Sesiunea de comunicări științifice studențești, ediția 2023
Sisteme software
Universitatea Politehnica din București

Moduri noi de utilizare ale microcontrolerelor folosind API-uri de modele de limbaj

Gul Farhad Ali Irinel, Niculae Andrei
farhadgul2001@gmail.com, andrei.niculae101@gmail.com
Coordonator: Conf. Dr. Ing. Dan Ștefan Tudose
dan.tudose@upb.ro

Abstract În această lucrare prezentăm câteva cazuri de utilizare a microcontrolerelor pentru a interacționa cu API-uri de modele de limbaj, cum ar fi ChatGPT, explorând modul în care putem integra mai bine aceste instrumente în fluxul nostru de lucru într-un mod flexibil și relativ necostisitor.

1. Introducere

Apariția modelelor largi de limbaj (Large Language Models - LLMs) a adus cu aceasta noi oportunități de a folosi microcontrolere necostisitoare, care au capabilitatea de a se conecta la internet (precum RP2040, ESP32), sau module WiFi precum cele bazate pe ESP8266, pentru a realiza comunicarea între un om și un chatbot AI, fără a mai avea nevoie de un sistem de calcul puternic care sa ruleze modelul local.

În cadrul proiectului vom descrie implementarea hardware, software si funcționalitățile a două dispozitive care interacționează cu un astfel de model AI: o boxă inteligentă și un dispozitiv plug & play, care te lasă să folosești modelul în orice aplicație și pe orice PC.

2. Motivația, utilitatea și scopul proiectului

În prezent, multe platforme prezente cu roluri diverse dispun de o interfață de comunicare, o interfață ce oferă transfer de date într-un mod mult mai simplu, fără a fi nevoie de integrare grafică specifică. Aceasta interfață se cheama API (Application Programming Interface).

Rolul unui API este să expună funcționalitatea unui sistem într-un mod organizat.

Platforma care oferă informațiile trebuie să descrie modul în care se poate comunica cu ea prin API. Comunicare cu API-urile alese în cadrul dispozitivelor se face la nivel de HTTP Requests. [1]

Acum că avem o metodă de a extrage informații de la diverse surse, trebuie să ne orientăm analiza spre ce platformă am ales să folosim. Puterea computațională de care este nevoie pentru a procesa un HTTP request este suficient de mică încât acest proces să poată fi realizat de un microcontroler.

Pentru acest proces, am ales să folosim un Raspberry Pi Pico W, deoarece aveam nevoie de un microcontroler care să poată utiliza informațiile obținute într-un mod practic.

Proiectul se bazează pe comunicarea cu mai multe API-uri, printre care și cel de bază - API-ul pus la dispoziție de OpenAI pentru a comunica cu ChatGPT. În momentul de față, este cea mai bună metodă de a comunica cu un AI dintr-un dispozitiv remote cu putere de procesare limitată.

În cadrul proiectului, am utilizat și alte API-uri pe care le vom enumera la aplicații și integrare software, pentru ca acesta sa aibă mai multe facilități oferite. Informațiile obținute au fost corelate cu ChatGPT pentru o prelucrare a datelor. Spre exemplu, proiectul dispune de o funcție meteo pentru vremea curentă sau pentru cea din următoarele zile în funcție de orașul dat. Alta funcție pe care o are este redarea vocală pentru un anumit text.

Un Pico nu are putere mare de procesare, iar din această cauză, am ales să folosesc mai multe API-uri pentru procesele computațional intensive precum procesarea pieselor muzicale descărcate de pe youtube pentru a putea fi redate de pe Pico. Piesele sunt procesate remote iar Pico descarcă piesa direct în formatul audio compatibil.

Având în vedere multitudinea de API care sunt puse la dispoziție, dar și o unitate de procesare accesibilă (Raspberry Pi Pico W), se pot face multe prelucrări de informații și idei interesante, doar cu o conexiune la internet.

Noi am ales să prezentăm 2 utilizări diferite care să pună accentul pe comunicarea cu ChatGPT:

1. Un dispozitiv portabil care preia comenzi din telefon, prin bluetooth și produce răspunsul vocal în 2 limbi posibile (engleză sau română, în funcție de modul selectat). Dispune de funcționalități precum: play la piese de pe youtube, vremea din orice oraș în momentul actual sau din următoarele zile, statistici locale de temperatură și umiditate. De asemenea, poate face request-uri la ChatGPT, să primească răspunsul și să-l transmită prin boxe.
2. Un dispozitiv care se conectează la PC prin USB, care poate asculta și decoda pachetele USB trimise de o tastatură și care poate face API request-uri către ChatGPT, iar răspunsul primit îl tastează la PC, ca și cum ar fi tastat de o tastatură normală. De asemenea, el poate funcționa și în mod standalone, conectând doar tastatura, fiind alimentat de un acumulator, iar procesul este același, însă răspunsul primit îl afișează pe un display TFT.

