R2-D2 Smart Home Assistant

Introducere

Proiectul își propune crearea unui **prototip de asistent pentru o casă inteligentă**, inspirat de roboțelul **R2-D2** din universul **Star Wars**. Dezvoltat pe platforma ESP32, acesta va interacționa cu utilizatorul prin comenzi vocale și redare audio, va monitoriza mediul ambiant și va dispune de funcții de securitate prin autentificare **2FA (RFID)** pentru administrarea locală.

Scopul principal este de a oferi o *soluție accesibilă și eficientă* pentru persoanele care nu își permit soluții comerciale precum Amazon Alexa sau Google Nest, ori care pur și simplu își doresc un asistent mult mai personalizabil decât opțiunile actuale de pe piață. ESP32-ul comunică prin Wi-Fi cu un server central (ideal găzduit pe un Raspberry Pi sau pe un laptop mai vechi). Acest server utilizează agenți AI pentru a procesa comenzile vocale și datele senzorilor, având totodată acces la diverse API-uri externe (de exemplu, pentru informații meteorologice). Adițional, o aplicație mobilă va permite configurarea asistentului, vizualizarea datelor înregistrate de senzori și modificarea modului curent de funcționare.

Laboratoare folosite: SPI, I2C, Intreruperi, GPIO.

Bonus: pentru audio → I2S, carcasă printată 3D, RTOS

Descriere generală

Arhitectura sistemului este formată din 3 componente logice:

- **Microcontrolerul ESP32:** Citește senzorii, procesează semnalele și trimite datele către server/dispozitivul conectat.
- **Serverul (Laptop):** Procesează datele primite prin Wi-Fi, trimite semnalul audio pentru a fi redat de microcontroler și permite modificarea stării logice (conectare aplicație) și alte configurări administrative, dar limitate.
- **Aplicație telefon/calculator (simulată de Laptop):** Permite configurarea microcontrolerului la nivel de admin, citirea datelor de pe senzori, modificarea stării logice (server) și alte configurări specifice administrării.

În cadrul proiectului, mă voi concentra asupra implementării hardware și software din cadrul nodului hardware (ESP32). Celelalte 2 componente vor fi implementate parțial sau simulate din cauza constrângerilor de timp. Doresc continuarea proiectului pe timpul verii, respectiv pe parcursul anului viitor, așadar acesta va fi primul milestone.

Logica de funcționare:

1. **Idle State:** Inelul LED respiră albastru. Modulul este conectat fie la Wi-Fi, fie la Bluetooth.
2. **Interrupt Trigger:** Apăsarea butonului fizic (sau detecția de prezență via VL53LDK) declanșează un ISR hardware.
3. **Recording State:** Cercul de LED-uri WS2812B devine roșu solid. Microfonul INMP441 înregistrează audio via I2S DMA.
4. **Processing State:** LED-urile își schimbă culoarea, așteptând răspunsul de la server.
5. **Playback State:** Inelul LED devine verde. Amplificatorul MAX98357A redă răspunsul audio generat de server.
6. **Background Tasks:** Polling pentru senzorii I2C la intervale regulate și ascultare pe interfața SPI pentru tag-uri RFID când este nevoie de 2FA sau pentru a trimite comenzi prestabilite.

3. Hardware Design

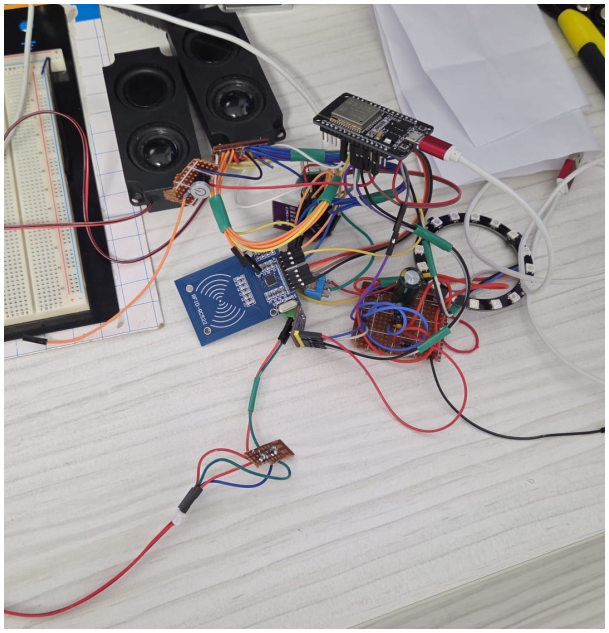
Sistemul este centrat în jurul unui modul ESP32 DevKit V1.

