

# Govorom upravljani ugradbeni računalni sustav temeljen na Amazon Alexa Voice Service tehnologiji

---

**Brlečić, Lidija**

**Master's thesis / Diplomski rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

*Permanent link / Trajna poveznica:* <https://urn.nsk.hr/urn:nbn:hr:168:643804>

*Rights / Prava:* [In copyright](#)/[Zaštićeno autorskim pravom.](#)

*Download date / Datum preuzimanja:* **2025-03-14**



*Repository / Repozitorij:*

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 84

**GOVOROM UPRAVLJANI UGRADBENI RAČUNALNI  
SUSTAV TEMELJEN NA AMAZON ALEXA VOICE SERVICE  
TEHNOLOGIJI**

Lidija Brlečić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 84

**GOVOROM UPRAVLJANI UGRADBENI RAČUNALNI  
SUSTAV TEMELJEN NA AMAZON ALEXA VOICE SERVICE  
TEHNOLOGIJI**

Lidija Brlečić

Zagreb, lipanj 2024.

## DIPLOMSKI ZADATAK br. 84

Pristupnica: **Lidija Brlečić (0036509510)**

Studij: Elektrotehnika i informacijska tehnologija

Profil: Elektroničko i računalno inženjerstvo

Mentor: prof. dr. sc. Davor Petrinović

Zadatak: **Govorom upravljani ugradbeni računalni sustav temeljen na Amazon Alexa Voice Service tehnologiji**

### Opis zadatka:

U okviru diplomskog rada potrebno je istražiti mogućnost izgradnje ugradbenog računalnog sustava koji omogućava upravljanje uređajem putem prirodnog govora, tj. primanje govorne naredbe korisnika putem mikrofona, odnosno reakciju sustava putem izgovorene poruke ili stvarne akcije izlaznih aktuatora ovog sustava. Sustav temeljiti na programskim rješenjima Amazon Alexa Voice Service, a kao operacijski sustav za ugradbeno računalno odabrati odgovarajuću Linux distribuciju. Istražiti i odabrati pogodne gotove sklopovske platforme male cijene poput Raspberry Pi Zero uz odgovarajući modul za audio ulaz i izlaz. Integrirati sustav i instalirati svu potrebnu programsku podršku na sustav i demonstrirati njegov rad, tj. govorno upravljanje sustavom. U konačnici istražiti mogućnost izgradnje cijelog sustava u vlastitoj izvedbi umjesto korištenja gotovih modula, uz analizu dobavljalivosti potrebnih elektroničkih komponenata i njihovu cijenu. Za sve detalje javiti se mentoru.

Rok za predaju rada: 28. lipnja 2024.



*Veliko hvala mentoru prof. dr. sc. Davoru Petrinoviću na prenesenom znanju te pruženoj potpori i razumijevanju tokom cijelog studija.*

*Mami Vlatki, tati Zdenku i sestri Heleni - bez vas danas ne bih bila ovdje gdje jesam. Svaki moj uspjeh je odraz vaše ljubavi, pomoći i potpore, stoga ovaj diplomski rad shvatite kao rezultat svega što ste mi pružili tokom studija, ali i puno prije njega.*

*P.S. Helena - I hope you are my sister in every universe*

# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Virtualni asistenti</b>	<b>2</b>
2.1. Općenito . . . . .	2
2.2. Amazon Alexa . . . . .	4
2.3. OpenAI ChatGPT . . . . .	7
<b>3. Hardverski dio izvedbe sustava</b>	<b>10</b>
3.1. Inicijalna ideja i problemi . . . . .	10
3.1.1. Raspberry Pi Zero W i ReSpeaker Pi HAT . . . . .	10
3.1.2. Raspberry Pi Codec Zero . . . . .	15
3.2. Raspberry Pi 4B, USB mikrofoni i zvučnici . . . . .	20
<b>4. Softverski dio izvedbe sustava</b>	<b>24</b>
4.1. Operacijski sustav . . . . .	24
4.2. Interakcija s ChatGPT-om . . . . .	27
4.3. Dodavanje LED diode . . . . .	32
<b>5. Vlastita izvedba sustava</b>	<b>37</b>
5.1. Minimum potrebnog hardvera . . . . .	37
5.2. Ideja vlastitog PCB-a . . . . .	40
<b>6. Zaključak</b>	<b>42</b>
<b>Literatura</b>	<b>43</b>

# 1. Uvod

Digitalna obrada govora je multidisciplinarno područje koje se bavi obradom signala ljudskog govora pomoću računalnih tehnologija. Neki od brojnih zadataka obrade govora uključuju prepoznavanje govora, govornika ili jezika, analizu govora, poboljšanje kvalitete govora i sintezu govora iz teksta.

Uz velike napretke tehnologije, pa tako i eksponencijalnim razvitkom umjetne inteligencije, došlo je do potrebe da se ona utilizira za čovjekovu interakciju s računalom, čime se dolazi do pojave glasovnih asistenata. Za potrebe ovog diplomskog rada bit će istražena mogućnost izgradnje ugradbenog računalnog sustava koji će omogućiti interakciju s upravo jednom takvom tehnologijom - Amazonovom Alexom, uz korištenje gotovih hardverskih modula. Kao što će se pokazati u nastavku, Alexu nije bilo moguće implementirati te je umjesto nje odabran chatbot asistent ChatGPT tvrtke OpenAI. Za potrebe rada bit će istraženi moduli tvrtke Raspberry Pi Foundation, te ostala hardverska i softverska rješenja koja bi omogućila izgradnju sustava. Na posljetku, cilj je istražiti mogućnost izgradnje sustava u vlastitoj izvedbi te dati analizu cijene i dobavljalivosti potrebnih električkih komponenata.

## 2. Virtualni asistenti

### 2.1. Općenito

Digitalna obrada govora se kao grana obrade signala razvija od 50-ih godina prošlog stoljeća, gdje su se koristili primitivni sustavi kojima je fokus rada bio na osnovnim zadacima poput prepoznavanja fonema i jednostavnih riječi. Nadalje se napredovalo, preko uvođenja skrivenih Markovljevih modela koji su omogućili statističko modeliranje govora i time uveli mogućnost varijacije izgovora i brzine govora, do 90-ih godina kada se pojavljuju prvi komercijalni proizvodi za prepoznavanje govora. Cijeli povijesni razvitak područja je vodio omogućavanju interakcije čovjeka i računala putem govora, što je pogotovo uzelo maha sve bržim razvojem tehnologije, primarno umjetne inteligencije.

Rapidan razvoj umjetne inteligencije doveo je do pojave virtualnih asistenata, čiji je napredak obilježen nizom inovacija i tehnoloških napredaka od samih početaka, pa sve do današnjih sofisticiranih modela. Virtualni asistenti su softverski agenti koji služe za obavljanje raznih zadataka putem prirodne interakcije između korisnika, koji naredbe zadaju najčešće u obliku tekstualnih ili govornih naredbi, i nekakvog računala. Prvi koji je izašao na tržište s istim je bila tvrtka Apple s mobilnim virtualnim asistenom Siri, kojeg je predstavila na mobilnom telefonu iPhone 4S krajem 2011. godine. U narednim godinama uslijedio je razvitak virtualnih asistenata od strana kompetitora, pa su tako danas na tržištu dostupni Google Assistant, Amazon Alexa, Microsoft Cortana, Samsung Bixby i drugi. Usprkos sve većem broju opcija na tržištu i različitih mogućnosti koje pojedine tvrtke nude, ključne funkcionalnosti i tehnologija kojom se one postižu su im svima zajedničke.

Tehnologija na kojoj počiva cijela ideja i koja je ključna u funkcioniranju virtualnih asistenata je obrada prirodnog jezika (engl. *Natural Language Processing*) (u daljnjem tekstu NLP). NLP je interdisciplinarno područje koje povezuje računalnu znanost, lingvistiku i umjetnu inteligenciju kako bi računalima omogućilo razumijevanje, interpretaciju i generiranje govora. Virtualni asistenti koriste NLP kako bi obradili korisničke zahtjeve i na njih prikladno odgovorili. Cijeli proces djelovanja NLP-a unutar asistenata je opisan u nastavku.

Prvo što virtualni asistent treba napraviti je prepoznati da korisnik želi izreći neki upit

ili komandu, odnosno prepoznati njegov govor. Tu se koristi automatsko prepoznavanje govora(engl. *Automatic Speech Recognition*)(u daljnjem tekstu ASR), što je tehnologija koja omogućava prepoznavanje i interpretaciju govora. Tu se najčešće koriste akustički i jezični modeli te rječnici; zadatak akustičkih modela je analiza zvučnih signala radi prepoznavanja fonema, dok su jezični modeli zaduženi za procjenu vjerojatnosti slijeda riječi u jeziku. ASR sustavi pretvaraju govorni jezik u tekst pomoću različitih algoritama i modela.

Nadalje se nad tekstom kojeg je ASR sustav generirao provodi niz koraka NLP-a radi razumijevanja sadržaja i konteksta upita. Prvi od njih je tokenizacija - proces "razbijanja" teksta na manje jedinice koje se nazivaju tokeni kako bi se olakšala daljnja analiza teksta. Tokeni mogu biti riječi, fraze ili ponekad čak cijele rečenice. Nakon toga se provodi lematizacija, odnosno redukcija riječi na njihov osnovni ili kanonski oblik(lema) radi smanjenja redundantnosti, te sintaktička i semantička analiza. Sintaktička analiza(engl. *parsing*) gleda na koji se način riječi povezuju i formiraju rečenice i time daje temelj za semantičku analizu. Ona se bavi razumijevanjem značenja teksta i stavljanje riječi i rečenica u kontekst, a ključna je za pružanje relevantnih odgovora na korisnički upit.

