reproduzir apenas músicas de um cartão micro SD. Definitivamente não, se você deseja transmitir música, mesmo de sua própria rede. ESP32 não pode usar WiFi e Bluetooth ao mesmo tempo.

→ Responder

Hoan ① 11 meses atrás

Quero que o inmp441 e o Max98357A rodem ao mesmo tempo, é possível?, pois vejo que você está usando o mesmo GP25

→ Responder

Jean Paul ① 11 meses atrás

Como posso usar DACs internos, por favor?

→ Responder

Jean Offenberg ⏱ 10 meses atrás

Muito sólido explicado, vou começar a comprar coisas e começar. Tx.

→ Responder

zen ⏱ 10 meses atrás

Oi,

Há um erro no seu código do MP3 player e ele não funcionará.

Você tem:

```
#define SPI_MISO 19  
#define SPI_SCK 18
```

Deveria ser:

```
#define SPI_MISO 18  
#define SPI_SCK 19
```

Grande projeto, porém, estou muito satisfeito por ter criado isso.

Obrigado

→ Responder

Byron Sterling ⏱ 9 meses atrás

Eu gostaria de ter encontrado você muito antes. Você salvou meu semestre! Estarei de volta no intervalo para mais. Tenho um projeto com o ESP32 e uma câmera com rastreamento.

Esta é uma coleção incrível de conhecimento e suporte.

Obrigado.

→ Responder

Diego ⏱ 9 meses atrás

Olá, parabenizo você pela excelente contribuição.

Estou precisando usar um ESP32-S3 para receber de um ADC por I2S, processá-lo no ESP como um DSP e transmiti-lo de volta por I2S. Pretendo receber 8 canais e transmitir esses 8 em algum mix de 4 canais. Você sabe se isso pode ser feito? Entendo que o ESP tem duas portas mas não sei se podem ser usadas ao mesmo tempo

Desa...to obrigado

→ Responder



daij25 ⏱ 9 meses atrás

Que dispositivo (amplificador) eu poderia usar para enviar para um alto-falante amplificado ou, pelo menos, para uma entrada de nível de linha?

→ Responder

Wim Rijk ⏱ 9 meses atrás

Olá Bill,
obrigado pelas excelentes aulas que deixam tudo muito claro para mim.

→ Responder

Paulo Pezalla ⏱ 8 meses atrás

Eu sou um amador nesse tipo de coisa, então, por favor, se você tiver a gentileza de responder, mantenha as coisas simples.
Estou tentando compilar o esboço do MP3 Player.

```
// E/S digital usada
#define SD_CS 5
#define SPI_MOSI 23
#define SPI_MISO 19
#define SPI_SCK 18
#define I2S_DOUT 25
#define I2S_BCLK 27
#define I2S_LRC 26
```

Áudio áudio;

Ele para na última linha com a mensagem de erro “Áudio' não nomeia um tipo”.

Agradeceríamos muito a orientação sobre como corrigir isso.

Responder

Rony 8 meses atrás

Resposta a [Paul Pezalla](#)

A biblioteca está faltando, leia o artigo novamente para saber como fazer.

Responder

JB 8 meses atrás

Bom dia Bill,

só uma sugestão sobre o problema dos direitos autorais do youtube: para mostrar a música basta conectar um LED (com resistor?) no lugar do alto-falante! Então você não ouvirá a música (sem problemas de direitos autorais), mas você (nós) veremos a modulação!

Obrigado pelo seu bom trabalho.

Responder

Sam 8 meses atrás

Não consigo fazer com que o arquivo mp3 do cartão SD seja reproduzido em loop; em vez disso, ele é reproduzido apenas uma vez. Eu pensei audio.loop(); deveria significar repetir o mesmo arquivo repetidamente.

Então encontrei nesta página

<https://github.com/schreibfaul1/ESP32-audiol2S/wiki>

que menciona que podemos definir isso

```
1 br
2     fileLoop(true);
```

No entanto, sem sucesso depois de tentar colocá-lo no void loop, void setup. alguém já passou por esse problema antes?

Responder

Alessandro Ferreira 8 meses atrás

olá, excelente trabalho, parabéns, gostaria de conhecer esse projeto, seria possível fazer uma conexão entre 2 pontos wireless em Frequência Modulada, um utilizando o conversor A/D de um lado e o outro um conversor D/A ?

Responder

Alex 8 meses atrás

TNX para todas as informações,
uma pergunta simples:
de uma fonte I2S, por exemplo um microfone, podemos enviar áudio para bluetooth
(usar o ESP como transmissor bluetooth)?

Responder

Dave  8 meses atrás

"O áudio digital com qualidade de CD tem uma resolução de 16 bits e uma taxa de amostragem de 44,1 kHz, enquanto o áudio digital com qualidade de telefone tem 8 bits e é amostrado a 8 kHz." está OK, mas acho que vale a pena mencionar que a qualidade também depende dos #canais, ou seja: estéreo no primeiro caso, mono no último.

 Responder

Alex  7 meses atrás

Olá! Obrigado pelo tutorial! Só uma dúvida, no código do microfone o número de amostras de uma leitura é definido da seguinte forma:

```
int16_t samples_read = bytesIn / 8;
```

Mas essa divisão não deveria ser por 2? Já que bits_per_sample é 16 (também conhecido como 2 bytes).

Além disso, por que precisamos calcular a média, imprimir os valores diretos não seria mais preciso? Obrigado!