Vom descrie funcționalitățile și întregul proces pentru ambele tipuri de dispozitive în cele ce urmează.

Ținem să precizăm că dispozitivele nu sunt în fază finală, fiind prototipuri funcționale care să ne susțină motivația și scopul prezentării.

În momentul de față, în funcție de experimentele realizate pe parcursul proiectului, putem afirma că bottleneck-ul procesării unei comenzi la nivelul folosite nu stă în capabilitatea hardware a microcontrolerului ci mai degrabă în platforma software care rulează pe acesta. Bibliotecile folosite pentru procesare audio (procesarea răspunsului de la API de text-to-speech) nu dispun la bază de redare audio în flux. Trebuie făcute anumite operații cu buffere care nu sunt suportate pe Pico. De aceea, scopul acestui proiect este acela de a îndrepta atenția și spre această zonă de proiectare cu microcontrolere, zona de proiectare folosind Cloud, care are la bază API-uri.

3. Componentele hardware

Vom analiza componentele hardware folosite pentru ambele dispozitive.

a. GPT Multimedia Assistant [18]:

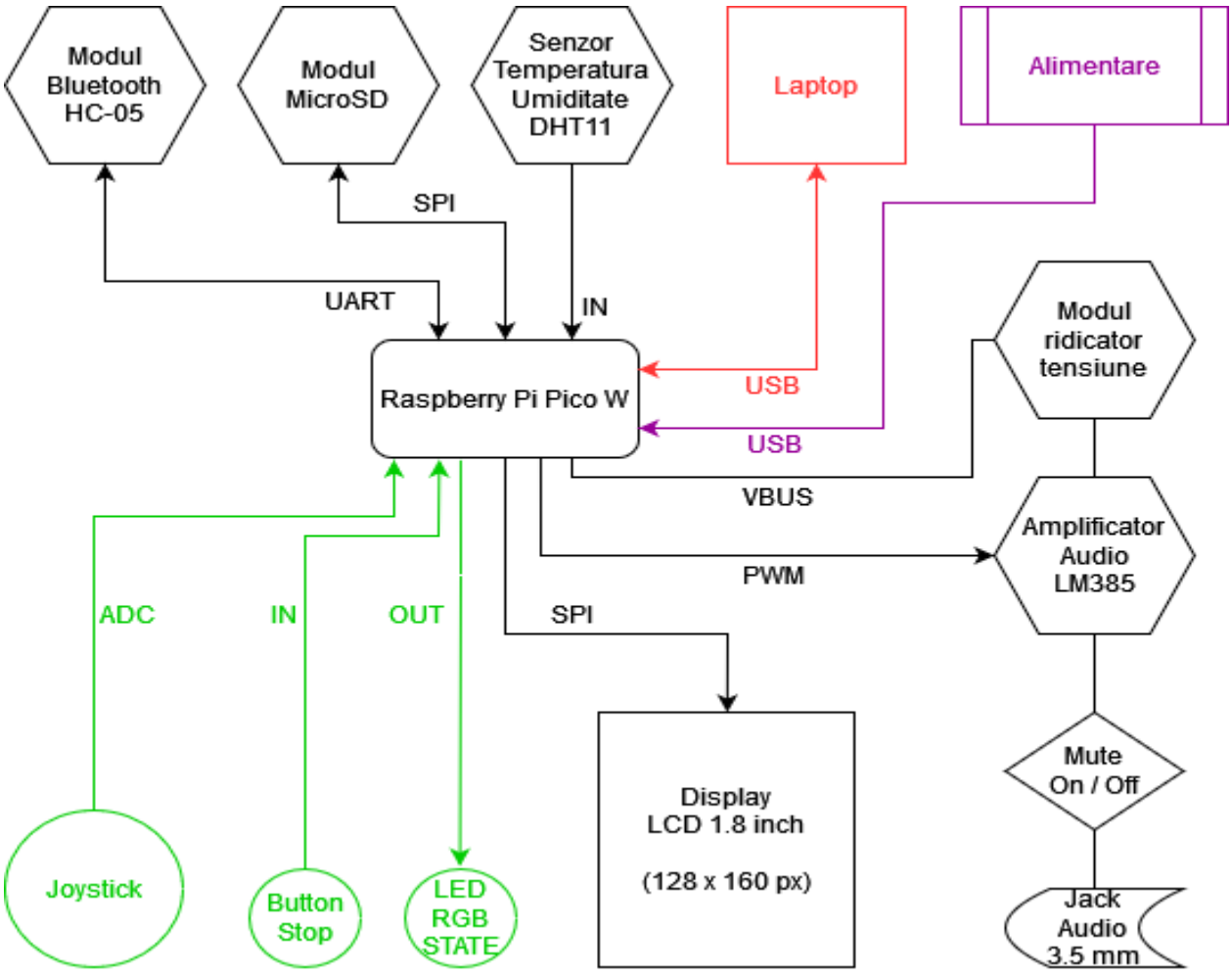

Fig. 1.1 Schema bloc

Am ales să diferențiez anumite aspecte prin culori. Spre exemplu, culorile roșu și mov indică faptul că aceste legături se realizează doar pe un singur port microUSB ceea ce duce la posibilitatea conectării la laptop sau la alimentator (baterie externă) neavând oricum sens să fie conectate simultan.