Componentă	Descriere	Protocol / Pinout
ESP32 DevKit V1	Microcontroller	-
INMP441	Microfon I2S	I2S IN: WS→15, SCK→14, SD→32
MAX98357A	Amplificator I2S (cu difuzor 8W)	I2S OUT: LRC→25, BCLK→26, DIN→27
BME680	Senzor T, H, P, Gaz	I2C (Adresa 0x76): SDA→21, SCL→22 CS→3.3V SDO→GND
OPT3001	Senzor de lumină ambientală	I2C (Adresa 0x44 - pin ADDR la GND): SDA→21, SCL→22
VL53LDK	Senzor ToF (Distanță / Prezență)	I2C (Adresa 0x29): SDA→21, SCL→22
MFRC522	Cititor RFID	SPI: SCK→18, MISO→19, MOSI→23, SDA→5, RST→17
WS2812B	Inel LED RGB	GPIO MUX: DIN→4
Push Button	Declanșator hardware	GPIO INT: PIN→13
RED LED Push Button	LED integrat în buton	GPIO PWM: PIN→33

Pentru a evita interferențele audio, partea de alimentare (5V) este decuplată și filtrată folosind condensatori de 1000µF, 100µF și 0.1µF, iar componentele împart un singur Common Ground Rail.

De asemenea, au fost folosite diferite rezistențe pentru a asigura integritatea electrică a componentelor: 330 ohm pentru LED-ul integrat în buton și 10 ohm pentru linia de alimentare a microfonului.

Nota 15.05.2026 - LED-ul de pe buton pare ca este ars, asa ca nu va mai fi conectat la ESP :(



Schema electrica

Componenta Software

În acest moment, implementarea software este complet funcțională, stabilă și integrată cu toate modulele hardware. Sistemul utilizează un **state machine** (automat de stări) pentru a gestiona asistentul inteligent:

- **`STATE_IDLE`**: Starea de repaus în care sistemul așteaptă declanșarea prin senzori.
- **`STATE_RECORDING`**: Înregistrarea activă a fluxului audio primit de la microfon.
- **`STATE_PROCESSING`**: Faza de simulare a procesării de rețea. Aceasta este faza în care asistentul va primi informații de la server.
- **`STATE_PLAYBACK`**: Redarea răspunsurilor. În acest moment, el redă înregistrarea vocii primită din starea de recording.

Task-urile secundare non-critice (citirea senzorilor de distanță și scanarea RFID) se realizează în background prin intermediul **FreeRTOS**. **FreeRTOS** este un sistem de operare “real-time” care are ca scop orchestrarea între procese. Deoarece citirea senzorilor prin I2C durează destul de mult, sistemul beneficiază de acest scheduling inteligent. Un video care a inspirat această abordare poate fi vizionat aici: https://www.youtube.com/watch?v=i_eU16X67qU.

Biblioteci folosite Pentru realizarea funcționalităților au fost integrate următoarele biblioteci:

- **`driver/i2s.h` (ESP-IDF API)**: Am preferat utilizarea API-ului nativ pus la dispoziție de Espressif în detrimentul bibliotecilor standard Arduino I2S. Această alegere oferă control deplin asupra bufferelor DMA și a configurării canalelor, element critic pentru microfonul INMP441, care necesită maparea specifică pe canalul drept (`I2S_CHANNEL_FMT_ONLY_RIGHT`) din cauza unui quirk de design hardware (a se vedea pagina 8 din documentația microfonului, în tabel la linia “L/R”).
- **`FastLED.h`**: Aleasă datorită eficienței, fiind o bibliotecă comună și extrem de stabilă. Permite actualizarea culorilor inelului RGB fără a introduce întârzieri blocante.
- **`MFRC522.h`**: Biblioteca standardizată pentru interfațarea prin protocolul SPI cu modulul de citire a

