OptiSense: Audio Vision Assistant

Autor: Grosulescu Despina-Alexandra

Grupa: 333CD

Introducere

OptiSense este o solutie embedded portabila destinata persoanelor cu deficiente de vedere, care converteste automat textul tiparit din mediul fizic in format audio. Sistemul integreaza o arhitectura hardware dedicata cu servicii cloud de recunoastere optica a caracterelor (OCR), automatizand complet procesul de la capturarea imaginii pana la redarea vocala a continutului.

La baza proiectului se afla doua microcontrolere: ESP32-CAM, ales pentru capacitatile sale integrate de procesare video si conectivitate wireless, si ESP32S NodeMCU, responsabil exclusiv de logica de redare audio. Impreuna, cele doua placi formeaza un sistem distribuit: ESP32-CAM captureaza si transmite imaginea, un server intermediar realizeaza recunoasterea textului, iar NodeMCU descarca rezultatul si il reda printr-un difuzor extern.

Proiectul demonstreaza ca o arhitectura de tip vision-to-audio poate fi implementata intr-un format hardware compact si accesibil, punand accent pe eficienta fluxului de date, simplitatea utilizarii si autonomia dispozitivului. Scopul final este cresterea gradului de independenta a utilizatorilor in activitatile lor de zi cu zi.

Implementare si Justificari

Sistemul este complet integrat si functional. ESP32-CAM captureaza imagini stabile si le trimite prin retea catre un server Flask ce ruleaza pe un PC local. Serverul prelucreaza imaginea (corectie geometrica prin rotire si oglindire), apoi o transmite catre un API extern de OCR care returneaza textul identificat. In paralel, NodeMCU executa o bucla de interogare (polling) care descarca textul analizat si controleaza secvential redarea audio prin modulul DFPlayer Mini, gestionand pauzele dintre caractere.

Cele doua placi nu comunica direct intre ele prin fire, ci exclusiv prin intermediul serverului Flask, folosind protocolul HTTP peste WiFi. Aceasta decizie de arhitectura decupleaza complet modulul de captura de cel de redare, facand sistemul mai robust si mai usor de extins.

Protocoale de comunicatie folosite

- **HTTP/1.1 (TCP/IP peste WiFi)**: Protocolul principal de comunicatie intre placile embedded si server. ESP32-CAM foloseste metoda POST pentru a trimite imaginea JPEG catre ruta /upload a serverului Flask. NodeMCU foloseste metoda GET pentru a interoga periodic ruta /get_text si a

descarca textul extras de OCR. Ambele placi se conecteaza la aceeasi retea WiFi (hotspot) si comunica cu serverul identificat printr-o adresa IP statica (172.20.10.3:5000).

- **UART (Serial asincron)**: Folosit pentru doua scopuri distincte. In primul rand, pentru programarea si monitorizarea ESP32-CAM prin intermediul modulului FT232RL (USB-to-UART), la 115200 baud, pe pinii U0T/U0R. In al doilea rand, pentru comunicatia dintre NodeMCU si modulul DFPlayer Mini, pe portul hardware Serial2 (GPIO 16 – RX, GPIO 17 – TX), la 9600 baud, conform specificatiei modulului MP3.
- **SPI (Serial Peripheral Interface)**: Protocolul prin care modulul DFPlayer Mini citeste fisierele audio de pe cardul MicroSD intern. Aceasta comunicatie este gestionata complet de DFPlayer, transparent pentru NodeMCU, care nu are nevoie sa acceseze direct cardul SD.
- **I2C (SCCB)**: Protocolul folosit intern de ESP32-CAM pentru configurarea senzorului OV2640 (pinii SIOD – GPIO26, SIOC – GPIO27). Initializarea parametrilor camerei (rezolutie, calitate JPEG, flip/mirror) se face prin acest protocol la pornirea sistemului.