Partea indicată cu verde este în principal partea de I/O, cu precizarea că este încă în lucru.

Componentele hardware utilizate sunt:

- Modul Bluetooth HC-05: utilizat pentru a transmite comenzile de la telefon la Pico, legat prin UART [2]
- Modul MicroSD: utilizat ca buffer de memorie pentru piese - aici sunt descărcate, salvate și citite fișierele mp3 ulterior, legat prin SPI [3]
- Senzor temperatură DHT11: folosit pentru a prelua temperatura și umiditatea din încăperea curentă
- Display LCD: utilizat pentru a afișa text și pentru eventuale activități
- Modul ridicator tensiune: folosit pentru a oferi o tensiune mai mare (12V) amplificatorului audio pentru o mai bună redare și acurateță
- Amplificator audio LM385: folosit pentru a amplifica semnalul PWM trimis de către Pico la boxă
- Jack Audio 3.5mm: utilizat pentru a oferi o interfață audio cu exteriorul, conectare la o boxă portabilă
- LED RGB State: indică statusul actual al dispozitivului, RED - ocupat să prelucreze comanda, GREEN - disponibil și gata să primească comenzi, BLUE - într-o activitate
- Mute ON / OFF: switch de mute, poate bloca output-ul pentru a nu reda sunetul
- Buton STOP: poate opri redarea fișierului mp3 curent

b. Plug & Play ChatGPT [19]:

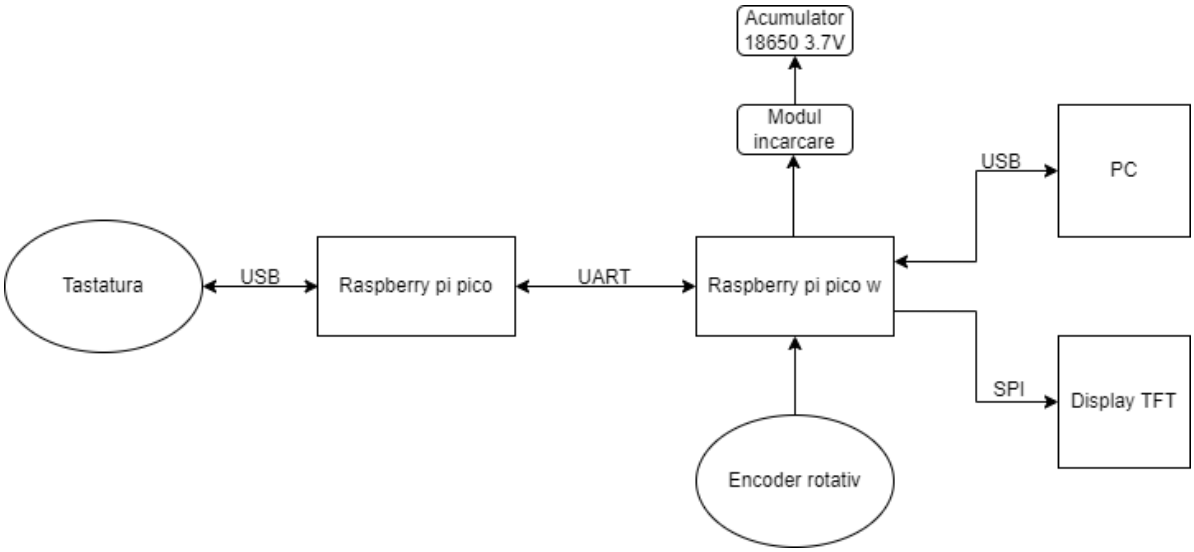

Fig. 2.1 Schema bloc - mod middleman

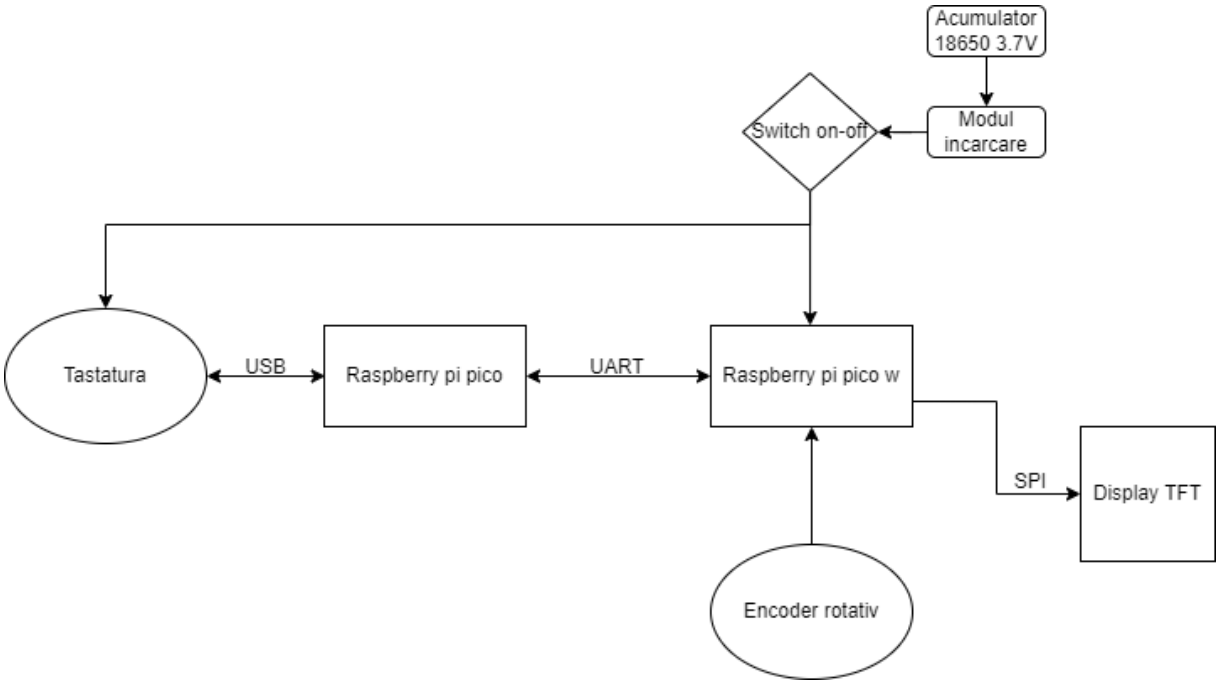

Fig. 2.2 Schema bloc - mod standalone

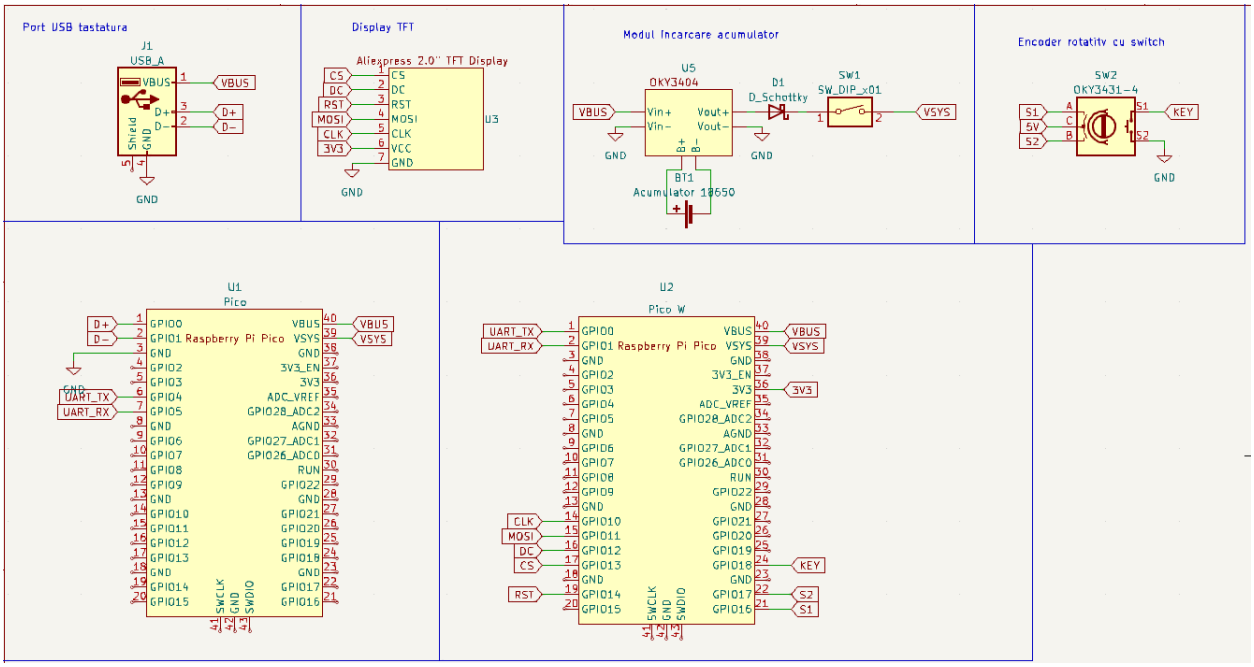

Fig. 2.3 Schema electrica

În cadrul proiectului am folosit următoarele componente:

- Raspberry pi pico: care rulează un program care folosește biblioteca TinyUSB pentru a recepționa pachetele USB de la tastatura
- Raspberry pi pico w: care rulează un program scris in CircuitPython, decodează pachetele primite de la primul Pico prin UART, face API requests, afișează răspunsul pe display sau îl tasteaza la PC
- Display TFT 2.0": utilizat pentru afișarea răspunsurilor sau a diferitelor comenzi dintr-un meniu.
- Encoder rotativ cu buton OKY3431-4: folosit pentru selectarea opțiunilor din meniu.
- Modul încărcare acumulator OKY3404: încarcă acumulatorul atunci când dispozitivul este conectat la PC, sau alimentează circuitul, folosind acumulatorul, in mod standalone.
- Acumulator 18650: cu capacitatea de 3800mAh, 3.7V
- Suport acumulator
- Modul port USB: prin care se poate conecta o tastatura la dispozitiv

- 1 x dioda Schottky: plasată după modulul de încărcare, pentru a nu lăsa tensiunea de 5V de pe VSYS sa ajungă la pinii de out de pe modul și să distrugă acumulatorul, în cazul în care dispozitivul este conectat la PC.
- Fire de legătura
- Cablaj de test

4. Conectarea la API

Acum că dispunem de hardware-ul necesar, trebuie să scriem codul care să facă tot procesul posibil.