tag-urilor RFID.
- **Wire.h**: Bibliotecă utilă pentru lucrul cu senzorul OPT3001, care nu beneficiază de o bibliotecă proprie standard.
- **Adafruit_BME680.h**: Biblioteca oficială pentru utilizarea senzorului de mediu BME680.
- **Adafruit_VL53L0X.h**: Biblioteca oficială pentru utilizarea senzorului de distanță (ToF) VL53L0X.
- **WiFi.h**: Bibliotecă folosită pentru conexiunea la o rețea Wi-Fi. Momentan a fost implementat doar init-ul (cod boilerplate), integrarea cu serverul urmând să fie adăugată ulterior.

Elemente de noutate și funcționalități din laborator Noutatea proiectului constă în crearea unui ecosistem complet de asistență care folosește doi declanșatori (autentificare RFID și senzor de apropiere) și oferă un feedback dual: vizual complex (inel RGB cu coduri de stare) și audio bidirecțional. O altă noutate este printarea integrală 3D a carcasei; robotelul R2-D2 a fost creat în integralitate și montat custom pentru aplicația noastră. De asemenea, utilizarea FreeRTOS reprezintă un element tehnic de noutate, orchestrând eficient task-urile direct pe microcontroler.

La nivel de software, proiectul se distinge prin realizarea unui **sistem intern de înregistrare-redare "in-memory"** adaptat pentru arhitectura ESP32. Deși microcontrolerul are un spațiu limitat de memorie RAM, sistemul reușește să stocheze și să proceseze blocuri de eșantioane audio de lungă durată (4 secunde) printr-un procedeu custom de downsampling și upsampling în timp real, eliminând necesitatea unui card SD extern. Acest proces este realizat în prezent pentru demonstrație, scopul final fiind trimiterea înregistrării prin rețea spre a fi procesată de server.

Proiectul integrează cu succes concepte studiate în cadrul laboratorului de PM:

- **Întreruperi Hardware**: Butonul de reset folosește o rutină de întrerupere mapată direct în memoria RAM rapidă (`IRAM_ATTR`). Acest lucru garantează oprirea instantanee a mașinii de stări din orice fază și resetarea asistentului la starea inițială.
- **SPI (Serial Peripheral Interface)**: Utilizat pentru comunicarea de mare viteză cu cititorul RFID MFRC522.
- **I2C**: Folosit pentru comunicarea cu senzorii I2C.

Arhitectura aplicației și interacțiunea modulelor Aplicația este modularizată pe fișiere cu responsabilități clare:

- **ui.cpp / ui.h**: Se ocupă de feedback-ul vizual (LED-uri, WS2812B) și de întreruperea butonului.
- **audioIn.cpp / audioIn.h**: Gestionează achiziția audio și calculul metricilor de semnal (RMS).
- **audioOut.cpp / audioOut.h**: Controlează redarea bufferelor audio pe amplificator.
- **rfid.cpp / rfid.h & sensors.cpp / sensors.h**: Module dedicate achiziției de date de la periferice.
- **main.cpp**: Conține bucla principală (loop-ul) și mașina de stări.
- **network.cpp / network.h**: Conține funcționalitatea de bază pentru Wi-Fi și Bluetooth.

Un ciclu complet de funcționare decurge astfel:

1. În starea **IDLE**, inelul LED are culoarea albastră.
2. Dacă se percepe un semnal de la senzorul de distanță (distanța < 50 mm, deci R2-D2 este atins) se schimbă starea în **RECORDING**. LED-urile devin roșii și începe înregistrarea. Alternativ, scanarea unui tag RFID validează prezența utilizatorului, redă un sunet scurt de confirmare și colorează inelul în gri.
3. După înregistrare, sistemul intră în starea **PROCESSING**, iar inelul devine galben.
4. În final, în starea **PLAYBACK**, inelul devine verde și se redă răspunsul audio, urmat de reproducerea vocii înregistrate anterior.

Validarea funcționării s-a efectuat prin corelarea datelor transmise pe interfața UART (monitorul Serial rulat la 115200 baud) cu modificările fizice observate. Mesajele de tipul `[State] Recording…` sau `[RFID] UID: XX XX XX XX` au asigurat trasabilitatea și corectitudinea tranzițiilor.