Motivarea bibliotecilor folosite

- **esp_camera.h**: Utilizata pe modulul de captura pentru interfatarea directa cu senzorul OV2640. Permite configurarea parametrilor video (rezolutie UXGA, compresie JPEG cu calitate 10) si controlul direct al buffer-ului de memorie PSRAM necesar stocarii temporare a cadrului inainte de trimitere. Fara aceasta biblioteca, initializarea senzorului ar fi necesitat programarea manuala a registrilor pe magistrala SCCB/I2C.
- **esp_http_client.h**: Biblioteca de nivel scazut folosita pe ESP32-CAM pentru constructia si trimiterea cererii HTTP POST cu payload binar (JPEG). A fost preferata fata de HTTPClient.h deoarece ofera un handler de evenimente (esp_http_client_event_handler_t) care permite citirea raspunsului serverului byte cu byte, necesara pentru a receptiona textul OCR returnat de Flask in acelasi request POST.
- **WiFi.h**: Ofera functionalitatile de baza pentru conectarea ambelor placi la reteaua locala (hotspot), facilitand arhitectura IoT complet wireless a sistemului. Folosita identic pe ambele placi pentru initializarea conexiunii si verificarea statusului in bucla principala.
- **HTTPClient.h**: Folosita pe NodeMCU pentru cereri HTTP GET periodice catre serverul Flask (polling). A fost aleasa pentru simplitatea API-ului sau de nivel inalt (http.begin(), http.GET(), http.getString()) care se potriveste perfect cu operatiunea simpla de descarcare a unui string text.
- **DFRobotDFPlayerMini.h**: Simplifica complet interactiunea hardware cu modulul MP3-TF-16P. In loc sa programeze manual pachetele de octeti pe magistrala seriala pentru fiecare comanda, biblioteca ofera functii predefinite. Comanda playLargeFolder(1, cod_ascii) translateaza direct codul ASCII al caracterului in adresa hardware a fisierului de pe cardul MicroSD, eliminand necesitatea unui tabel de mapare explicit.
- **PIL / Pillow (Python, pe server)**: Biblioteca Python esentiala in faza de pre-procesare a imaginii pe server. Folosita pentru corectia geometrica a pozei primite (rotire si aplicarea unui filtru mirror) inainte de a o trimite catre motorul de OCR, compensand orientarea fizica a camerei montate pe dispozitiv.
- **Flask (Python, pe server)**: Framework web Python folosit pentru constructia serverului intermediar. A fost ales datorita simplitatii de configurare a rutelor HTTP (/upload pentru POST si /get_text pentru GET), vitezei de prototipare si integrarii facile cu bibliotecile Python de procesare a imaginilor si OCR.

Elementul de noutate al proiectului

Noutatea consta in arhitectura distribuita cu server intermediar si decuplarea completa a modulului de captura de cel de redare. Cele doua microcontrolere nu comunica direct intre ele – tot schimbul de date trece prin serverul Flask, care actioneaza ca un broker. Aceasta abordare muta toata puterea de calcul necesara OCR-ului pe un sistem cu resurse suficiente (PC), lasand placile embedded sa

indeplineasca exclusiv rolul pentru care sunt optimizate: captura video si redare audio. Sistemul devine astfel scalabil (serverul poate fi inlocuit cu un serviciu cloud), usor de intretinut si extensibil cu noi functionalitati fara modificari hardware.

Justificarea utilizarii functionalitatilor din laborator

- **Lab 0 (GPIO)**: Utilizat pentru configurarea pinului butonului tactil (GPIO 13 pe ESP32-CAM) cu modul INPUT_PULLDOWN, eliminand necesitatea rezistentelor externe. Pinul pentru Flash LED (GPIO 4) este configurat ca OUTPUT si controlat direct prin digitalWrite() pentru iluminarea scenei la captura.
- **Lab 1 (UART / Comunicatie seriala)**: Aplicat in doua contexte. Primul: portul Serial hardware (U0T/U0R, 115200 baud) folosit impreuna cu modulul FT232RL pentru programarea ESP32-CAM si citirea mesajelor de debug in Serial Monitor. Al doilea: portul Serial2 hardware al NodeMCU (GPIO 16/17, 9600 baud) prin care se trimit comenzile de control catre DFPlayer Mini, respectand protocolul UART al acestuia.
- **Lab 3 (Timere si Polling)**: Bucla principala din NodeMCU implementeaza un mecanism de polling periodic: la fiecare 1000 ms (delay(1000)) este trimisa o cerere HTTP GET catre server. Intre redarea caracterelor audio sunt folosite intarzieri calibrate (delay(700) intre caractere, delay(1000) pentru spatii) care asigura sincronizarea corecta a sunetelor si evita suprapunerea redarilor.
- **Lab 5 (SPI)**: Protocolul SPI este folosit de modulul DFPlayer Mini pentru citirea fisierelor MP3 de pe cardul MicroSD. Desi aceasta comunicatie este gestionata intern de DFPlayer si nu direct de NodeMCU, intelegerea protocolului SPI (master/slave, linii MOSI/MISO/SCK/CS) este necesara pentru a intelege limitarile de viteza de acces si motivul pentru care organizarea fisierelor pe card influenteaza viteza de raspuns a modulului.