Iduća korištena tehnologija koju NLP koristi je prepoznavanje entiteta. Prepoznavanjem entiteta(engl. *Named Entity Recognition*)(u daljnjem tekstu NER) se tokeni teksta pregledavaju kako bi se identificirali oni koji potencijalno predstavljaju entitete. Takvi tokeni se zatim razvrstavaju u odgovarajuće unaprijed definirane kategorije kao što su imena ljudi, lokacija, događaja i slično. Moderni NER sustavi koriste algoritme strojnog i dubokog učenja za obavljanje svoje funkcionalnosti. Duboko i strojno učenje također igra značajnu ulogu u razumijevanju namjere(engl. *Intent Recognition*) - procesu koji služi za identifikaciju i klasifikaciju svrhe upita, kako bi se mogao dati ispravan odgovor.

Nakon identificirane namjere, sustav generira odgovor koristeći algoritme generiranja prirodnog jezika(engl. *Natural Language Generation*)(u daljnjem tekstu NLG). U ovom koraku se prikupljaju informacije temeljene na namjeri upita, te se one strukturiraju na smisleni način. Korištenjem jezičnog modela pomoću kojeg se odabiru odgovarajuće riječi i sintaksa, generira se gramatički ispravan i lako razumljiv tekst, odnosno odgovor na upit korisnika u obliku teksta. Konačno, virtualni asistent izvršava odgovarajuće naredbe poput pružanja informacija, reprodukcije glazbe, stavljanja podsjetnika u kalendar i slično.

Nemali broj puta kada se priča o virtualnim asistentima podrazumijeva se pojam glasovnog asistenta, to jest da odgovor ne ostaje isključivo(ili uopće) u tekstualnom, već u govornom obliku.

Tehnologija koja to omogućava i pretvara tekstualni sadržaj u govor naziva se sinteza govora(engl. *Text-to-Speech*, TTS). Postoji više metoda kojima se sinteza može postići, primjerice koristeći formante(rezonantne frekvencije vokalnog trakta) ili konkatencijom kra-

ćih segmenata snimljenog govora - konkatenativna sinteza. Ona se temelji na velikoj biblioteci govornih fragmenata koje je izgovarala jedna osoba, koji se onda spajaju i tvore riječi i zvukove. Postoje i naprednije metode sinteze temeljene na dubokim neuronskim mrežama, primjerice WaveNet i Tacotron.

WaveNet je duboka neuronska mreža razvijena od strane tvrtke DeepMind, predstavljena 2016. godine, pomoću koje se generiraju relativno realistični glasovi tehnikom direktnog modeliranja zvučnih signala. Mreža je trenirana na snimkama stvarnog govora te je sustav svojom pojavom na tržištu nadmašio neke od dotadašnjih najboljih sustava za sintezu. Drugi od modela je Googleov Tacotron, generativni *end-to-end* model sinteze govora koji uzima sekvencu fonema i kao izlaz daje odgovarajući spektrogram. Ti spektrogrami se zatim pretvaraju u zvučne signale korištenjem vokodera kao što je WaveNet. Postoji naprednija i novija verzija Tacotron 2 koja kombinira model za generiranje spektrograma s upravo spomenutim vokoderom kako bi se postigla visoka kvaliteta sintetiziranog govora.

## 2.2. Amazon Alexa

Alexa, virtualni(odnosno glasovni) asistent kojeg je razvila tvrtka Amazon, je jedna od najprepoznatljivijih i najkorištenijih glasovnih asistenata današnjice. Stvoren je s ciljem da omogući prirodan način interakcije između ljudi i pametnih uređaja, pritom odgovarajući na upite vezane za prikupljanje informacija s interneta, reproduciranja glazbe, postavljanja alarma i slično. Tehnologije koje se koriste su ranije objašnjeni NLP i grane umjetne inteligencije. Alexa je javnosti predstavljena 2014. godine zajedno s pametnim zvučnikom Amazon Echo prikazanog na slici 2.1. Od tada se Alexa konstantno unaprjeđuje i integrira u mnoge uređaje, što Amazonove, što druge.



**Slika 2.1:** Nekoliko generacija Amazon Echo zvučnika

Još jedna od opcija zbog kojih je uporaba Alexe toliko raširena je mogućnost integriranja u *smart home* sustave. Drugim riječima, korisnici mogu glasovnim komandama upravljati različitim funkcijama u domu poput paljenja i gašenja svjetla i električnih uređaja, podizanje

i spuštanje roleta, upravljanje interfonom itd. Korisnicima je to omogućeno kroz Alexa Voice Service alate, kao i Alexa Skills Kit.

Alexa Skills Kit(u daljnjem tekstu ASK) je zbirka alata, API funkcija i dokumentacije koja korisnicima omogućuje izrade vlastitih vještina za Alexu, što su zapravo aplikacije koje proširuju njenu funkcionalnost. Kada se te *third-party* aplikacije objave, one postaju dostupne svim uređajima koji podržavaju Alexu(u daljnjem tekstu Alexa-enabled uređaji) a korisnici ih mogu omogućiti putem Alexine aplikacije.

Alexa Voice Service(u daljnjem tekstu AVS) je također skup alata i SDK-ova(engl. *Software Development Kit*) koji omogućuju integraciju Alexe na različite vrste hardvera: od televizora, termostata, pametnih satova pa sve do automobila. Preko AVS-a se pristupa Alexinim glasovnim sposobnostima u oblaku, gdje se odvijaju ASR, NLP, TTS i ostali uobičajeni koraci u glasovnim asistentima koji su opisani u prethodnom potpoglavlju. Uređaji oslušuju za ključnu riječ(najčešće "Alexa") kako bi se aktivirali, potom šalju podatke na Amazonove servere gdje se onda obrađuju. Nakon obrade vraćaju odgovor u obliku zvučnog signala koje uređaji reproduciraju korisniku.

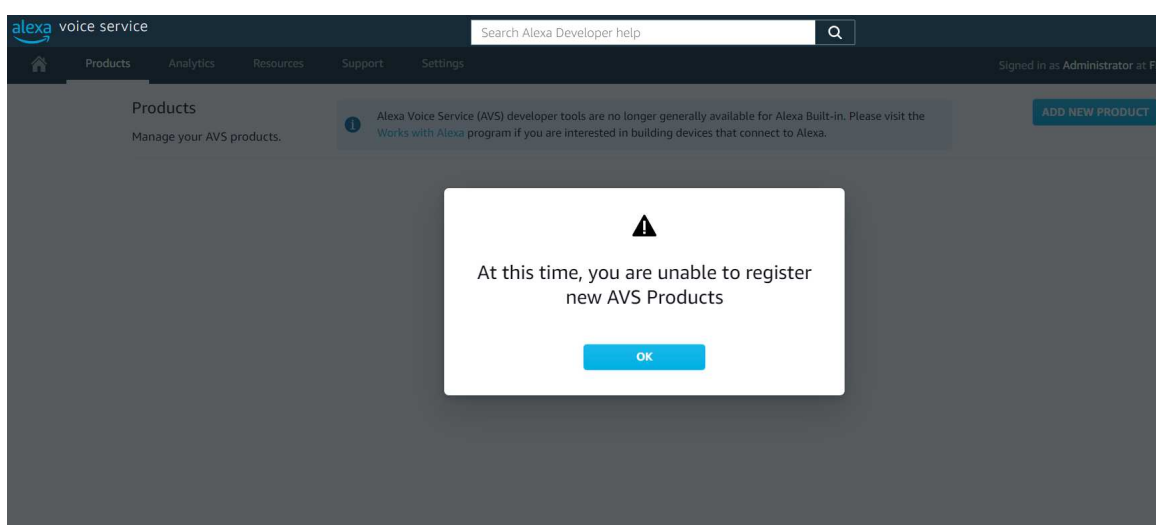
Kao što je spomenuto u uvodu, inicijalna ideja diplomskog rada je bila izgradnja ugradbenog računalnog sustava upravljanog prirodnim govorom odnosno primanje naredbi govornim putem, a kao softverski dio sustava omogućiti Alexu. Drugim riječima, učiniti ugradbeni sustav Alexa-enabled. Međutim, unazad nekoliko mjeseci je došlo do značajnih promjena kako developeri mogu koristiti AVS usluge; AVS SDK i API više nisu javno dostupni i mogući za korištenje. Bez obzira na recentne izmjene u uvjetima korištenja, upućen je upit Amazonovoj podršci za developere bi li izašli ususret s obzirom da se radi o akademskom radu te je dio njihovog odgovora prikazan na slici 2.2. Ukratko, SDK alati se više ne održavaju ni podržavaju, a samim time se ni ne mogu koristiti. Works with Alexa na kojeg upućuju je program certificiranja koji je namijenjen proizvodnji uređaja za komercijalne svrhe, pa s obzirom na to nije pogodan u svrhu izrade diplomskog rada. Na slici 2.3 prikazan je zaslon Amazonove developer konzole u trenutku kad se pokušava dodati novi AVS proizvod što je prije bio jedan od prvih koraka pri izgradnji Alexa-enabled uređaja.

However, as of January 15th, 2024, there have been significant changes in how developers can build Alexa-enabled devices. Specifically, the Alexa Voice Service (AVS) SDK tools have been retired. This means the resources on GitHub are now "read only" and can no longer be used to build new Alexa Built-In (ABI) devices. Additionally, Amazon no longer maintains or updates the AVS SDK, nor do we provide design, development, certification, or support resources for new ABI products.

