Pythonコード

// 必要なライブラリをインクルード

#include <WiFi.h>

#include <HTTPClient.h> // HTTP通信用

#include "Audio.h" // オーディオ再生用ライブラリ

// Wi-Fi設定

const char\* ssid = "ishiさんのS23"; // ご自身のWi-Fiネットワーク名

const char\* password = "fcsjvy9bwsx29sa"; // ご自身のWi-Fiパスワード

// 再生するオーディオファイルのURL

const char\* audioURL = "http://10.58.174.247:5000/uploads/天使の声援.wav";

// Audioライブラリのインスタンス

Audio audio;

// --- GPIOピンの定義 ---

// あなたの回路接続に合わせて、これらのGPIO番号を正確に設定してください！

// MAX98357のアンプ出力用

#define I2S\_BCLK\_PIN 26 // MAX98357のBCLKとINMP441のSCKが接続されているGPIOピン

#define I2S\_LRCL\_PIN 25 // MAX98357のLRCLとINMP441のWSが接続されているGPIOピン

#define I2S\_DOUT\_PIN 23 // MAX98357のDINが接続されているGPIOピン

// INMP441マイク入力用 (このコードではまだ使用しませんが、ピン定義として記載)

#define I2S\_DIN\_MIC\_PIN 35 // INMP441のSDが接続されているGPIOピン

// L/RピンとGNDの接続は物理的な配線で完結するので、コードでの定義は不要

void setup() {

// シリアル通信の初期化（デバッグ出力用）

Serial.begin(115200);

// Wi-Fi接続開始

Serial.print("Connecting to WiFi...");

WiFi.begin(ssid, password);

// Wi-Fi接続が確立するまで待機

while (WiFi.status() != WL\_CONNECTED) {

delay(500); // 500ミリ秒待機

Serial.print("."); // 進捗を示すドットを表示

}

Serial.println("Connected!"); // 接続成功メッセージ

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); // ESP32に割り当てられたIPアドレスを表示

// --- オーディオ設定 ---

// audio.setPinout(BCLK\_PIN, LRCL\_PIN, DIN\_PIN);

// ここで、アンプへのI2S出力ピンを設定します。

// あなたのMAX98357アンプの接続に合わせてピン番号を記述します。

// BCLK (D26), LRCL (D25), DIN (D23)

audio.setPinout(I2S\_BCLK\_PIN, I2S\_LRCL\_PIN, I2S\_DOUT\_PIN);

// MAX98357のSDピンを3V3に接続しているため、常にONです。

// もしSDピンをESP32のGPIOピンに接続して制御する場合は、

// ここで pinMode(SD\_PIN, OUTPUT); digitalWrite(SD\_PIN, HIGH); などでONにします。

// 指定されたURLからオーディオファイルを再生

Serial.print("Connecting to audio source: ");

Serial.println(audioURL);

audio.connecttohost(audioURL);

// 接続エラーやファイルが見つからないなどのイベントを処理するためのコールバック関数を設定することもできます

// audio.setConnectionStateHandler(handleConnectionState);

}

void loop() {

// audio.loop() は、オーディオ再生に必要な内部処理を継続的に行います。

// これがないと音声が途切れたり、再生されなかったりします。

audio.loop();

// 再生が終了したら、何か別の処理をしたい場合（例：繰り返し再生、次のファイルを再生など）

// if (!audio.isRunning() && audio.getLastStatusCode() == 200) {

// Serial.println("Audio playback finished. Reconnecting...");

// audio.connecttohost(audioURL); // 繰り返し再生の例

// }

}

// 必要に応じて、オーディオイベントを処理する関数

// void audio\_info(const char \*info){

// Serial.print("Info: "); Serial.println(info);

// }

// void audio\_id3data(const char \*info){

// Serial.print("ID3data: "); Serial.println(info);

// }

// void audio\_eof\_mp3(const char \*info){

// Serial.print("EOF MP3: "); Serial.println(info);

// }

// void audio\_showstation(const char \*info){

// Serial.print("Station: "); Serial.println(info);

// }

// void audio\_showstreamtitle(const char \*info){

// Serial.print("Streamtitle: "); Serial.println(info);

// }

// void audio\_bitrate(const char \*info){

// Serial.print("Bitrate: "); Serial.println(info);

// }

// void audio\_commercial(const char \*info){

// Serial.print("Commercial: "); Serial.println(info);

// }

// void audio\_icyurl(const char \*info){

// Serial.print("Icy Url: "); Serial.println(info);

// }

// void audio\_lasthost(const char \*info){

// Serial.print("Last Host: "); Serial.println(info);

// }

// void audio\_eof\_speech(const char \*info){

// Serial.print("EOF Speech: "); Serial.println(info);

// }

Arduinoコード

// the setup function runs once when you press reset or power the board

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

// ここを修正！ LED\_BUILTIN を GPIO2 に変更します

pinMode(2, OUTPUT); // ESP32の多くのボードで内蔵LEDはGPIO2に接続されています

}

// the loop function runs over and over again forever

void loop() {

// turn the LED on (HIGH is the voltage level)

// ここを修正！ LED\_BUILTIN を GPIO2 に変更します

digitalWrite(2, HIGH);

delay(1000); // wait for a second

// turn the LED off by making the voltage LOW

// ここを修正！ LED\_BUILTIN を GPIO2 に変更します

digitalWrite(2, LOW);

delay(1000); // wait for a second

}