 Responder

Raj  7 meses atrás

Olá, segui a configuração do cartão i2s/sd, mas estou com um problema, o mp3 que estou reproduzindo trava após 2 segundos de reprodução

 Responder

Francisco Augusto Medeiros-Logeay  7 meses atrás

Olá,
este artigo e seu vídeo do YouTube são realmente úteis! Muito obrigado!

Fiquei me perguntando uma coisa: ao fazer o experimento estéreo com dois MAX98357, você preferiu usar 5V. Acontece que pretendo funcionar com bateria, então não terei 5V. Funcionará bem? E será alto o suficiente? Meu plano é montar um pequeno rádio de internet para a cozinha.

Atenciosamente,
Francisco

 Responder

Chanaier  5 meses atrás

Melhor tutorial que encontrei até agora. Obrigado!

 Responder

verão  5 meses atrás

Bom trabalho.

Você tem algum código para ESP32S3 usado I2S ou DMA?

Obrigado.

 Responder

DanM  5 meses atrás

Comecei a assistir seus vídeos no YouTube há algum tempo, com uma certa estação móvel de rádio 2 DJ que minha mãe GF não poderia receber se ela morasse. Uma pesquisa rápida encontrou áudio esp32, escavado em bits e bobs para pedidos de chips MAX e temos um volume de estação única funcional controlado em um antigo gabinete de alto-falante de PC de 3W. Obrigado. Agora só adicionar mais alguns presets e dourado!

Responder

Shan 5 meses atrás

Este é um ótimo projeto para começar com som. Como podemos passar para alto-falantes reais agora com muito mais potência que pode encher uma sala com som?
Podemos continuar a usar o MAX98357 ou precisamos encontrar outro amplificador?

Responder

Danilo há 4 meses

Olá,
é possível alterar o áudio reproduzido em tempo de execução? Não consigo fazer isso usando o connecttoFS(SD, newAudio).
Muito obrigado!

Responder

Andy há 4 meses

Acabei de fazer isso e está funcionando muito bem.
Fiz a rádio internet (mono) e adicionei o potenciômetro de volume.

Responder

Tony há 4 meses

Tentei apenas acessar o cartão micro SD. Nenhum amplificador conectado.
Infelizmente, continuo recebendo a mesma mensagem de erro: « Erro ao acessar o cartão microSD! ». A fiação foi verificada. O tamanho do cartão SD pode causar esse erro? Está no ESP32.

Responder

Frederico há 4 meses

Olá, você sabe onde posso conseguir a biblioteca <driver/i2s.h>? Este projeto funcionará usando um wemos d1 mini (tem um esp8266 em vez de um esp32)?

Responder

Shankar 3 meses atrás

há orientação da API i2s para ESP32, mencionada i2s_std_config_t, mas no seu exemplo alguma outra API, qual usar, como definir a taxa de amostragem necessária e o relógio (que cronometra BCLK, MCLK) no modo escravo i2s

Responder

surya 3 meses atrás

No ESP32 MP3 Player podemos reproduzir o áudio armazenado no SPIFFS em vez de usar o micro SDcard, isso é possível

Última edição feita 3 meses atrás por Surya

Responder

Michael Allen 1 mês atrás

tentei algo semelhante à configuração do rádio estéreo. estou tentando enviar um fluxo mono para dois alto-falantes diferentes. tenho 1 esp32 conectado em paralelo a 2 DACs. Cada DAC está conectado a um amplificador de 100W. e cada amplificador está conectado a 1 alto-falante. o esp32 alimenta os DACs em paralelo

e uma fonte de alimentação separada alimenta os amplificadores em paralelo.
quando eu ligo todos juntos, não ganho nada. se eu desconectar o LRC, BCLK e DIN
de um dos DACs, obtenho áudio de um alto-falante. O que estou fazendo de errado?
eu coloquei um ... [Leia mais »](#)

Responder

Steve 1 mês atrás

Belo texto.

Infelizmente, na demonstração em MP3, recebo um erro:

Tenho que indicar o caminho do áudio, caso contrário o padrão usa a versão Teensy
e falha.

```
#include "C:\Users\steve\Documents\Arduino\libraries\ESP32-audioI2S-
master\src\Audio.h"
```

Mas então recebo um erro de compilação:

'Áudio' não nomeia um tipo

Não tenho certeza de como consertar isso

Responder

Categorias

Projeto Rover de 6 Rodas

Arduíno

Construa um robô REAL

Eletrônicos

ESP32

Apresentou

Em geral

IoT

Projeto

Quad'

Raspberry Pi

Projetos Raspberry Pi 10

Robôs

Programas

Tutorial

Fornecedores

O que há de novo?

Seeeduino XIAO ESP32S3 Sense Board – Uma pequena câmera ESP32

LoRa – Primeiros passos com Arduino, ESP32 e Pico

ArduinoNano ESP32

Arduino Uno R4 – Mínimos e WiFi

Detectão de objetos ESP32-CAM com impulso de borda

Amplificadores operacionais – blocos de construção analógicos

Arduino Giga R1 WiFi

Use um controlador PS3 com um ESP32

Usando ChatGPT para escrever código para Arduino e ESP32

Usando o DAC ESP32 (e fazendo um instrumento frutado)

Carro robô de roda Mecanum com controle remoto ESP-NOW

Conduzindo motores DC com microcontroladores

Meça a qualidade do ar com microcontroladores

Pico W com o Arduino IDE

Construindo uma estação de trabalho de inicialização dupla

O que é popular

Arduíno

Tutoriais

ESP32

Eletrônicos

Projetos

Raspberry Pi

Primeiros passos com ESP32

Som com ESP32 - Protocolo I2S

Usando servo motores com ESP32

Usando módulos LCD redondos GC9A01



© 2023 Workshop DroneBot