Am ales să realizăm aceste dispozitive pe CircuitPython, fiind o platformă bazată pe python ce dispune de suport integral pentru ce urmează să prezentăm la aplicații și integrare software.

Pentru a realiza un API Request avem în primul rând nevoie de conexiune la internet. După conectarea Pico la internet prin WIFI, avem nevoie de o bibliotecă care să poată face request-uri. Am ales biblioteca adafruit_requests, capabilă de requesturi simple. [4]

Tot codul este complex și lung, vom pune accentul pe baza acestuia și anume requestul la ChatGPT. Pentru a face acest request, avem nevoie de preconfigurare de socket, ssl, pentru ca mai apoi să deschidem o sesiune de request.

Formatul prompt-ului trimis este de forma: full_prompt = [{"role": "user", "content": payload},], payload fiind textul primit ca input. În cadrul request-ului, trebuie precizat ce model vrem să folosim, noi am folosit gpt-3.5-turbo. De asemenea, am precizat și authorization key-ul, cel obținut de pe pagina celor de la OpenAI. [5]

Pentru a trimite request-ul, am folosit requests.post cu parametrii: URL, modelul dorit și mesajul ca JSON și Authorization Key ca header.

Răspunsul primit poate fi extras sub formă de JSON din response.content.

Fiecare API poate avea un model de împachetare a datelor diferit, la fel și tipul de cerere care trebuie făcută pentru a afla informațiile. Modelul de request de mai sus se aplică mai departe și pentru alte API-uri pe care le-am folosit în proiect cu modificările necesare aferente.