For your academic and research purposes, this means you won't be able to use the AVS SDK to implement Alexa on your Raspberry Pi directly. However, there are still opportunities for exploring Alexa integration through the Works with Alexa program, which supports connecting devices to Alexa. You can find more details about this program and its requirements here:

Works with Alexa Program: [https://developer.amazon.com/en-US/docs/alexa/smarthome/wwa-overview.html?mkt\\_tok=MzY1LUVGSS0wMjYAAAGQMryK\\_ygmLmFAF851rbYzpNyL7YrboWjcg0MvVa3mp8VWHXajgyWlftV9cSl7jim6qSB\\_sJjv21\\_c7Ws9mvFLH4Rzr3gr1B\\_DAgC58M2QzePV4s](https://developer.amazon.com/en-US/docs/alexa/smarthome/wwa-overview.html?mkt_tok=MzY1LUVGSS0wMjYAAAGQMryK_ygmLmFAF851rbYzpNyL7YrboWjcg0MvVa3mp8VWHXajgyWlftV9cSl7jim6qSB_sJjv21_c7Ws9mvFLH4Rzr3gr1B_DAgC58M2QzePV4s)

## Slika 2.2: Odgovor Amazonove podrške na upit



Slika 2.3: Pokušaj dodavanja novog AVS proizvoda

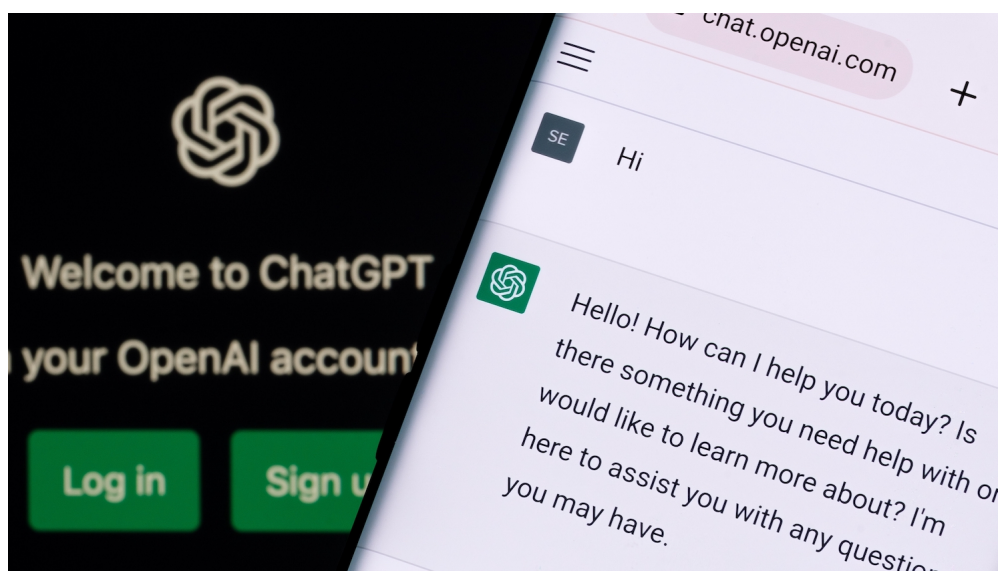
S obzirom na neočekivanu nemogućnost korištenja AVS SDK-a i korištenje Alexe općenito, bilo je potrebno pronaći alternativnu tehnologiju koja bi bila pogodna za omogućavanje slične funkcionalnosti sustava.



## 2.3. OpenAI ChatGPT

Kao alternativa Alexi odabran je napredni chatbot i virtualni asistent ChatGPT tvrtke OpenAI predstavljen tržištu krajem 2022. godine. Chatbotovi su općenito softverske aplikacije ili web sučelja osmišljena kao imitacije ljudskog dijaloga kroz tekstualnu ili govornu interakciju. Dok moderni chatbotovi koriste sustave umjetne inteligencije, izvedbe jednostavnijih chatbotova postoje već desetljećima.

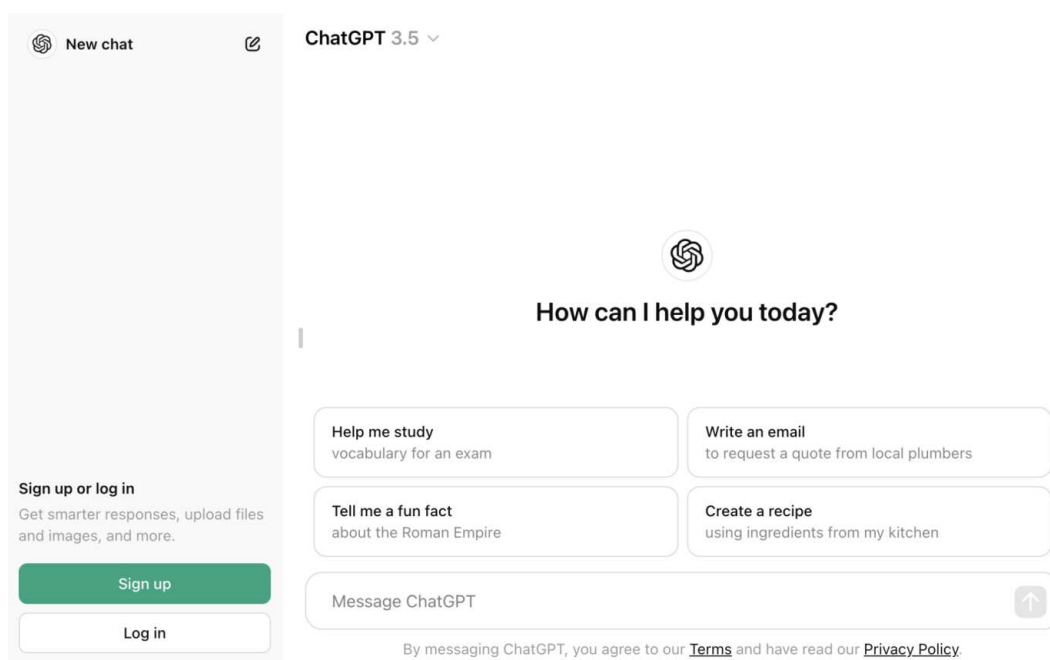
ChatGPT je temeljen na specifičnim generativnim predtreniranim transformatorima(engl. *Generative Pre-trained Transformers*)(u daljnjem tekstu GPT). Transformator modeli su u ovom kontekstu arhitekture dubokog učenja koje su revolucionarizirale NLP. Tekst se pretvara u tokene te se uz tablicu ugrađivanja riječi oni konvertiraju u vektore. GPT-jevi su dizajnirani za generiranje tekstualnog sadržaja koji je koherentan i odgovara kontekstu upita. Na slikama 2.5 i 2.4 prikazano je kako izgleda početno sučelje i tekstualna interakcija s ChatGPT-om.



Slika 2.4: Primjer interakcije s ChatGPT-om

Trening modela se sastoji od faze predtreniranja i faze *fine-tuninga*. Predtreniranje uključuje obuku modela na velikoj količini tekstualnih podataka te se na taj način uči statističkim obrascima jezika. Kod *fine-tuninga* se model dodatno trenira na specifičnijim skupovima podataka radi poboljšanja točnosti i relevantnosti odgovora. Svi algoritmi treniranja modela se temelje na stojnom učenju(engl. *Machine Learning*), u slučaju ChatGPT-ja koriste se metode nadziranog i nenadziranog učenja. Nenadzirano učenje koristi neoznačene podatke za samostalnu analizu istih kako bi model sam pronašao grupacije i slične odnose među njima, te se koristi u fazi predtreniranja. Nadzirano se učenje s druge strane koristi označenim podacima za treniranje modela, odnosno svaki ulazni podatak ima pripadajuću

etiketu ili odgovor. Na taj način model uči prepoznavati obrasce između ulaza i izlaza kako bi mogao predviđati odgovore za nove, neoznačene podatke. Takvo se učenje koristi u fazi *fine-tuninga*.



**Slika 2.5:** Prikaz početnog sučelja

ChatGPT je u ovome radu izabran kao virtualni asistent zbog svoje opravdane popularnosti te mogućnosti koje pruža u olakšavanju dohvata informacija s interneta i asistiranju u svakodnevnom životu. Tvrtka OpenAI uz chatbot također razvija i druge platforme koje koriste umjetnu inteligenciju za generiranje slika i glasova, a od posebnog je značaja njihova API platforma koja developerima omogućuje pristupanje GPT modelima putem jednostavnih naredba i HTTP zahtjeva. Upravo ona omogućuje integraciju umjetne inteligencije i mogućnosti NLP-a u različite uređaje bez potrebe za detaljnijim znanjima o modelima dubokog učenja ili potrebnoj infrastrukturi za njihovo izvođenje. API sustav je izgrađen kako bi mogao podržavati različite zahtjeve, od jednostavnijih i manjih zadataka sve do velikih korporativnih projekata.

S obzirom da NLP modeli rade na način da tekst razbijaju na manje jedinice odnosno tokene, kako je objašnjeno u prethodnim potpoglavljima, tako se i pristup GPT modelima naplaćuje prema potrošenim tokenima tijekom korištenja usluge. To uključuje razlaganje korisničkog upita API-ju na tokene kao i odgovor na upit. Tipično je jedan token ekvivalentan otprilike 4 znaka engleskog teksta ili 0.75 riječi. Cijeli niz znakova uključujući razmake, interpunkcijske i posebne znakove se također računa u tokene. Naplata po tokenima za pristup najnovijem modelu GPT-4o, koji sadrži pristup dostupnim informacijama sve do listopada 2023. godine, prikazan je na slici 2.6.

Pricing  Show prices per 1K tokens

## GPT-4o

GPT-4o is our most advanced multimodal model that's faster and cheaper than GPT-4 Turbo with stronger vision capabilities. The model has 128K context and an October 2023 knowledge cutoff.

[Learn about GPT-4o ↗](#)

Model	Pricing	Pricing with Batch API*
gpt-4o	\$5.00 / 1M input tokens	\$2.50 / 1M input tokens
	\$15.00 / 1M output tokens	\$7.50 / 1M output tokens
gpt-4o-2024-05-13	\$5.00 / 1M input tokens	\$2.50 / 1M input tokens
	\$15.00 / 1M output tokens	\$7.50 / 1M output tokens

**Slika 2.6:** Sustav naplate korištenje GPT-4o putem tokena

Detaljan pregled i objašnjenje načina povezivanja bit će objašnjen u jednom od idućih poglavlja.

## 3. Hardverski dio izvedbe sustava

U narednom poglavlju bit će dan detaljan pregled hardverskih modula koji su bili uzeti u obzir za realizaciju ugradbenog računalnog sustava koji podržava glasovnu interakciju s ChatGPT-om.

