به نام خدا

گزارش نـهایی

«ساخت رابط دستیار صوتی مبتنی بر LLMها برای HMI شرکت Siemens»

ESP32 AI Home Assistant

نام استاد:

دکتر اجلالی

گردآورندگان:

حمید رضا علیپور

عارفه بوشهریان

سینا مشتاقیون

دانشگاه صنعتی شریف

تابستان 1403

فهرست مطالب

مقدمه

هدف این پروژه ساخت یک دستیار صوتی هوشمند است تا بتواند دستورات کاربر در ارتباط با تابلو برق را متوجه شود و ضمن پاسخ دادن با کاربر به صورت صوتی آن دستورات را اجرا کند و بدین ترتیب کاربر مجبور نباشد که نحوه کار کردن با تابلو برق را بلد باشد.

در این پروژه ابتدا قرار بود که از HMI شرکت siemens به عنوان نمایشگر استفاده شود، ولی به دلیل مهیا نبودن آن در آزمایشگاه از شبیه‌ساز tia portal استفاده کردیم تا plc را برای ما شبیه‌سازی کند و بتوانیم در آن tag ها را تغییر دهیم و برای مثال یک چراغ را روشن و خاموش کنیم.

بخش مهم دیگر این پروژه میکروکنترلر ESP32 بود که به عنوان بخش مرکزی و رابط عمل می‌کرد. کاربر به صورت صوتی درخواست خود را از طریق میکروفون به ESP32 می‌رساند تا ESP32 این درخواست را از طریق شبکه به سرور که یک کامپیوتر باشد بدهد. در این قسمت سرور درخواست صوتی را به متن تبدیل می‌کند و آن را برای ChatGPT می‌فرستد و جواب مناسب را از آن دریافت می‌کند و پس از تبدیل آن جواب به یک فایل صوتی آن را پخش می‌کند.

نرم افزار tia portal بر روی یک کامپیوتر دیگر در حال اجرا است. در صورت درست و قابل اجرا بودن درخواست کاربر، سرور از طریق شبکه عملیات صحیح را برای tia portal ارسال می‌کند تا بدین ترتیب درخواست کاربر اجرا شود.

ابزارهای مورد نیاز

نیازمندی‌های سخت‌افزاری:

- میکروکنترلر ESP32
- میکروفون Inmp441 یا هر میکروفون I2S دیگر
- BreadBoard
- سیم

نیازمندی‌های نرم‌افزاری:

- ESP32 Aurduino core
- شبیه‌ساز tia portal
- PLCSIM

مراحل انجام پروژه

در ابتدا تلاش کردیم کار با tia portal و نحوه عوض کردن tag ها و کار با آن‌ها و برنامه نویسی Ladder را بیاموزیم. بخش مهم دیگر پروژه اتصال ESP32 با Aurduino IDE بود تا بتوانیم برنامه دلخواهمان را بر روی آن بریزیم. در این بخش نیز با چالش‌هایی روبرو شدیم که در نهایت رفع شد. قدم بعدی پیدا کردن بهترین روش برای ارتباط با ChatGPT و تبدیل متن و صوت به یکدیگر بود. در این راستا تحقیقاتی انجام دادیم و در نهایت از api key برای ارتباط با ChatGPT و همچنین از کتابخانه‌های

speech_recognition و gtts در پایتون برای تبدیل صوت و متن به یکدیگر استفاده کردیم. Prompt لازم را نیز آماده کردیم تا ChatGPT با دستورات valid آشنا باشد و بداند برای هر دستور چه اقدامی را باید انجام دهد. در مرحله بعد این اتصالات و کد ها را به فرمت سرور با استفاده از کتابخانه flask تبدیل کردیم و با نوشتن یک کلاینت در زبان پایتون صحت عملکرد آن را بررسی کردیم.

تا این مرحله از پروژه می‌توانستیم سوالات خود را به صورت صوتی از طریق میکروفون لپ‌تاپ به ChatGPT بدهیم و پاسخ را نیز به صورت صوتی دریافت کنیم. گام بعدی استفاده از میکروفون و ESP32 برای دریافت و فرستادن صدا بود. برای اینکار میکروفون را به ESP32 وصل کردیم تا بتواند صدا را دریافت کند. حافظه ESP32 بسیار کم است و در نتیجه اگر می‌خواستیم دستورات کاربر را به طور کامل دریافت کنیم و سپس از ESP32 آن‌ها را برای کامپیوتر بفرستیم تا پردازش‌های لازم را انجام دهد، احتمالا با مشکل حافظه روبرو می‌شدیم. در نتیجه از شبکه و web socket ها استفاده کردیم و صداهای دریافتی را به صورت chunk های کوچک تر و online برای سرور می‌فرستادیم و با کنار هم قرار دادن آن‌ها در سرور دستور کامل دریافت می‌شود.