5. Aplicații și integrare software

a. GPT Multimedia Assistant:

Un caz special de request pe care vreau să-l tratez este cel către API-ul de text-to-speech, cel care dă voce și viață dispozitivului 1. Cererea se face asemănător, doar că în loc de post am folosit requests.get.

Din cauza memoriei RAM destul de mici pe care o are un Pico, răspunsul trebuie parcurs pe iterații, neputând procesa tot răspunsul deodată, fiind vorba de un fișier audio. Astfel, iterez în blocuri de 2048 și salvez rezultatul pe cardul SD. Aici intervine limitarea platformei de redare sunet, care nu poate procesa audio-ul pe bucăți, încercasem chiar și cu fișier wav, din cauza buffer-ului intern audio, nefolosind eficient memoria RAM de care dispune Pico.

Procesul de îmbinare ale acestor API-uri le voi prezenta în secțiunea de funcționalități.

Întrucât pot apărea probleme de memorie ocupată din cauza lucrului cu fișiere mari, ca metodă de siguranță și fiabilitate, la fiecare eroare pe care o întâlnește, dispozitivul repornește. Pentru cazurile de bază această situație nu este prezentă, dar în cazuri speciale poate apărea și de aceea este importantă o metodă care să asigure viața prelungită a dispozitivului.

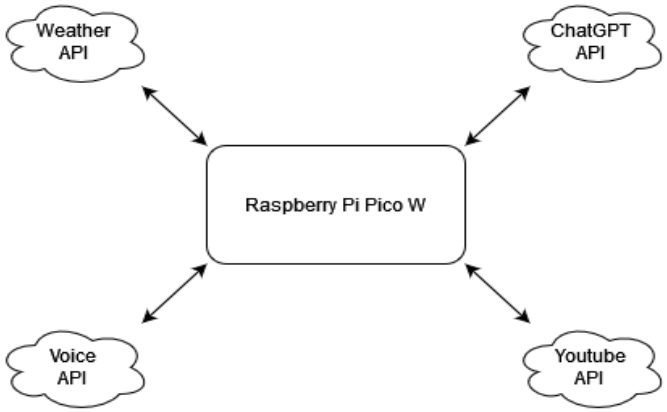

Fig. 1.2 Comunicarea în cloud

API-uri folosite:

- ChatGPT
- Vreme [6]
- Text-to-speech [7]

- Youtube API [8]

Codul scris pentru acest dispozitiv va fi disponibil la finalul semestrului. [17]

b. Plug & Play ChatGPT:

Prima problemă care trebuie rezolvată este citirea pachetelor USB care sunt generate atunci când se apasă pe taste. Acest lucru nu este exact trivial, întrucât există diferite convenții folosite de diferite tastaturi pentru comunicarea prin USB. Există însă biblioteca TinyUSB [10], și implementarea sa pentru microcontrolerul RP2040 [11], care abstractizează din complexitatea protocolului. Astfel, se poate folosi capabilitatea de PIO (programmable I/O) a microcontrolerului pentru a adăuga și citi interfete USB suplimentare. Implementarea aceasta este scrisa in C, însă majoritatea bibliotecilor destinate microcontrolerului sunt scrise folosind limbajul CircuitPython, asa că am modificat exemplul *capture_hid_report.c,* pentru a trimite rapoartele USB printr-o conexiune UART catre un Raspberry pi pico w, programat folosind CircuitPython.

Următorul pas este parsarea acestor pachete. Formatul pachetelor depinde de asemenea de tipul tastaturii și a functionalitatilor sale (5-key rollover vs n-key rollover, diferite taste cu roluri macro/mouse etc.). Implementarea din proiect ia în calcul cazul cel mai simplu: 1 byte care reprezinta tastele modifier (Shift, Alt, Win, Ctrl), și un range de bytes care reprezinta tastele apăsate (care sunt decodați folosind acest tabel [12]). Acești parametrii sunt specificați în cadrul funcției care parsează pachetele.

Acum că putem citi un text de la tastatură, putem face un API request si recepționa răspunsul așa cum a fost descris mai sus. Pentru a putea tasta răspunsul la un PC, există biblioteca *adafruit_hid,* prin care putem face microcontrolerul să se comporte ca o tastatură. Răspunsul este tastat pe măsura ce este primit de catre microcontroler.