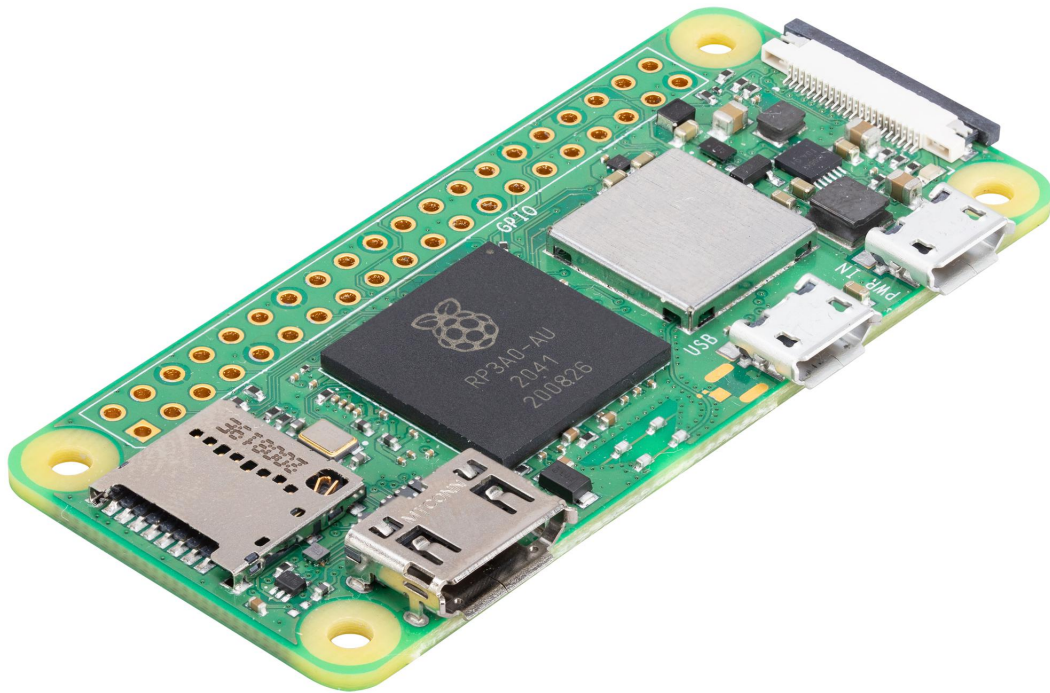
### 3.1. Inicijalna ideja i problemi

Kao što je spomenuto u uvodu, inicijalna ideja rada je uključivala izvedbu sustava koristeći minimum potrebnog hardvera i module male cijene za izvedbu istog. Prvi modul koji je bio razmotren i isproban je Raspberry Pi Zero W u kombinaciji s ReSpeaker 2-Mics Pi HAT dodatkom.

#### 3.1.1. Raspberry Pi Zero W i ReSpeaker Pi HAT

Raspberry Pi Zero W je mikroprocesor razvijen od tvrtke Raspberry Pi Foundation i lansiran 2017. godine kao poboljšana verzija Raspberry Pi Zero modela, ugrađujući mu mogućnosti za bežično povezivanje. Prikazan je na slici 3.1 (bez zalemljenih headera). Sastoji se od Broadcom BCM2835 procesora s ARM11 32-bitnom jezgrom i radi na 1 GHz i 512 MB radne memorije LPDDR2 SDRAM. Nudi mogućnosti bežičnog povezivanja na 802.11 b/g/n Wi-Fi i Bluetooth 4.1, kao i Bluetooth Low Energy (BLE). Od portova za povezivanje periferija ima:

- Mini HDMI port za video izlaz maksimalne rezolucije 1080p pri 60 fps
- Micro USB 5W port za napajanje
- Micro USB port za podatke
- 40 GPIO pinova
- CSI kamera konektor



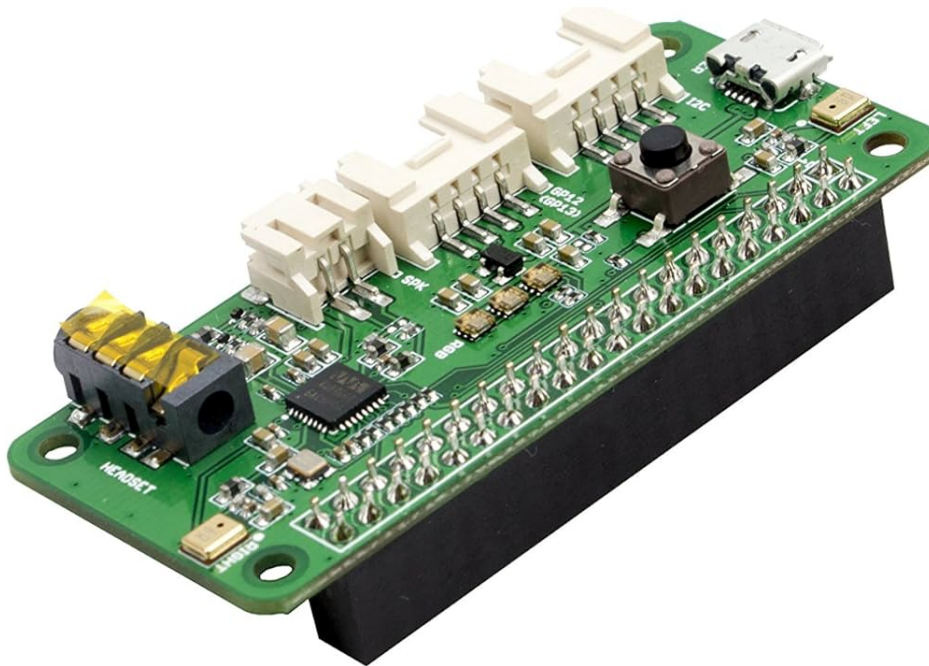
**Slika 3.1:** Raspberry Pi Zero W

Sadrži i utor za MicroSD memorijsku karticu koja služi za pohranu podataka i operacijskog sustava. Malih je dimenzija(65 mm x 30 mm x 5 mm) i niske cijene(oko 15€). Kao takav je idealan za embedded sustave i Internet of Things(u daljnjem tekstu IoT) projekte zbog svoje kompaktnosti i niske potrošnje energije. Ograničavajući faktor je samo jedan podatkovni port, pa ako se žele spojiti dodatne periferije poput tipkovnice i miša, potrebno je dodati razdjelnik. Za zahtjevnije aplikacije su ograničenja performanse zbog procesora i RAM-a. Što se operacijskog sustava tiče, podržava razne Linux distribucije prilagođene ARM arhitekturi kao Raspberry Pi OS, Ubuntu i druge. Također se mogu koristiti brojni programski jezici za razvoj aplikacija poput Pythona, C/C++, Jave itd.

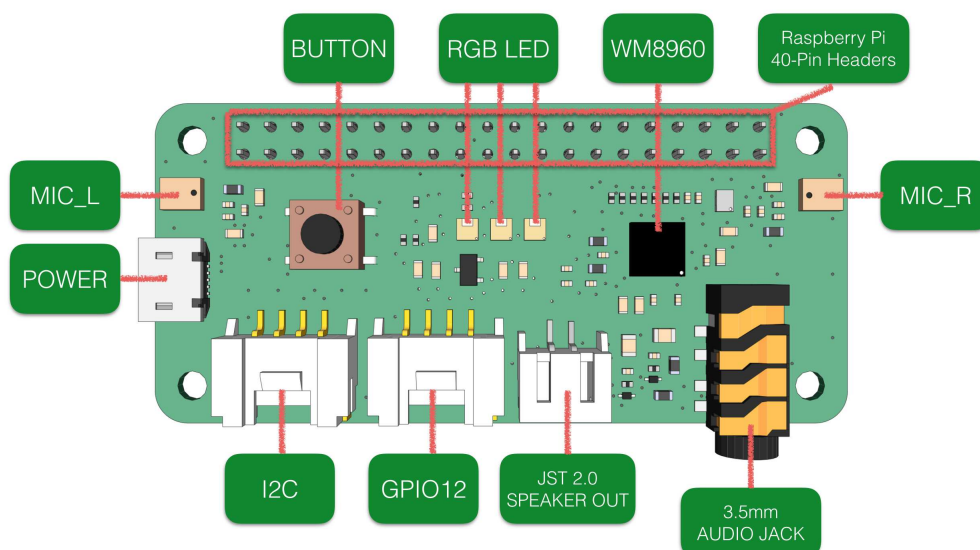
ReSpeaker 2-Mics Pi HAT tvrtke Seeed Studio je dvo-mikrofonski *expansion board* za Raspberry Pi dizajniran za AI i glasovne primjene. Ima slijedeće specifikacije:

- 2 analogna mikrofona
- WM8960 audio kodek male snage
- 3 programibilne APA102 RGB LED diode
- Taktilni korisnički gumb
- Izravno spajanje na Raspberry Pi
- Kompatibilnost s mnogim Raspberry Pi modelima
- 3.5 mm Jack ili JST 2.0 audio izlazi
- Maksimalna frekvencija otipkavanja: 48 kHz

HAT(engl. *Hardware Attached on Top*) na obje strane ima analogne mikrofone koji primaju govorni signal i šalju ga na kodek, gdje su integrirani analogno-digitalni konverteri i pojačalo za mikrofonski signal. Kodek nadalje šalje podatke Raspberry Pi-u preko I2S komunikacije u standardnom I2S ili PCM formatu. Na slici 3.2 prikazan je ReSpeaker HAT, a na slici 3.3 pregled hardverskih komponenti na njemu.