گام نهایی اتصال کامپیوتر سرور که کد مربوط به ChatGPT بر روی آن اجرا می‌شود با tia portal است. اتصال آن‌ها نیز از طریق شبکه و کتابخانه pymodbus انجام شد و کامپیوتر سرور توانست با موفقیت به رابط ethernet در شبیه ساز tia portal متصل شود و دستورات را برای آن ارسال کند تا آن‌ها را اجرا کند.

چالش‌ها

در این پروژه ما با چالش‌های زیادی روبرو شدیم. اولین چالش این بود که نمایشگر HMI در آزمایشگاه نبود پس باید به جای آن از شبیه‌ساز tia portal استفاده می‌کردیم تا plc را شبیه سازی کند و تغییرات را بر روی آن اعمال کنیم. چالشی که در این قسمت با آن روبرو شدیم این بود که tia portal بر روی مک نصب نمی‌شد و ما فقط یک لپ‌تاپ ویندوزی داشتیم پس مجبور بودیم با آن کار کنیم. و البته نسخه ۱۷ tia portal دارای مشکلات زیادی بود و کل تنظیمات لپ‌تاپ را بهم زد. و روند پاک کردن آن به طور کامل و نصب یک نسخه قدیمی تر بسیار زمان برد. چالش دیگری که در این بخش داشتیم مشکلات شبیه ساز tia portal بود که IP و port مربوط به ethernet ایجاد شده در آن را تنظیمات خود tia portal متفاوت بود و در برقراری ارتباط با کتابخانه snap7 که مخصوص plcهای سری s7 شرکت زیمنس ساخته شده مشکلات زیادی داشت از جمله ارورهای متفاوت و عجیبی که در حین connect شدن با آن مواجه شدیم و مجبور به انتخاب بسترهای دیگر ارتباطی مانند modbus و socket شدیم.

دومین چالش این بود که Aurduino IDE را نمی‌توانستیم در لپ‌تاپ ویندوز به ESP32 متصل کنیم. در نهایت با تغییر لپ‌تاپ و امتحان مجدد این مشکل درست شد و توانستیم کدها را بر روی ESP32 آپلود کنیم. ولی خب چالش جدید ایجاد شده این بود که حالا کامپیوتر اصلی که به ESP32 متصل می‌شد با کامپیوتری که plc بر روی آن اجرا می‌شد فرق داشت. برای حل این مشکل در ابتدا از شبکه استفاده

کردیم تا این دو کامپیوتر بتوانند با هم ارتباط داشته باشند و دستورات از کامپیوتر متصل به ESP32 به کامپیوتر دوم فرستاده شود اما مشکلی که در ادامه داشتیم این بود که بستر ارتباطی ethernet که در شبیه ساز tia portal بوجود می‌آمد تنها قابل استفاده در کامپیوتر اجرا کننده بود و برای همین مجبور به درست کردن مشکل کامپیوتر ویندوزی شدیم که سرانجام با استفاده از هاب این مشکل نیز برطرف شد.

چالش بعدی ما api key برای ChatGPT و همچنین انتخاب روش مناسب برای برای تبدیل صوت و متن به یکدیگر بود. برای شارژ api key باید پول میدادیم که با کمی گشتن یک روش رایگان برای آن پیدا کردیم که با تغییر base url و فرستادن درخواست به یک آدرس دیگر و سپس برای خود ChatGPT کار می‌کرد. برای تبدیل صوت و متن به یکدیگر انتخاب اول google clouds بود ولی این ابزار هم پولی بود. در نتیجه تصمیم گرفتیم که از کتابخانه‌های خود پایتون که اینکار را انجام می‌دهند استفاده کنیم و api نخریم.

چالش بعدی فرستادن سوال صوتی کاربر به سرور بود. دو راه برای این کار وجود داشت. راه اول این بود که صدای فرد در خود ESP32 ذخیره شود و سپس به سمت سرور ارسال شود. این راه به توجه به محدود بودن حافظه ESP32 راه خوبی نبود. راه دوم این بود که این صدا از طریق شبکه و به صورت زنده برای سرور فرستاده شود که در نهایت از همین راه استفاده کردیم. البته در این بین چالشی که داشتیم ذخیره کردن این درخواست صوتی در سرور بود که با تحقیقاتی که انجام دادیم فهمیدیم چطور می‌توان آن را از یک url خاص که با توجه به نت مشترک ESP32 و سرور مشخص می‌شود، دانلود و در سرور ذخیره کرد.

