# Speech-to-Text Converter 1. Objective

The goal of this project is to develop a compact and reliable speech-to-text system using the Seeed Studio XIAO ESP32S3 Sense microcontroller. The system captures voice via the onboard microphone, stores it as a .wav file, and sends it to the Deepgram Speech Recognition API for transcription.

# 2. Hardware Components

|  |  |
| --- | --- |
| Component | Description |
| XIAO ESP32S3 Sense | Microcontroller with integrated MEMS mic |
| USB-C Cable | Must support data (not just charging) |
| PC/Laptop | Used for code deployment and monitoring |
| (Optional) Button | Used for manual audio trigger input |

# 3. Software & Environment

- Arduino IDE ( v2.0+)  
- Board Support: Seeed XIAO ESP32S3  
- Partition Scheme: 4MB Flash with SPIFFS / Default with SPIFFS  
- Required Libraries:  
 - WiFi.h – Connect to internet  
 - HTTPClient.h – Send POST request to API  
 - SPIFFS.h / FS.h – File system for temporary WAV storage  
 - I2S.h – Audio capture from digital MEMS mic

## 3.1 Deepgram API Setup

To access the Deepgram Speech Recognition services, you need a valid API key. Follow the steps below to create and configure it:  
  
1. Visit the Deepgram Developer Portal: https://console.deepgram.com  
2. Log In or Sign Up using your Google, GitHub, or email account.  
3. Create a New Project:  
 - Go to Projects → New Project → Enter a name → Create  
4. Generate an API Key:  
 - Inside the project, go to API Keys tab → Create API Key → Give it a name → Create  
5. Copy the Secret Key Immediately:  
 - The key will only be shown once,it will be hidden in website itself . Save it securely.  
6. Use the key in your code like so:  
 const char\* deepgramApiKey = "your\_actual\_key\_here";

# 4. Project Architecture

[User Speech] → [Onboard Mic ] → [ESP32S3 - Audio Buffering → WAV File] → [Deepgram API] → [ESP32S3 - JSON Parsing] → [Serial Monitor Output]

# 5. Implementation Workflow

1. Wi-Fi Setup  
 - ESP32 connects to local network using standard Wi-Fi credentials.  
2. Microphone Configuration  
 - Onboard mic is initialized using I2S protocol.  
3. Audio Capture  
 - 5 seconds of 16-bit mono audio at 16kHz is saved as a .wav file to SPIFFS.  
4. API Call to Deepgram  
 - WAV file is sent via HTTP POST to Deepgram API.  
5. Response Handling  
 - Transcription is extracted from JSON and printed to serial monitor.

# 6. Output Validation

Sample Serial Output:  
📶 Connecting to WiFi...  
✅ WiFi connected  
🎤 Recording...  
✅ Recording complete  
🗣️ Recognized Speech:  
turn on the fan

# 7. Optional Extensions

- OLED Display: Show live transcription  
- Web Dashboard: View text via ESP32-hosted webpage  
- Button Trigger: Start recording via GPIO  
- Voice Commands: Map keywords to actions  
- Real-time Stream: Use WebSocket version of Deepgram API

# 8.Source Code (Main.ino)

#include <I2S.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <SPIFFS.h>

#include <FS.h>

const char\* ssid = "your\_hotspot\_name";

const char\* password = “your\_password”;

const char\* deepgramApiKey = "your\_actual\_key\_here ";

#define SAMPLE\_RATE 16000

#define BITS\_PER\_SAMPLE 16

#define CHANNELS 1

#define RECORD\_TIME\_SEC 5

#define FILE\_NAME "/recorded.wav"

void writeWavHeader(File file, int dataLen) {

uint8\_t header[44] = {

'R','I','F','F', 0,0,0,0,'W','A','V','E','f','m','t',' ',

16,0,0,0, 1,0, 1,0, 0,0,0,0, 0,0, 0,0,

2,0, 16,0,'d','a','t','a', 0,0,0,0

};

int byteRate = SAMPLE\_RATE \* CHANNELS \* (BITS\_PER\_SAMPLE / 8);

int fileSize = dataLen + 36;

header[4] = fileSize;

header[5] = fileSize >> 8;

header[6] = fileSize >> 16;

header[7] = fileSize >> 24;

header[24] = SAMPLE\_RATE;

header[25] = SAMPLE\_RATE >> 8;

header[26] = SAMPLE\_RATE >> 16;

header[27] = SAMPLE\_RATE >> 24;

header[28] = byteRate;

header[29] = byteRate >> 8;

header[30] = byteRate >> 16;

header[31] = byteRate >> 24;

header[40] = dataLen;

header[41] = dataLen >> 8;

header[42] = dataLen >> 16;

header[43] = dataLen >> 24;

file.write(header, 44);

}

void setup() {

Serial.begin(115200);

delay(1000);

if (!SPIFFS.begin(true)) {

Serial.println("❌ SPIFFS mount failed");

return;

}

Serial.print("📶 Connecting to WiFi");

WiFi.begin(ssid, password);

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

delay(300);

Serial.print(".");

}

Serial.println("\n✅ WiFi connected");

Serial.println("🎙 Starting mic...");

I2S.setAllPins(-1, 42, 41, -1, -1); // BCK=-1, WS=42, SD=41

if (!I2S.begin(PDM\_MONO\_MODE, SAMPLE\_RATE, BITS\_PER\_SAMPLE)) {

Serial.println("❌ I2S init failed");

while (1);

}

File file = SPIFFS.open(FILE\_NAME, FILE\_WRITE);

if (!file) {

Serial.println("❌ Failed to open file");

return;

}

int totalBytes = SAMPLE\_RATE \* (BITS\_PER\_SAMPLE / 8) \* RECORD\_TIME\_SEC;

int16\_t buffer[512];

int bytesWritten = 0;

Serial.println("🎤 Recording...");

writeWavHeader(file, totalBytes);

unsigned long startTime = millis();

while ((millis() - startTime) < (RECORD\_TIME\_SEC \* 1000)) {

int bytesRead = I2S.read(buffer, sizeof(buffer));

if (bytesRead > 0) {

file.write((uint8\_t\*)buffer, bytesRead);

bytesWritten += bytesRead;

}

}

file.close();

I2S.end();

Serial.println("✅ Recording complete");

File audioFile = SPIFFS.open(FILE\_NAME);

if (!audioFile) {

Serial.println("❌ Audio file open failed");

return;

}

HTTPClient http;

http.begin("https://api.deepgram.com/v1/listen");

http.addHeader("Authorization", String("Token ") + deepgramApiKey);

http.addHeader("Content-Type", "audio/wav");

int httpResponseCode = http.sendRequest("POST", &audioFile, audioFile.size());

if (httpResponseCode > 0) {

String response = http.getString();

int start = response.indexOf("\"transcript\":\"") + 14;

int end = response.indexOf("\"", start);

String transcript = response.substring(start, end);

Serial.println("🗣 Recognized Speech:");

Serial.println(transcript);

} else {

Serial.print("❌ HTTP Error: ");

Serial.println(httpResponseCode);

}

audioFile.close();

http.end();

}

void loop() {

// idle

delay(2000);

setup();

}