Dorim de asemenea sa afișăm textul recepționat pe un display TFT, pentru acesta existând bibliotecile *displayio* (pentru interfațarea prin SPI)*, adafruit_ili9341* (pentru câteva setări specifice display-ului ales)*, terminalio* (pentru a afisa text)*.*

În final, pentru a interfața cu display-ul, pentru a selecta diferite template-uri (despre care voi vorbi in sectiunea funcționalități), vom folosi un encoder rotativ cu buton, care este controlat prin biblioteca *rotaryio.*

Codul pentru cele doua microcontrolere va fi disponibil la acest Github [13], după ce vor fi implementate toate funcționalitățile.

6. Funcționalități

În momentul de față, am descris motivația, platforma hardware și cea software pentru a putea face posibil tot procesul. Acum urmează să vedem cum funcționează, cum un utilizator s-ar putea folosi de dispozitivele realizate de noi.

a. GPT Multimedia Assistant:

După cum am descris anterior, acest dispozitiv are rolul de a oferi mai multe facilități utilizatorilor, neavând nevoie de componente cu putere mare de procesare pentru a face diverse activități. Pentru acest dispozitiv, până în prezent, am ales să implementez următoarele:

- Vremea în diverse orașe, în prezent sau în viitorul apropiat. Rezultatul este sub formă de json, care este parsat la ChatGPT iar acesta compune un text cu datele obținute
- Discutat cu ChatGPT, compus povești cu input-ul utilizatorului ca punct de start și cam orice se poate face cu ChatGPT prin GUI.
- Toate răspunsurile primite de la ChatGPT vor fi transpuse în mp3 audio și redate prin boxă
- Pot fi redate piese muzicale, cardul microSD fiind folosit ca zonă buffer unde se stochează piesa descărcată. Fișierul mp3 este prelucrat de către API-ul de youtube pentru a putea fi redat de Pico fără probleme (mono, 44.1kHz ca sample rate). Mai departe, piesa este stocată pe cardul SD și redată. Următoarea dată când va fi cerută aceeași piesă, aceasta nu va mai fi descărcată, ci va fi direct redată direct din cardul SD.

Utilizatorul se conectează prin bluetooth la modulul HC-05. Acesta va folosi o aplicație de comunicare pe care o poate găsi pe magazin play. Un exemplu de aplicație compatibilă ar fi Arduino Bluetooth Control [9].

Conectarea se face prin terminalul aplicației. Am ales să prefixez comenzile cu „!”. Utilizatorul poate să execute următoarele comenzi în terminal:

- !help: afișează comenzile disponibile
- !ping: se poate verifica conexiunea între telefon și Pico. Se va afișa un „Pong!” pe ecran.

- !restart / !reload: se rulează de la început fișierul code.py
- !p <payload>: se poate reda piesa cu titlul specificat în <payload>
- !tell <payload> / !say <payload>: dispozitivul va transmite audio inputul primit
- !mem: memoria RAM disponibilă
- !wf <payload>: vremea în viitorul apropiat din orașul dat ca input, dacă <payload> este gol, atunci se va analiza vremea în viitorul apropiat din București
- !w <payload>: vremea în acest moment din orașul dat ca input, dacă <payload> este gol, atunci se va analiza vremea în acest moment din București
- !last: se va reda ultimul răspuns primit de la ChatGPT
- !lan ro / en: se va seta limba pe ro / en, în funcție de cum este precizat. Limba este utilizată și pentru API-ul text-to-speech, selectându-se vocea potrivită limbii selectate
- !stats: statistici de temperatură și umiditate din încăpere
- <payload>: scriind direct textul, neprefixat de „!”, acesta va fi prelucrat de ChatGPT și se va oferi un răspuns

Spre exemplu, pentru o cerere de vreme, se trimite un API Request către API-ul de vreme, se așteaptă răspunsul. După ce a fost primit, se trimite json-ul mai departe la ChatGPT pentru a fi prelucrat într-un text cursiv. La final, răspunsul primit de la ChatGPT este trimis la API-ul de text-to-speech care prelucrează textul și trimite răspuns fișierul binar mp3 care este salvat pe cardul SD și redat ulterior. Acest caz îl vom analiza la concluzia lucrării.

b. Plug & Play ChatGPT:

Funcționalitatea de bază a proiectului a fost de a avea un dispozitiv plug & play, care poate fi folosit pentru a accesa funcționalitățile unui API precum ChatGPT, pe orice PC și in orice aplicație de pe acesta - de exemplu, pentru a face refactoring/documentare pe o bucată de cod, pentru a genera cod boilerplate, pentru a corecta/reformula o secțiune dintr-un text, pentru a scrie un template de e-mail, traduceri de text, sau pur si simplu pentru a afla o informație nouă, într-un mod mai rapid decât dacă am deschide browser-ul. Acest lucru se face prin apăsarea unui shortcut predefinit, tastarea interogării care sa fie trimisă la API și apoi tastarea răspunsului de către microcontroler. Atunci când nu se ascultă interogarea, controlerul pur și simplu redirecționează pachetele USB către PC. In modul simplu, microcontrolerul poate primi doar date de la tastatura, însă putem extinde funcționalitățile daca rulăm un script batch pe PC, care ne lasă sa trimitem un text selectat pe calculator către microcontroler, prin monitorul serial.

Folosind display-ului TFT, pentru a selecta diferite prompt-uri, care să fie inserate înaintea textului scris de la tastatura, pentru a comenzi folosite des, spre exemplu: *"Translate this text to english:"*, *"Refactor this code, don't write any other comments: ", "Correct any mistakes you find in this text:".*

Adaugarea display-ului a dus la crearea modului standalone: dispozitivul să poată fi folosit și în cazul în care nu există acces la un calculator, iar din câteva teste, am observat că acesta are o autonomie de ~3 ore.

7. Încheiere și idei de îmbunătățire

Întrucât în exemplul oferit la funcționalitatea dispozitivului 1 sunt mulți pași de realizat, un audio binar nu poate fi redat pe blocuri și nu există suficient RAM pentru a prelucra tot fișierul mp3 deodată, aceste operațiuni pot să consume niște timp.

Pe parcursul lucrării, am vorbit doar despre folosirea API-ului de la OpenAI, principalul motiv fiind ca gpt-3.5-*turbo* este unul dintre cele mai capabile modele, raportat la prețul de folosire (0.002$/1k tokeni), la momentul scrierii. Trebuie să ținem cont însă că apar constant noi modele (precum Llama [14], LAION [15], Bard[16]), care pot fi mai capabile decât cel de la OpenAI.

Cu timpul, dacă atenția este îndreptată spre această zonă de dezvoltare și proiectare cu microcontrolere, putem să eficientizăm aceste operații, care la prima vedere par destul de complexe.

Până recent, nu ne-am fi gândit la o astfel de întrebuințare pentru un Pico, de aici și scopul principal al lucrării, să arătăm că se poate, este posibil. O idee de eficientizare pentru procesul descris ar fi realizarea unor biblioteci pe Pico care să eficientizeze operații precum prelucrarea paralelă a operației de primire de răspuns și de trimitere a acestuia mai departe, în următoarea fază de procesare.

Aceste dispozitive prezentate în lucrare pot servi la funcționalități ca cele descrise și nu numai. Se pot realiza o multitudine de proiecte în această zonă. Prin intermediul API-urilor, putem realiza lucruri care la prima vedere par dificile și complexe prin o serie de request-uri. Cel mai important step-up este apariția ChatGPT și disponibilizarea unui API de către aceștia care poate să ofere unui Pico putere mărginită doar de imaginația programatorului.

8. Referințe

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[2] https://www.circuitbasics.com/basics-uart-communication/

[3] https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[4] https://docs.circuitpython.org/projects/requests/en/latest/api.html

[5] https://platform.openai.com/account/api-keys

[6] https://openweathermap.org/api

[7] https://rapidapi.com/voicerss/api/text-to-speech-1

[8] https://github.com/FarhadGUL06/YoutubeAPI

[9] https://play.google.com/store/apps/details?id=com.broxcode.arduinobluetoothfree

[10] https://github.com/hathach/tinyusb

[11] https://github.com/sekigon-gonnoc/Pico-PIO-USB

[12] https://github.com/hathach/tinyusb/blob/5e023fa2ca4c20f06b6e0dc12f6f044a7d4e14bd/src/class/hid/hid.h#LL1016C19-L1016C19

[13] https://github.com/nan-dre/plug-n-play-chatgpt

[14] https://research.facebook.com/publications/llama-open-and-efficient-foundation-language-models/

[15] https://laion.ai/

[16] https://ai.google/static/documents/google-about-bard.pdf

[17] https://github.com/FarhadGUL06/PicoAssistant

[18] https://ocw.cs.pub.ro/courses/pm/prj2023/danield/gptma

[19] https://ocw.cs.pub.ro/courses/pm/prj2023/ncaroi/plug