Descriere Generala

Functionarea sistemului este organizata in cinci etape distincte:

1. Captura

Utilizatorul pozitioneaza camera in dreptul unui text (eticheta, panou sau carte) si apasa butonul fizic. ESP32-CAM detecteaza tranzitia HIGH pe GPIO 13, declanseaza aprinderea flash-ului LED (GPIO 4) pentru 500 ms, apoi comanda senzorului OV2640 captarea unui cadru foto. Imaginea este stocata in buffer-ul PSRAM in format JPEG (rezolutie UXGA – 1600×1200, calitate 10 daca PSRAM este disponibil, SVGA altfel).

2. Transmisie

ESP32-CAM initializeaza o sesiune HTTP folosind biblioteca esp_http_client cu metoda POST. Imaginea JPEG din buffer este trimisa catre adresa serverului Flask (http://172.20.10.3:5000/upload) cu header Content-Type: image/jpeg. Conexiunea este mentinuta pana la primirea raspunsului de la server (timeout 60s). Textul returnat de server in corpul raspunsului HTTP este citit prin event handler-ul HTTP si stocat in variabila globala text_primit_global, dupa care este transmis pe Serial1 catre NodeMCU (varianta cu conexiune directa seriala intre placi – alternativa la varianta WiFi).

3. Procesare (pe server)

Imaginea receptionata pe ruta /upload este preluata ca stream de octeti si deschisa cu Pillow. Se

aplica corectiile geometrice necesare (rotire si mirror), apoi imaginea corectata este trimisa prin POST catre API-ul extern de OCR. Serviciul OCR returneaza un string cu textul recunoscut. Serverul curata rezultatul (elimina caractere speciale, pastreaza litere A-Z, cifre 0-9 si spatii) si il stocheaza intr-o variabila globala accesibila pe ruta /get_text.

4. Mapare si Interogare

NodeMCU executa la fiecare 1 secunda o cerere HTTP GET catre /get_text. Cand primeste un string nevid, il converteste la majuscule si il parcurge caracter cu caracter. Pentru fiecare caracter, functia getTrackNumber() calculeaza numarul track-ului corespunzator: cifrele 0-9 sunt mapate la track-urile 1-10, literele A-Z sunt mapate la track-urile 11-36. Numarul de track este transmis ca argument catre DFPlayer prin Serial2.

5. Redare

Pentru fiecare cod numeric calculat, comanda player.play(track) este trimisa catre DFPlayer Mini prin Serial2. DFPlayer acceseaza fisierul corespunzator de pe cardul MicroSD prin SPI, il decodeaza si trimite semnalul audio amplificat catre difuzorul de 3W/8Ω. Intre redari, programul asteapta 700 ms pentru a asigura redarea completa inainte de urmatorul caracter, respectiv 1000 ms la intalnirea unui spatiu.

Hardware Design

Componenta	Descriere
ESP32-CAM	Unitate centrala captura: procesor dual-core Xtensa LX6, 520KB SRAM, 4MB PSRAM, senzor OV2640, WiFi 802.11 b/g/n integrat Datasheet
ESP32S NodeMCU	Unitate centrala audio: procesor dual-core Xtensa LX6, WiFi 802.11 b/g/n integrat, 4MB Flash Datasheet
FT232RL	Modul USB-to-UART pentru programarea ESP32-CAM si monitorizare seriala; furnizeaza totodata alimentarea de 5V a placii
DFPlayer Mini MP3-TF-16P	Modul MP3 standalone cu decodor audio integrat, citire MicroSD prin SPI intern, amplificator audio 3W, control prin UART Datasheet
Difuzor 3W / 8 Ohm	audio pentru redarea fisierelor MP3 Datasheet
Card MicroSD 16GB	Stocare fisiere audio MP3; organizat in folderul /01/ cu fisiere numerotate
Buton tactil push	Declansare manuala captura foto pe ESP32-CAM
Fire jumper M-F si M-M	Conexiuni pe breadboard intre module

Alocarea Pinilor

ESP32-CAM

Componenta	Pin ESP32-CAM	Functie
FT232RL (TXD)	U0R (RX)	Receptie date seriale – programare si Serial Monitor
FT232RL (RXD)	U0T (TX)	Transmisie date seriale – programare si Serial Monitor
Buton captură	IO13	Detectare apasare buton (INPUT_PULLDOWN, activ HIGH)
Flash LED	IO4	Iluminare la captura (OUTPUT, HIGH = pornit)