Calibrare și optimizări software Pentru microfonul INMP441, datele brute citite prin I2S vin pe un format aliniat de 32 de biți (dar semnalul util ocupă doar 24 de biți). Pentru calibrare, s-a efectuat o deplasare bit cu bit la dreapta cu 16 poziții pentru a aduce semnalul la un format standard pe 16 biți PCM (semnat). Ulterior, s-a implementat o funcție matematică de calcul a valorii medii pătratice (**RMS - Root Mean Square**) și transformarea acesteia în decibeli (`rmsDb = 20.0f * log10f(…)`), stabilindu-se pragul de zgomot ambiental al camerei. De asemenea, senzorul de proximitate ToF a fost calibrat software în interiorul funcției de polling pentru a asigura declanșarea asistentului doar când utilizatorul se află la o distanță de sub 50 mm.

O optimizare critică a fost necesară pentru gestionarea memoriei în timpul înregistrării. Deoarece asistentul înregistrează continuu timp de 4 secunde, o rată nativă de 16 kHz cu eșantioane pe 16 biți ar fi necesitat un buffer masiv de **128 KB** de RAM, resursă de care ESP32 Dev Kit nu dispune liber. Soluția aplicată a fost realizarea unui **downsampling**: deși hardware-ul citește la 16 kHz, software-ul extrage doar primul eșantion din două (efectiv 8 kHz), reducând la jumătate memoria utilizată. Pentru faza de playback, fluxul suferă un **upsampling**, fiecare eșantion fiind duplicat pentru a reconstrui semnalul necesar amplificatorului.

Cod: https://github.com/Mars-Zero/AI-Home-Assistant

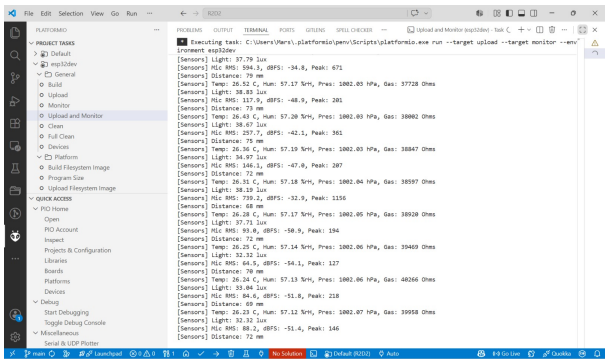
Rezultate Obţinute

Demo: https://youtube.com/shorts/jC4Zk_jOYps?feature=share

Concluzii

Chiar daca nu este chiar ce voiam, am ajuns destul de departe cu implementarea. State machine-ul

este complet, functiile de conectare la WI-FI+bluetooth fiind simulate, impreuna cu functia de verificare a administratorului



Jurnal

- 11.03.2026 - Initial commit
- 7.04.2026 - Schema electrica aproape terminata
- 30.04.2026 - Inceput partea software pentru testare componente
- 5.05.2026 - Schema electrica modificata pentru senzorul BME680(verificare datasheet)
- 10.05.2026 - Printat 3D corpul R2-D2
- 12.05.2026 - Inceput implementare hardware pe placuta de prototipare + scris niste cod boilerplate i2c pentru verificare
- 15.05.2026 - Terminat de lipit si testat pe placuta de prototipare prototipare
- 16.05.2026 - Plecat la Dedeman pentru niste piulite pentru a prinde corpul de picioarele robotului.
- 17.05.2026 - Asamblat robotul + “translplantul de creier + adaugat codul state machine si implememental modularizat codul
- 19.05.2026 - Rafinat comportamentul robotului.

Bibliografie/Resurse

https://www.espboards.dev/esp32/esp32doit-devkit-v1/

https://www.farnell.com/datasheets/1824785.pdf

https://www.analog.com/media/en/technical-documentation/data-sheets/max98357a-max98357b.pdf

https://www.bosch-sensortec.com/en/products/environmental-sensors/gas-sensors/bme680#technical

https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1777933152891

https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html#documentation

https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

https://cdn.sparkfun.com/datasheets/Components/LED/WS2812.pdf

https://makerworld.com/en/models/942089-r2d2-echo-dot-3-dock#profileId-908012

Export to PDF