Slika 3.2: ReSpeaker 2-Mics Pi HAT

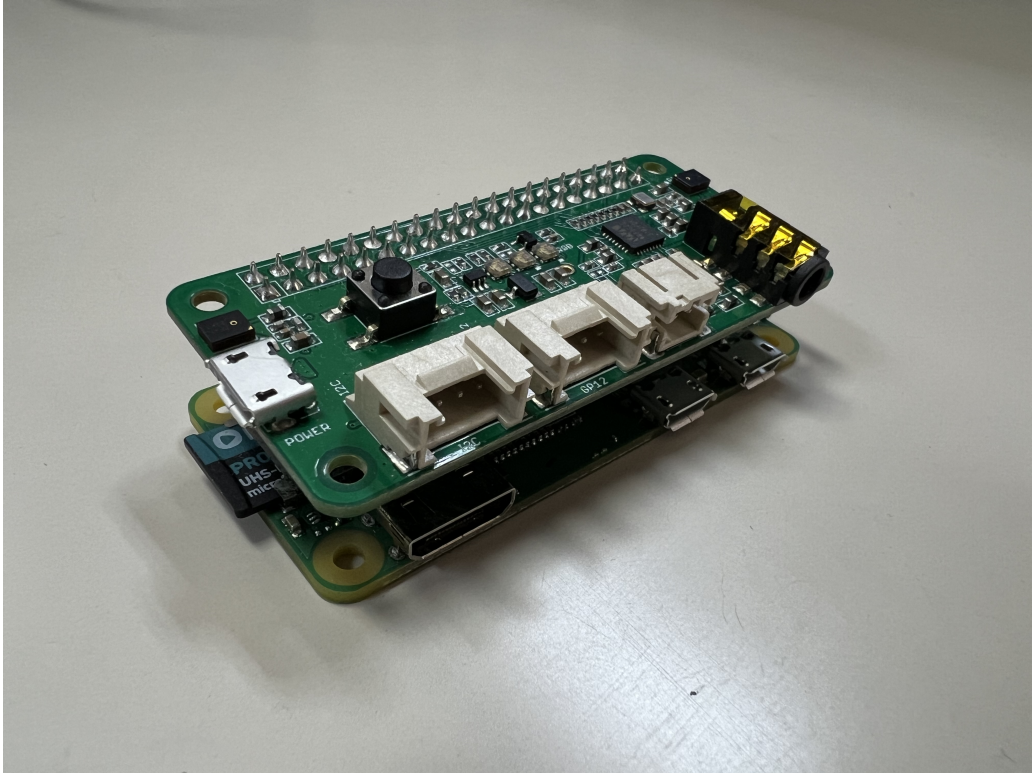


Slika 3.3: Raspored komponenti na HAT-u

S obzirom da su dimenzije modula jednake dimenzijama Raspberry Pi Zero W mikroprocesora, i da je modul napravljen upravo za primjene koje uključuju glasovnu interakciju



i implementaciju asistenata temeljenih na umjetnoj inteligenciji, njihova se kombinacija za potrebe ovog rada činila dobrim izborom. Na slici 3.4 je tu kombinaciju moguće vidjeti. Međutim, prilikom konfiguracije i pokušaja spajanja s Raspberry Pi Zero W-om, izveden je zaključak da novije verzije operacijskog sustava nemaju podršku za HAT.



**Slika 3.4:** Raspberry Pi Zero W s ReSpeaker 2-Mics Pi HAT-om

Na službenim stranicama proizvođača postoje upute kako konfigurirati i instalirati drivere na Raspberry Pi-ju: prvi korak je klonirati GitHub repozitorij i pokrenuti instalaciju putem Bourne shell skripte, a drugi provjeriti prepoznaje li Raspberry Pi zvučnu karticu za snimanje ili reproduciranje zvuka. Na slikama 3.5 i 3.6 su prikazane snimke zaslona Linux terminala koji izbacuje greške pri pokretanju instalacijske skripte.

```
Building module:
Cleaning build area...
make -j1 KERNELRELEASE=6.6.31+rpt-rpi-v6 -C /lib/modules/6.6.31+rpt-rpi-v6/build M=/var/lib/dkms/seeed-voicecard/0.3/build...
.....(bad exit status: 2)
Error! Bad return status for module build on kernel: 6.6.31+rpt-rpi-v6 (armv6l)
Consult /var/lib/dkms/seeed-voicecard/0.3/build/make.log for more information.
Error! One or more modules failed to install during autoinstall.
Refer to previous errors for more information.
dkms: autoinstall for kernel: 6.6.31+rpt-rpi-v6 failed!
run-parts: /etc/kernel/postinst.d/dkms exited with return code 11
dpkg: error processing package linux-image-6.6.31+rpt-rpi-v6 (--configure):
 installed linux-image-6.6.31+rpt-rpi-v6 package post-installation script subprocess returned error exit status 1
Setting up linux-image-6.6.31+rpt-rpi-v7 (1:6.6.31-1+rpt1) ...
/etc/kernel/postinst.d/dkms:
dkms: running auto installation service for kernel 6.6.31+rpt-rpi-v7.
Sign command: /lib/modules/6.6.31+rpt-rpi-v7/build/scripts/sign-file
Signing key: /var/lib/dkms/mok.key
Public certificate (MOK): /var/lib/dkms/mok.pub

Building module:
Cleaning build area...
make -j1 KERNELRELEASE=6.6.31+rpt-rpi-v7 -C /lib/modules/6.6.31+rpt-rpi-v7/build M=/var/lib/dkms/seeed-voicecard/0.3/build...
.....(bad exit status: 2)
Error! Bad return status for module build on kernel: 6.6.31+rpt-rpi-v7 (armv6l)
Consult /var/lib/dkms/seeed-voicecard/0.3/build/make.log for more information.
Error! One or more modules failed to install during autoinstall.
Refer to previous errors for more information.
dkms: autoinstall for kernel: 6.6.31+rpt-rpi-v7 failed!
run-parts: /etc/kernel/postinst.d/dkms exited with return code 11
```