OV2640 – XCLK	IO0	Clock extern senzor camera (20 MHz)
OV2640 – SIOD	IO26	SCCB/I2C Data – configurare senzor
OV2640 – SIOC	IO27	SCCB/I2C Clock – configurare senzor
OV2640 – Y9..Y2	IO35, IO34, IO39, IO36, IO21, IO19, IO18, IO5	Date video paralele (8 biti)
OV2640 – VSYNC	IO25	Sincronizare verticala cadru
OV2640 – HREF	IO23	Sincronizare orizontala linie
OV2640 – PCLK	IO22	Pixel clock
OV2640 – PWDN	IO32	Power down senzor

ESP32S NodeMCU

Componenta	Pin NodeMCU	Functie
DFPlayer Mini (RX)	GPIO17 (TX2)	Transmisie comenzi UART Serial2 catre DFPlayer (9600 baud)
DFPlayer Mini (TX)	GPIO16 (RX2)	Receptie status UART Serial2 de la DFPlayer (9600 baud)
Difuzor	SPK_1 / SPK_2	Iesire audio amplificata de la DFPlayer catre difuzor

Nota: DFPlayer Mini acceseaza cardul MicroSD prin SPI intern – aceasta comunicatie este gestionata complet de modul, fara pini suplimentari din NodeMCU.

Schema Electrica

Schema electrica prezinta doua sectiuni distincte:

- **Stanga – Camera (ESP32-CAM)**: Modulul ESP32-CAM conectat la programatorul FT232RL prin pinii U0T/U0R. Butonul de captura este conectat pe IO13 cu pull-down intern activat prin software. Flash LED-ul intern este controlat pe IO4.
- **Dreapta – Sound (ESP32S NodeMCU)**: Modulul NodeMCU conectat la DFPlayer Mini prin Serial2 (GPIO 16/17). DFPlayer este conectat direct la difuzorul de 3W/8Ω prin pinii SPK_1 si SPK_2. Cardul MicroSD este inserat direct in slotul DFPlayer-ului.

Diagrama Bloc

Optimizari si Calibrare

Calibrarea camerei

Senzorul OV2640 integrat in ESP32-CAM produce implicit imagini oglindite vertical, ceea ce ar face

textul capturat inutilizabil pentru motorul OCR. Calibrarea a fost realizata prin apelarea functiilor set_vflip(1) si set_hmirror(1) pe obiectul sensor_t dupa initializarea camerei. Aceasta corecteaza orientarea direct la nivelul senzorului, eliminand necesitatea procesarii suplimentare a imaginii pe server si reducand latenta totala a sistemului. Ajustarea fina finala (rotire/mirror suplimentara) a ramas in stratul de pre-procesare Pillow de pe server, pentru a acoperi si diferentele de montaj fizic al camerei.

Optimizarea accesului la fisierele audio

O problema majora intalnita a fost latenta ridicata la redarea fisierelor MP3: DFPlayer Mini citeaza lent tabela de alocare a fisierelor (FAT) de pe card, in functie de ordinea copierii fizice a fisierelor. Solutia implementata consta in utilizarea comenzii playLargeFolder(1, cod_ascii) in loc de play(track). Aceasta comanda adreseaza direct folderul numeric /01/ si asociaza imediat codul ASCII primit cu fisierul corespunzator (ex: 065.mp3 pentru 'A'), garantand identificarea instantanee a fisierului indiferent de ordinea copierii pe card.

Utilizarea memoriei PSRAM

ESP32-CAM este echipat cu 4MB de memorie PSRAM externa. Prin detectarea prezentei PSRAM (psramFound()) in faza de initializare, sistemul alege automat rezolutia UXGA (1600×1200) si calitatea JPEG 10, in loc de SVGA cu calitate 12. Aceasta optimizare mareste semnificativ acuratetea extractiei textului de catre motorul OCR, in special pentru texte mici sau cu font subtire.