چالش دیگری که داشتیم بحث اتصال tia portal به شبکه بود. برای این کار در ابتدا از کتابخانه snap7 استفاده کردیم که برای اتصال plc به شبکه استفاده می‌شود اما در این کار مشکلات و ارورهای زیادی داشتیم. در ابتدا متوجه شدیم که IP ایی که در شبیه ساز set شده با IP ایی که ما در tia portal مشخص کرده بودیم تفاوت دارد. بعد از درست کردن این مورد در کد باز هم به ارور های عجیب دیگری بر می‌خوردیم که متوجه شدیم به دلیل متفاوت بودن port بستر ارتباطی ایجاد شده با حالت دیفالت آن در plc هاست. بعد از درست کردن این مورد نیز باز به ارور جدیدی برخوردیم که در اینترنت توضیح داده بودند به دلیل کمبود حجم قابل ارسال است با اینکه ما در مرحله اتصال به این ارور خورده بودیم. به همین دلیل بستر را ابتدا به socket تغییر دادیم که بخش اتصال به درستی انجام می‌شد اما در فرستادن پیام هیچ responseایی دریافت نمی‌کرد برای همین بستر را به modbus تغییر دادیم.

توضیحات مربوط به کد

فرستادن فایل صوتی به سرور

فایل client.py مسئولیت این بخش را بر عهده دارد که رابط بین ESP32 و سرور است. صدای فرستاده شده از سمت ESP32 از طریق یک url مخصوص در دسترس است و این url با توجه به اینترنتی که

ESP32 و سرور به آن متصل‌اند مشخص می‌شود. در این فایل متغیر audio_url برابر این url قرار داده می‌شود. سپس اگر دستور download زده شود این فایل دانلود شده و سپس با دستور send می‌توان آن را برای سرور و یا همان کامپیوتر مرکزی فرستاد تا process مربوط به ChatGPT انجام شود.

Audio process

تابع process_audio در فایل server.py فایل فرستاده شده را بررسی می‌کند که که از نظر فرمت درست باشد(فرمت مورد نظر wav است) در این قسمت در صورت درست بودن فرمت فایل تابع process_audio_file صدا زده می‌شود تا تبدیل صوت به text و بالعکس و ارتباط با ChatGPT شکل بگیرد.

ارتباط با ChatGPT

برای ارتباط با ChatGPT از api key استفاده شد. البته در حالت عادی برای شارژ api key باید اکانت می‌داشتیم و پول پرداخت می‌کردیم ولی یکی از repositoryهای گیتهاب api key را به صورت رایگان در اختیار می‌گذاشت که از آن استفاده می‌کردیم. در واقع این api key با تغییر base url ابتدا سوالات را به آدرس دیگری می‌فرستاد و سپس از آن جا برای ChatGPT ارسال می‌کرد.

کد این بخش در ادامه آمده است. سوال مورد نظر را می‌توان در قسمت content پرسید و جواب را در متغیر response دریافت کرد:

```
from openai import OpenAI
client = OpenAI(
    api_key="sk-M4Jl5jb0AthtsEo1mFaULZMTb8rJfWMclyDQuFOsGte3ZeDm",
    base_url="https://api.chatanywhere.tech/v1"
)

response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": "question"}],
    stream=True
)
for chunk in response:
    if chunk.choices[0].delta.content is not None:
        print(chunk.choices[0].delta.content, end="")
```

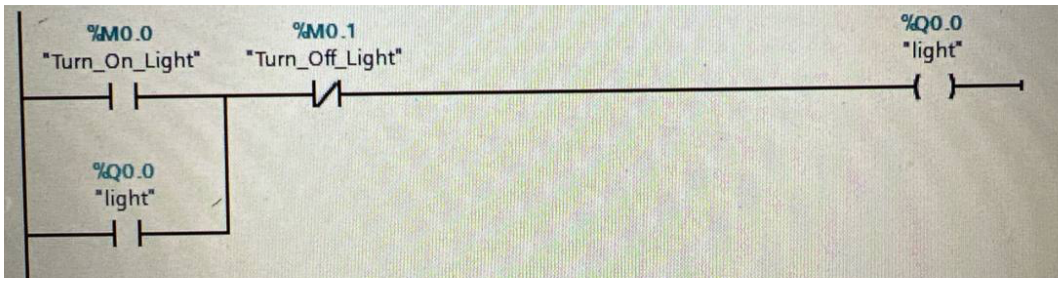
در این قسمت prompt ای نیز آماده شد تا سوال کاربر قابل فهم برای ChatGPT باشد و بتواند دستورات valid را از دستورات غیر valid تشخیص دهد و ۲ جواب، یکی tag مربوطه که باید تغییر کند و دیگری متن جوابی که می‌خواهیم به کاربر دهیم را به ما در ۲ خط متفاوت و با ترتیب مشخص خروجی دهد. در حالت فعلی که ما تعریف کردیم تنها دو دستور روشن و خاموش کردن چراغ وجود دارد.