Slika 3.5: Prvi primjer grešaka koji javlja terminal

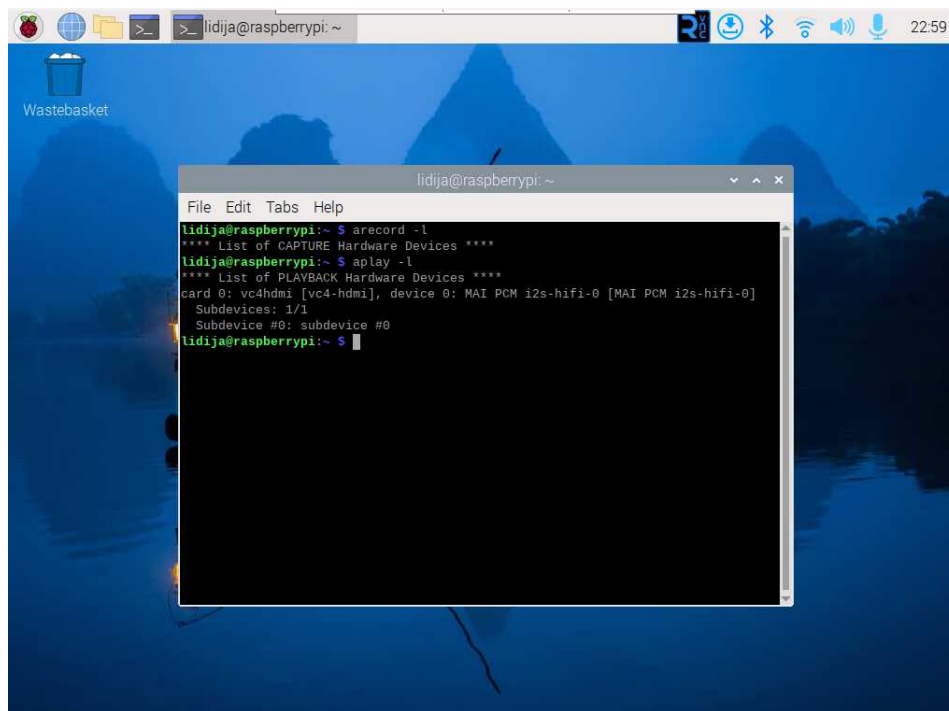
```
dpkg: error processing package linux-headers-rpi-v7 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of linux-headers-rpi-v6:
 linux-headers-rpi-v6 depends on linux-headers-6.6.31+rpt-rpi-v6 (= 1:6.6.31-1+rpt1); however:
 Package linux-headers-6.6.31+rpt-rpi-v6 is not configured yet.

dpkg: error processing package linux-headers-rpi-v6 (--configure):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 linux-image-6.6.31+rpt-rpi-v6
 linux-image-6.6.31+rpt-rpi-v7
 linux-image-rpi-v6
 linux-image-6.6.31+rpt-rpi-v7l
 linux-headers-6.6.31+rpt-rpi-v7l
 linux-image-rpi-v7
 linux-headers-rpi-v7l
 linux-headers-6.6.31+rpt-rpi-v7
 linux-headers-6.6.31+rpt-rpi-v6
 linux-image-rpi-v7l
 linux-headers-rpi-v7
 linux-headers-rpi-v6
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

Slika 3.6: Drugi primjer grešaka koji javlja terminal

Navedene greške su pojavljuju iznova i iznova te terminal završava u beskonačnoj petlji iz koje je moguće izaći jedino zatvaranjem prozora terminala. Po ponovnom pokretanju terminala, naredbama `arecord -l` i `aplay -l` se dobiva popis uređaja kojima Raspberry Pi može snimiti, odnosno reproducirati zvuk. Kao što je vidljivo na slici 3.7, prepoznaje se jedan uređaj za reprodukciju a to je HDMI izlaz - to je ujedno i defaultni audio izlaz. Za audio ulaz se ne prepoznaje niti jedan uređaj, dakle došlo je do pogreške u instalaciji drivera i konfiguraciji općenito. U službenim uputama također stoji da su zadnji put ažurirane 2021. godine što daje naslutiti da trenutne verzije operacijskog sustava ne podržavaju "stare" drivere.





Slika 3.7: Popis audio ulaz/izlaz uređaja

Nakon nekoliko pokušaja debugiranja se odustalo od navedenih modula; od HAT-a iz objašnjenih razloga, a od Raspberry Pi-ja Zero W zbog osjetnih nedostataka u brzini procesora, nemogućnosti pristupa internetu preko Chromiuma te manjka portova za spajanje periferija. Došlo se do zaključka da iako ima svoje prednosti i sigurno bi bio pogodan za neke jednostavnije aplikacije, za konkretne zahtjeve nije dovoljno prikladan ni jednostavan. Umjesto njega je odabran drugi mikroprocesor u istoj porodici uređaja, Raspberry Pi 4B, o kojemu će biti riječ u idućim potpoglavljima.

### 3.1.2. Raspberry Pi Codec Zero

Raspberry Pi Codec Zero (prije poznat pod nazivom IQAudio Codec Zero) je audio add-on modul kompatibilan s bilo kojim Raspberry Pi modelom koji ima 40-pinski GPIO header, a dizajniran je da bude kompatibilan dimenzijama modela Zero W. On omogućuje dvosmjernu komunikaciju digitalnih audio signala između Raspberry Pi-ja i svog kodeka putem I2S-a. Codec Zero ima slijedeće specifikacije:

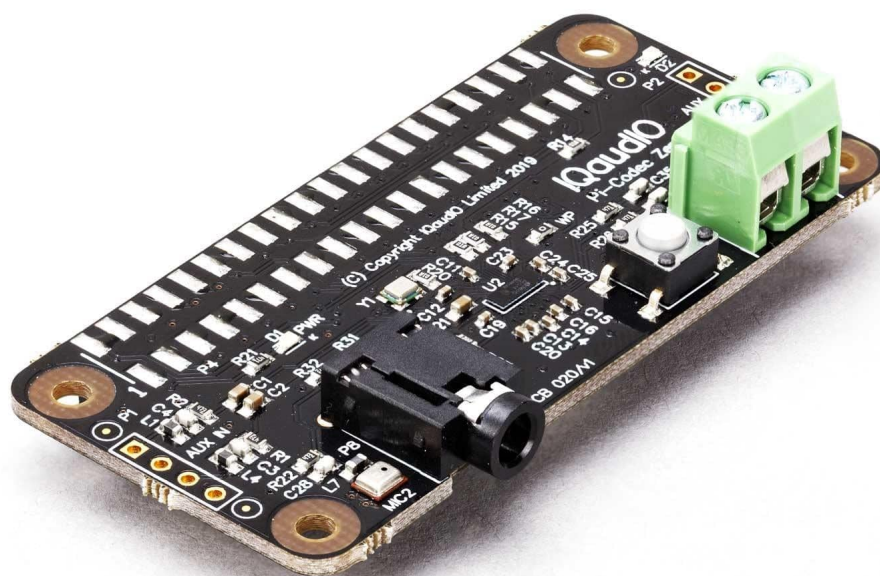
- 24-bitni, 96 kHz Dialog Semiconductor DA7212 digitalni audio kodek (DAC)
- Vanjski izvor napajanja nije potreban, već se napaja preko Raspberry-ovog headera
- LED dioda koja gori kada je priključen na napajanje
- Dodatna zelena i crvena LED dioda za status
- Taktilni korisnički gumb

- *Built-in* mono elektretski i MEMS mikrofoni
- HAT EEPROM memorija

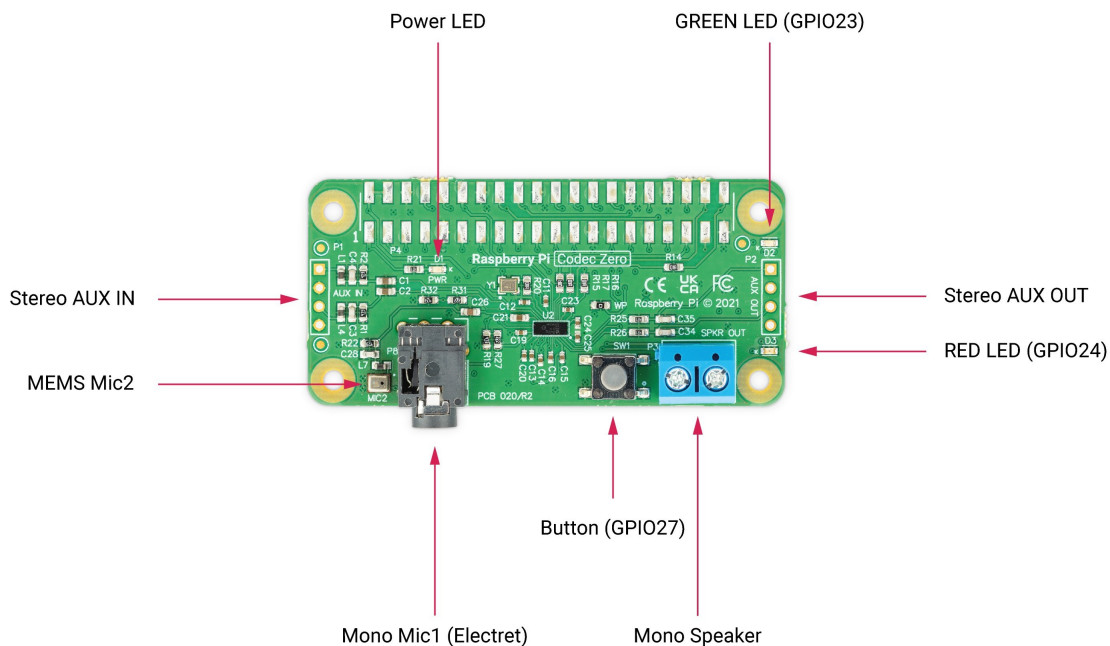
Nudi podršku za:

- Dodatni vanjski mono mikrofoni kroz 3.5 mm ulaz
- 1.2W 8 Ohmski mono zvučnik
- Ulazne i izlazne stereo kanale: AUX IN i AUX OUT pinovi

Kao što se može vidjeti iz priloženog, HAT omogućuje korištenje različitih ulaznih i izlaznih audio uređaja i kao takav je pogodna početna točka za audio projekte poput pametnog zvučnika, *streaming* uređaja itd. Na slici 3.8 prikazana je starija verzija prije rebrandiranja uređaja, a slikom 3.9 trenutna verzija uz označen dio komponenata. Jedine razlike između dvije verzije su naziv i boja pločice, no mehanički i električki je sve ostalo isto.



**Slika 3.8:** IQAudio Codec Zero

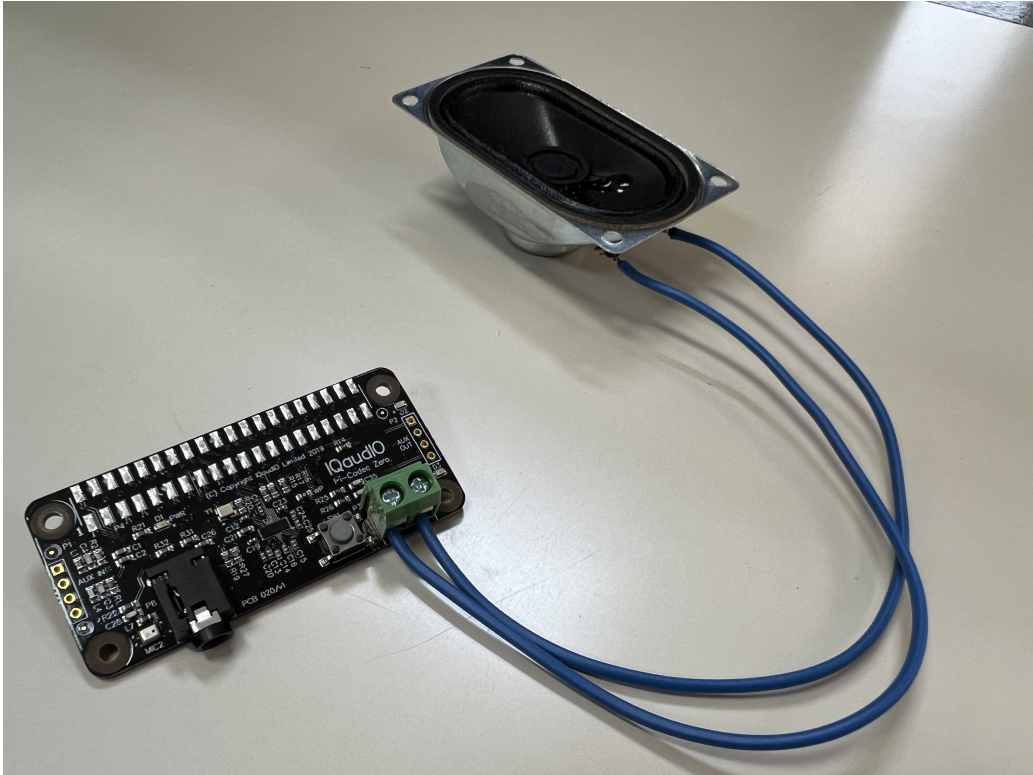


**Slika 3.9:** Raspberry Pi Codec Zero s oznakama komponenti

Opisani HAT modul je zamišljen kao *plug and play* solucija za projekte koji se bave ulaznim i izlaznim audio signalima, međutim je i ovdje došlo do iscrpnih pokušaja utilizacije modula, specifično mikrofona, bez pozitivnog ishoda.

Na slici 3.10 je moguće vidjeti Codec Zero uz dodan vanjski zvučnik, koji je pomoću odstojnika postavljen na Raspberry Pi 4B i prikaz toga je dan na slici 3.11.

Konfiguracija kodeka je napravljena prema uputama u službenoj dokumentaciji, a uključivala je male izmjene `config.txt` datoteke, kloniranje GitHub repozitorija i spremanje `.state` datoteke - predkonfigurirane skripte koja govori ALSA (engl. *Advanced Linux Sound System*) sustavu da za audio ulaz koristi MEMS mikrofona, a za audio izlaz vanjski zvučnik. Po završetku konfiguracije je ustanovljeno da Raspberry Pi 4B prepoznaje kodek kao uređaj koji ima i mikrofona i zvučnik, što je vidljivo iz slike 3.12 kao i na slici 3.13 po rezultatima komandi koje su ranije objašnjene. Zvučna kartica kodeka je i za ulaz i za izlaz prepoznata kao `card 2`. Međutim usprkos tome što ga sustav prepoznaje, zvučnik je radio, dok mikrofona nije. Greška koja se javlja prilikom pokretanja komande `arecord` koja snima zvuk preko mikrofona je prikazana slikom 3.14.



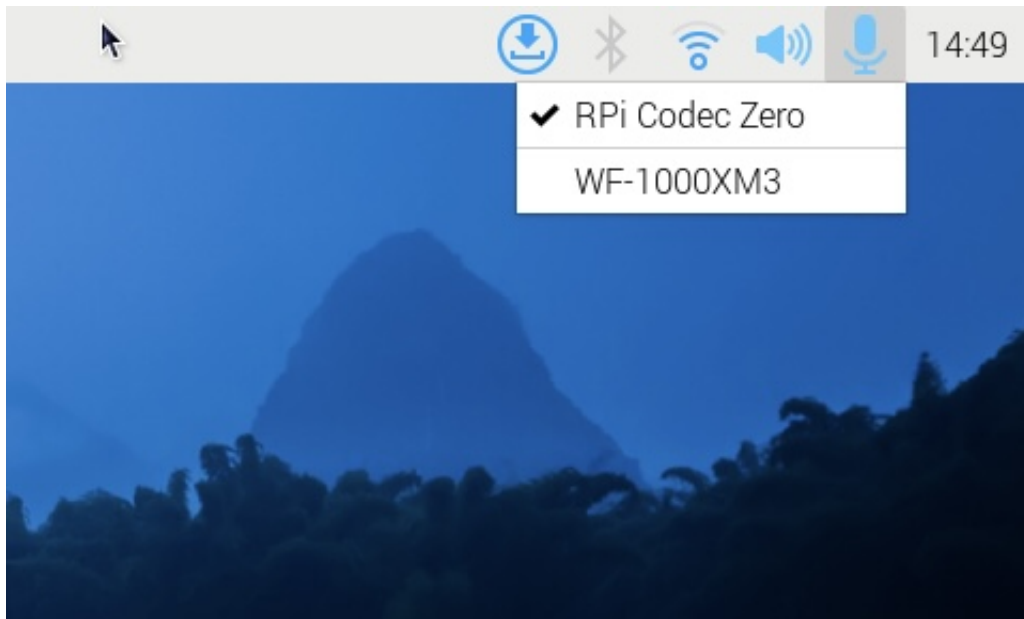
**Slika 3.10:** Codec uz dodan vanjski zvučnik



**Slika 3.11:** Raspberry Pi 4B s Codecom i zvučnikom

Uz dugotrajne pokušaje otklanjanja grešaka što za Raspberry Pi Codec Zero, što za prethodno opisani ReSpeaker 2-Mics Pi HAT modul, zaključeno je da proizvođači nažalost u





Slika 3.12: Odabir audio ulaza