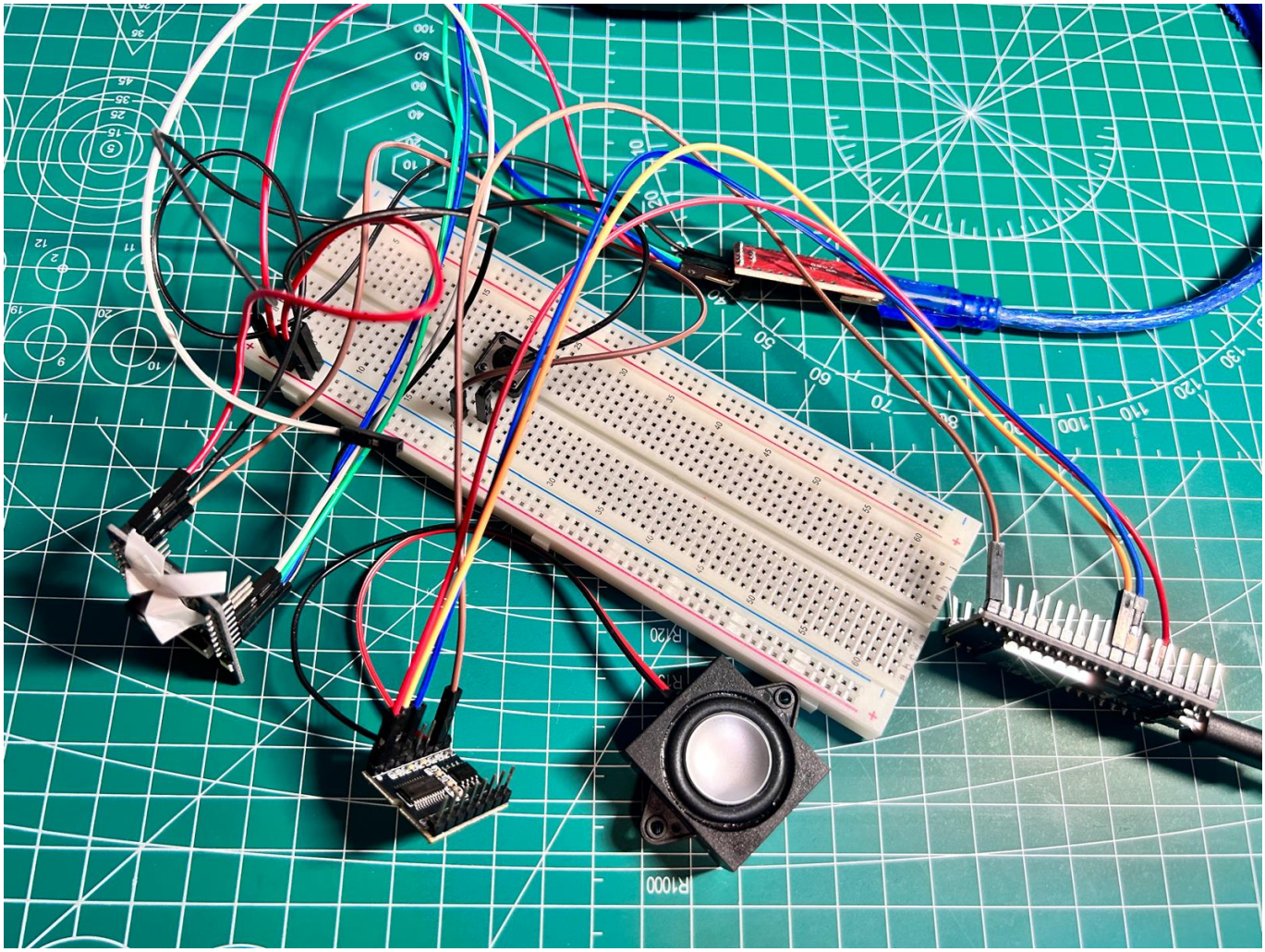
Debouncing buton

In bucla principala, apasarea butonului este validata prin doua citiri succesive cu un delay de 50 ms intre ele, eliminand semnalele false generate de vibratiile mecanice ale contactului.

Demo Video

Am realizat un scurt videoclip demonstrativ care prezinta functionalitatea completa a sistemului OptiSense. In clip se observa cum, la apasarea butonului, ESP32-CAM face captura textului “APPLE”, il trimite catre serverul Flask, care il proceseaza prin OCR API. Textul extras este descarcat wireless de NodeMCU, care il transforma in sunet redand secvential fiecare litera prin fisierele audio de pe cardul SD.

Urmareste demonstratia video pe YouTube



Jurnal de Activitate

- **26.03.2026**: Primirea componentelor fizice si documentarea tehnica initiala.
- **08.05.2026**: Definirea ideii de baza, a cazurilor de utilitate si inceperea dezvoltarii fluxului IoT.
- **09.05.2026**: Realizarea operatiunilor de lipire pentru pinii componentelor.
- **15.05.2026**: Structurarea schemelor electrice, a tabelelor cu interfete si testarea partiala (testare scanner imagine, text detectat).
- **24.05.2026**: Optimizarea si detalierea rapoartelor de conectivitate si functionalitate, plus inregistrarea si adaugarea videoclipului demonstrativ.

Descarca Proiect OptiSense ca PDF