prompt_question = "Hi GPT, i have following commands for my plc, i want you to process the following text and find what command should be run or there is no command suitable for it. the format of answer i want is that in first line just write the command name or None and then in the next line write your descriptions. "

prompt_question += " commands: 1. turn_on_light 2. turn_off_light "

برنامه PLC

برنامه‌ایی که برای این بخش نوشتیم شامل ۳ تگ turn_on_light, turn_off_light, light بود که دوتا اول از جنس حافظه و آخری از جنس خروجی است. خود برنامه نیز به شکل زیر نوشته شده است. برنامه به این صورت است که اگر turn_on_light در آن true باشد و turn_off_light در آن false باشد مقدار light در انتها true می‌شود و در غیر اینصورت false می‌باشد. برای همین نیز در سرور ۲ تابع مربوط به روشن یا خاموش کردن لامپ داریم که مقادیر این ۲ تگ را تغییر می‌دهند.



تبدیل صوت به متن

گرفتن google cloud api برای این تبدیل دردسرهای خود را داشت پس تصمیم گرفته شد که از کتابخانه‌های آماده زبان پایتون برای تبدیل صوت به متن و برعکس استفاده کنیم. برای تبدیل صوت به متن از کتابخانه speech_recognition استفاده کردیم. با استفاده از این کتابخانه یک recognizer می‌توان ساخت که یک فایل صوتی wav را دریافت کند و آن را به string تبدیل کند. در کد زیر فایل record_out.wav خوانده می‌شود و خروجی آن به صورت متن در متغیر question قرار می‌گیرد.

```
import speech_recognition as sr
from pydub import AudioSegment

recognizer = sr.Recognizer()
recognizer.energy_threshold = 300

audio = AudioSegment.from_wav("record_out.wav")
audio_file = sr.AudioFile("record_out.wav")

with audio_file as source:
    audio_file = recognizer.record(source)
    try:
        question = recognizer.recognize_google(audio_file)
```

```
    except sr.UnknownValueError:
        print("unknown error")
```

تبدیل متن به صوت

برای تبدیل متن به صوت از کتابخانه gtts استفاده می‌کنیم که کار با آن بسیار ساده است. در کد زیر جواب ChatGPT که به صورت متنی است و در فایل answer_text داده شده است به این کتابخانه پاس داده می‌شود و آن را به یک فایل صوتی به نام response.mp3 تبدیل می‌کند که می‌توان آن را در صورت نیاز دستی با استفاده از AudioSegment.from_mp3 به فرمت wav. تبدیل کرد.

```
tts = gTTS(text=answer_text, lang='en', slow=False)
mp3_path = "response.mp3"
tts.save(mp3_path)
```

اتصال به plc

در سرور کد های مربوط به این قسمت با اجرا شدن تابع handling_plc_command_sending اجرا می‌شود و به طور خاص برای تست این بخش فایل connecting_to_plc.py را درست کردیم که کار این قسمت را انجام می‌دهد. ما باید بتوانیم سرور را به plc وصل کنیم تا بتواند دستورات را به آن بفرستد. توابع موجود در این فایل این کار را برای ما انجام دادند و به وسیله توابع write_register و read_register می‌توانستیم register های plc که مربوط به tag های بخش مموری هستند را بنویسیم و بخوانیم.

نتیجه

هدف این پروژه ساخت یک دستیار صوتی هوشمند بود که بتواند به کاربر در کار کردن با تابلو برق و یا نمایشگر کمک کند. تمام مراحل این پروژه با موفقیت انجام و تست شد. اتصال ESP32 به کامپیوتر سرور به طور کامل انجام شد. دستورات کاربر از طریق ESP32 برای سرور فرستاده می‌شود و سرور دستور صوتی فرستاده شده را تبدیل به متن می‌کند و با format مناسب برای ChatGPT می‌فرستد و پاسخ را نیز دریافت می‌کند.

پاسخ به درستی تبدیل به صوت شده و برای کاربر پخش می‌شود تا از وضعیت دستور خود اطلاع پیدا کند. در صورت اجرایی بود دستور ارتباط میان سرور و tia portal به درستی انجام می‌شود و دستور فرستاده می‌شود. تنها ایرادی که در این پروژه نتوانستیم آن را رفع کنیم این است که این دستور در سمت

tia portal واقعا اجرا شود و برای مثال در صورتی که کاربر درخواست روشن شدن LED را دارد، این اتفاق واقعا بیفتد که آن نیز با تحقیقاتی که انجام دادیم به این علت بود که PLCSIM شبیه سازی اتصالات شبکه‌ایی را نمی تواند به صورت کامل انجام دهد و چالش‌هایی در این ارتباطات برای آن وجود دارد که در plc با آن روبرو نخواهیم شد و انتظار می‌رود کدهای مربوط به برقراری ارتباط به درستی در plc اجرا شوند.