```
lidija@raspberrypi: ~  
File Edit Tabs Help  
lidija@raspberrypi:~ $ arecord -l  
**** List of CAPTURE Hardware Devices ****  
card 2: Zero [RPi Codec Zero], device 0: Raspberry Pi Codec Zero HiFi da7213-hifi-0 [Raspberry Pi Codec Zero HiFi da7213-hifi-0]  
Subdevices: 1/1  
Subdevice #0: subdevice #0  
lidija@raspberrypi:~ $ aplay -l  
**** List of PLAYBACK Hardware Devices ****  
card 0: vc4hdmi0 [vc4-hdmi-0], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]  
Subdevices: 1/1  
Subdevice #0: subdevice #0  
card 1: vc4hdmi1 [vc4-hdmi-1], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]  
Subdevices: 1/1  
Subdevice #0: subdevice #0  
card 2: Zero [RPi Codec Zero], device 0: Raspberry Pi Codec Zero HiFi da7213-hifi-0 [Raspberry Pi Codec Zero HiFi da7213-hifi-0]  
Subdevices: 1/1  
Subdevice #0: subdevice #0  
lidija@raspberrypi:~ $
```

Slika 3.13: Naredbe arecord -l i aplay -l

nekom trenutku prestaju održavati drivere za postojeće proizvode na novijim verzijama operacijskih sustava, pa je prije naručivanja HAT-a preporučljivo pretražiti internet forume i članke kako bi se vidjelo je li ga u danom trenutku i dalje moguće koristiti, po mogućnosti uz što manje utrošenog vremena na otklanjanje grešaka. U slučaju ovoga rada je odlučeno isprobati drugačiji, jednostavniji pristup s USB mikrofonom i zvučnicima, što će biti opisano

