Code for the Ai glasses

1. Code for http request from the esp32 to send:-

#include <WiFi.h>

#include <HTTPClient.h>

#include <SD.h>

const char\* ssid = "yourSSID";

const char\* password = "yourPASS";

const char\* serverUrl = "http://yourserver.com/upload";

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) delay(500);

if (!SD.begin()) {

Serial.println("SD init failed");

return;

}

File imageFile = SD.open("/photo.jpg");

File audioFile = SD.open("/sound.wav");

if (!imageFile || !audioFile) {

Serial.println("File open failed");

return;

}

WiFiClient client;

HTTPClient http;

String boundary = "----ESP32Boundary";

String contentType = "multipart/form-data; boundary=" + boundary;

http.begin(client, serverUrl);

http.addHeader("Content-Type", contentType);

// Build multipart body manually

String bodyStart = "--" + boundary + "\r\n";

bodyStart += "Content-Disposition: form-data; name=\"image\"; filename=\"photo.jpg\"\r\n";

bodyStart += "Content-Type: image/jpeg\r\n\r\n";

String bodyMiddle = "\r\n--" + boundary + "\r\n";

bodyMiddle += "Content-Disposition: form-data; name=\"audio\"; filename=\"sound.wav\"\r\n";

bodyMiddle += "Content-Type: audio/wav\r\n\r\n";

String bodyEnd = "\r\n--" + boundary + "--\r\n";

int totalSize = bodyStart.length() + imageFile.size() + bodyMiddle.length() + audioFile.size() + bodyEnd.length();

http.addHeader("Content-Length", String(totalSize));

int httpCode = http.sendRequest("POST");

client.print(bodyStart);

while (imageFile.available()) client.write(imageFile.read());

client.print(bodyMiddle);

while (audioFile.available()) client.write(audioFile.read());

client.print(bodyEnd);

imageFile.close();

audioFile.close();

http.end();

Serial.printf("HTTP response: %d\n", httpCode);

}

1. Code for recoding the audio:-

#include "SPIFFS.h"

#include "ESP\_I2S.h"

#define MIC\_DATA 41

#define MIC\_CLOCK 42

#define BUTTON\_PIN 2 // Push button connected to GPIO2

#define AUDIO\_FILE "/audio.wav"

#define SAMPLE\_RATE 16000

#define BITS\_PER\_SAMPLE 16

#define NUM\_CHANNELS 1

I2SClass i2s;

bool isRecording = false;

File audioFile;

unsigned long dataBytesWritten = 0; // track size of recorded data

void setup() {

Serial.begin(115200);

delay(1000);

Serial.println("=== Hold Button to Record WAV ===");

pinMode(BUTTON\_PIN, INPUT\_PULLUP); // Press = LOW

if (!SPIFFS.begin(true)) {

Serial.println("SPIFFS Mount Failed!");

while (1);

}

// Initialize I2S for internal mic

if (!i2s.begin(I2S\_MODE\_PDM\_RX, SAMPLE\_RATE, I2S\_DATA\_BIT\_WIDTH\_16BIT, I2S\_SLOT\_MODE\_MONO)) {

Serial.println("Failed to initialize I2S mic!");

while (1);

}

i2s.setPinsPdmRx(MIC\_CLOCK, MIC\_DATA);

Serial.println("Ready! Hold the button to start recording...");

}

void loop() {

if (digitalRead(BUTTON\_PIN) == LOW && !isRecording) {

startRecording();

}

if (digitalRead(BUTTON\_PIN) == HIGH && isRecording) {

stopRecording();

}

if (isRecording) {

recordAudioChunk();

}

}

void startRecording() {

Serial.println("🎙 Recording started...");

audioFile = SPIFFS.open(AUDIO\_FILE, "w");

if (!audioFile) {

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

return;

}

// Write placeholder WAV header

writeWavHeader(audioFile, 0);

dataBytesWritten = 0;

isRecording = true;

}

void stopRecording() {

Serial.println("🛑 Recording stopped.");

// Update WAV header with correct sizes

audioFile.seek(0);

writeWavHeader(audioFile, dataBytesWritten);

audioFile.close();

isRecording = false;

Serial.printf("✅ Saved %lu bytes of audio data to %s\n", dataBytesWritten, AUDIO\_FILE);

}

void recordAudioChunk() {

uint8\_t buffer[512];

size\_t bytesRead = i2s.read(buffer, sizeof(buffer));

if (bytesRead > 0) {

audioFile.write(buffer, bytesRead);

dataBytesWritten += bytesRead;

}

}

// ---------------- WAV HEADER FUNCTION ----------------

void writeWavHeader(File &file, uint32\_t dataSize) {

uint32\_t fileSize = dataSize + 36;

uint32\_t byteRate = SAMPLE\_RATE \* NUM\_CHANNELS \* (BITS\_PER\_SAMPLE / 8);

uint16\_t blockAlign = NUM\_CHANNELS \* (BITS\_PER\_SAMPLE / 8);

// RIFF header

file.write("RIFF", 4);

file.write((uint8\_t\*)&fileSize, 4);

file.write("WAVE", 4);

// fmt subchunk

file.write("fmt ", 4);

uint32\_t subchunk1Size = 16;

uint16\_t audioFormat = 1; // PCM

file.write((uint8\_t\*)&subchunk1Size, 4);

file.write((uint8\_t\*)&audioFormat, 2);

file.write((uint8\_t\*)&NUM\_CHANNELS, 2);

file.write((uint8\_t\*)&SAMPLE\_RATE, 4);

file.write((uint8\_t\*)&byteRate, 4);

file.write((uint8\_t\*)&blockAlign, 2);

file.write((uint8\_t\*)&BITS\_PER\_SAMPLE, 2);

// data subchunk

file.write("data", 4);

file.write((uint8\_t\*)&dataSize, 4);

}

1. Code for the python to recieve the files:-

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/upload', methods=['POST'])

def upload():

image = request.files['image']

audio = request.files['audio']

image.save("received\_photo.jpg")

audio.save("received\_sound.wav")

return "Files received", 200

1. code for the flask server to send the audio :-

from flask import Flask, request, send\_file

app = Flask(\_\_name\_\_)

@app.route('/get-audio', methods=['POST'])

def get\_audio():

data = request.json

print("ESP32 sent:", data)

# You can dynamically generate or select a WAV file based on `data`

return send\_file("output.wav", mimetype="audio/wav")