```
lidija@raspberrypi: ~  
File Edit Tabs Help  
lidija@raspberrypi:~$ arecord -l  
**** List of CAPTURE Hardware Devices ****  
card 2: Zero [RPi Codec Zero], device 0: Raspberry Pi Codec Zero HiFi da7213-hif  
i-0 [Raspberry Pi Codec Zero HiFi da7213-hifi-0]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
lidija@raspberrypi:~$ aplay -l  
**** List of PLAYBACK Hardware Devices ****  
card 0: vc4hdmi0 [vc4-hdmi-0], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
card 1: vc4hdmi1 [vc4-hdmi-1], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
card 2: Zero [RPi Codec Zero], device 0: Raspberry Pi Codec Zero HiFi da7213-hif  
i-0 [Raspberry Pi Codec Zero HiFi da7213-hifi-0]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
lidija@raspberrypi:~$ arecord -D hw:2,0 -f cd -t wav test11.wav  
Recording WAVE 'test11.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo  
arecord: pcm_read:2221: read error: Input/output error  
lidija@raspberrypi:~$
```

Slika 3.14: Greška pri naredbi arecord

u idućim potpoglavljima.

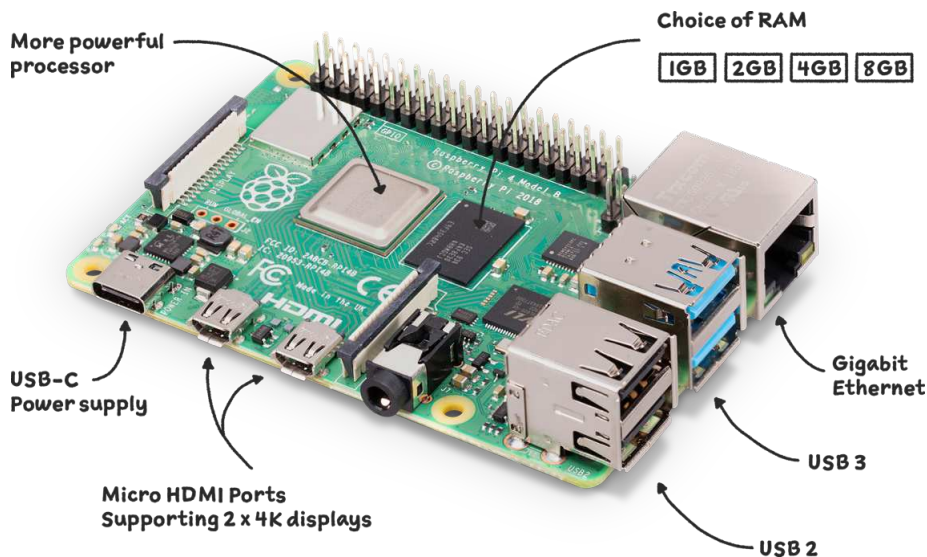
## 3.2. Raspberry Pi 4B, USB mikrofoni i zvučnici

Raspberry Pi 4 Model B je popularno i moćno *single-board* računalo dizajnirano za širok spektar primjena. U nastavku je dan pregled mogućnosti i tehničkih specifikacija:

- CPU - Broadcom BCM2711, Quad-core Cortex-A72(ARM v8) 64-bitni SoC frekvencije 1.8 GHz
- Radna memorija - 1, 2, 4 ili 8GB LPDDR4-3200 SDRAM
- 2.4 GHz i 5.0 GHz IEEE 802.11ac Wi-Fi
- Povezivanje na Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 2.0 porta i 2 USB 3.0 porta
- Standardni 40-pinski GPIO header
- 2 micro HDMI porta
- MIPI DSI display i CSI kamera portovi
- Četveropolni stereo audio i video port

– Napajanje 5V DC preko USB-C konektora

Kao što je vidljivo iz opisanog, fleksibilan je i svestran alat koji je namijenjen raznim korisnicima i za širok raspon primjena, npr. kao centralna jedinica za IoT projekte, za razne hobističke projekte, *smart home* sustave, kao i industrijske aplikacije i razvoj novih proizvoda. Za ovaj rad je odabrana verzija s 8GB radne memorije koja je prikazana slikom 3.15.



Slika 3.15: Raspberry Pi 4 Model B s oznakama pojedinih komponenti

Za potrebe ovog rada odabrana je verzija s 8GB radne memorije što se pokazalo i više nego dovoljno, a sam model Raspberry Pi-ja je povoljan izbor upravo zbog velike procesorske moći i raznovrsnih mogućnosti koje nudi.

Ugradbeni računalni sustav je u konačnici izveden pomoću njega, jednostavnog mikrofona i zvučnika koji su prikazani slikama 3.16 i 3.17.

Konkretno je korišten mikروفон Primo Desk tvrtke Trust, koji je svojim 3.5 mm konektorom preko dodatnog adaptera spojen na USB utor Raspberry Pi-ja, a stereo zvučnici su direktno spojeni na A/V izlaz preko 3.5 mm konektora. Njima je potrebno dovesti vanjsko napajanje za rad. Cijeli prikaz hardverskog dijela sustava dan je slikom 3.18.



**Slika 3.16:** Mikrofon



**Slika 3.17:** Zvučnici





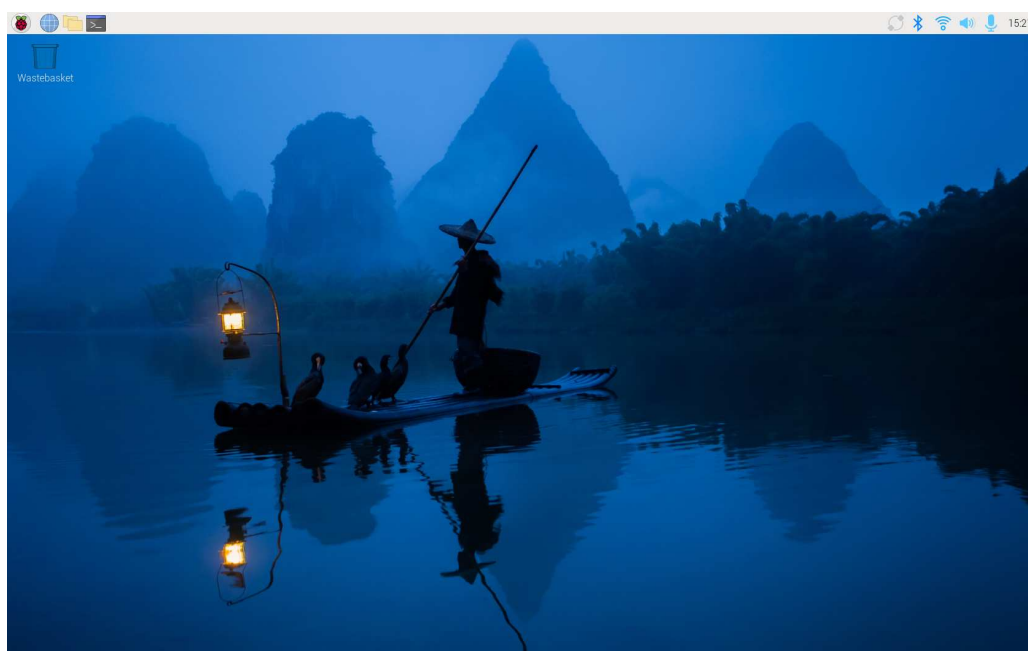
**Slika 3.18:** Finalna izvedba sustava

## 4. Softverski dio izvedbe sustava

### 4.1. Operacijski sustav

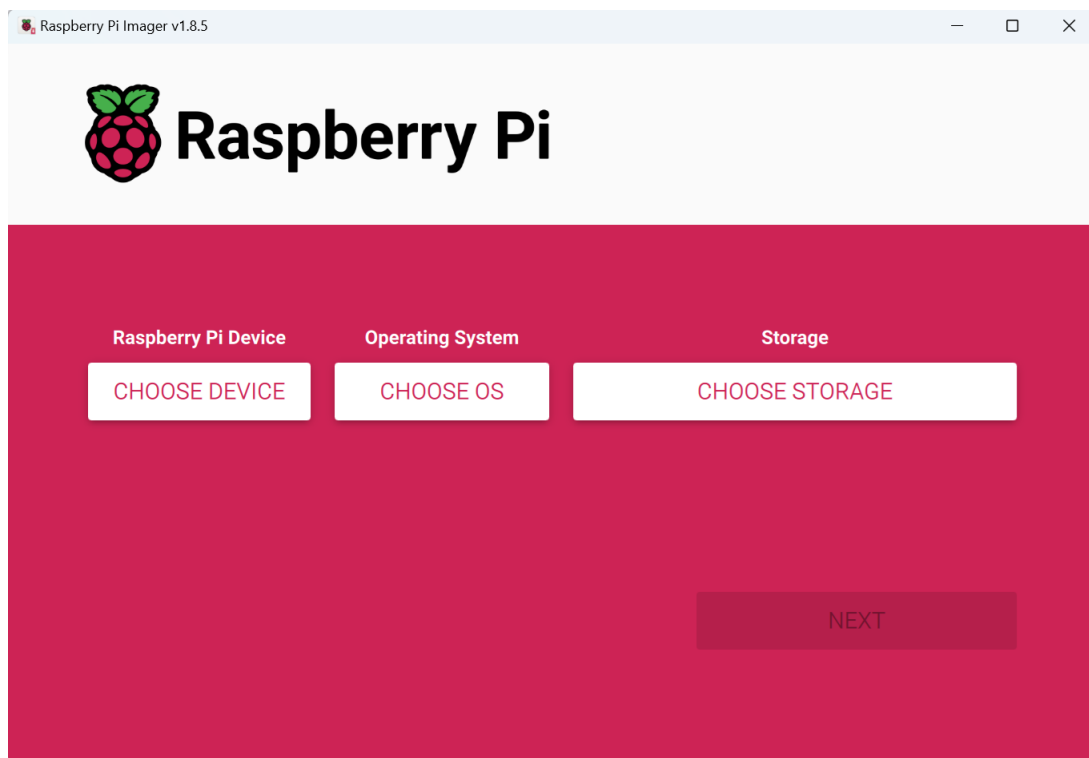
Operacijski sustav(u daljnjem tekstu OS) korišten za potrebe ovog rada je Raspberry Pi OS. On je temeljen na Debian distribuciji Linuxa i podržava preko 35 000 Debian paketa, a preporuča se za korištenje na gotovo svim Raspberry Pi modelima. Najnovija verzija OS-a je temeljena konkretno na Debian Bookworm distribuciji, dok su prethodne bile na Debian Bullseye-u.

Korisničko sučelje OS-a sadrži desktop okruženje zvano PIXEL(kratica za engl. *Pi Improved Xwindows Environment Lightweight*) koje izgleda slično kao mnogi češće korišteni desktopovi, primjerice macOS i Microsoft Windows. Alatna traka je pozicionirana na vrh ekrana i sadrži aplikacijski menu te prečice za web pretraživač(Chromium), file manager i terminal. Na drugom kraju trake se nalaze ikone za Bluetooth, Wi-Fi i audio input/output menu. Pakete je preporučeno instalirati, nadograditi ili micati pomoću APT-a(kratica za engl. *Advanced Package Tool*). Izgled početnog desktopa prikazan je slikom 4.1.



Slika 4.1: Raspberry Pi početni zaslon

S obzirom da Raspberry Pi računala nemaju integriranu ROM memoriju, OS je potrebno instalirati na microSD memorijsku karticu koju je zatim potrebno umetnuti u odgovarajući utor kako bi se uređaj bootao. Termin koji se ovdje uglavnom koristi je napraviti imaging SD kartice, što znači kreirati identičku kopiju file sistema OS-a i kopirati i napisati ju na karticu. Postoji više načina kako se OS može instalirati, no najjednostavniji i preporučeni način je koristeći Raspberry Pi Imager. To je softverski alat koji pomaže u preuzimanju i pisanju slika na macOS, Windows ili Linux, a uključuje mnoge slike OS-ova za Raspberry Pi. Pokretanjem Imager-a se otvara prozor prikazan slikom 4.2, gdje se u prvom koraku odabire uređaj, OS i memorija. Nadalje je moguće prilagoditi OS svojim potrebama prije prvog boota i predkonfigurirati informacije poput korisničkog imena i lozinke, Wi-Fi detalja, vremenske zone, tipkovnice, udaljenog spajanja i slično. Proces image-anja je gotov kroz nekoliko minuta i uređaj je spreman za prvi boot.

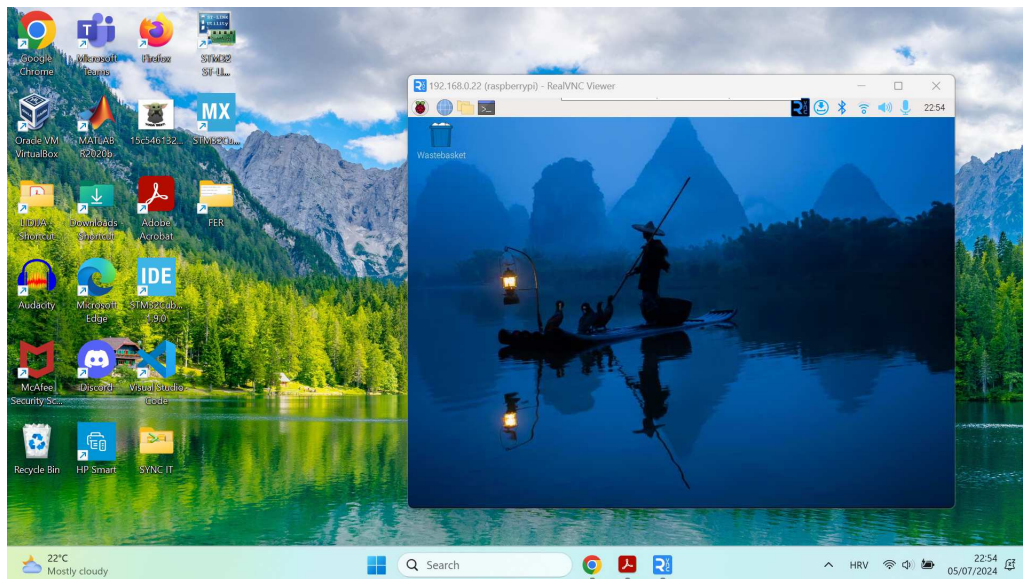


**Slika 4.2:** Raspberry Pi Imager

Nadalje, jednom kad je OS instaliran i prvi boot obavljen, na Raspberry Pi se moguće spojiti na dva načina: klasično, korištenjem periferija poput monitora, miša i tipkovnice, i udaljeno(u daljnjem tekstu engl. *remote*). Oba pristupa imaju svoje prednosti i mane, za potrebe ovog rada korištena su oba načina povezivanja, a u nastavku će ukratko biti opisan potonji.

*Remote* povezivanje, odnosno korištenje Raspberry Pi-ja u *headless* varijanti(bez korištenja periferija), podrazumijeva upravljanje preko drugog računala. Za to se koristi VNC protokol(engl. *Virtual Network Computing*), a OS već ima predinstaliran program RealVNC

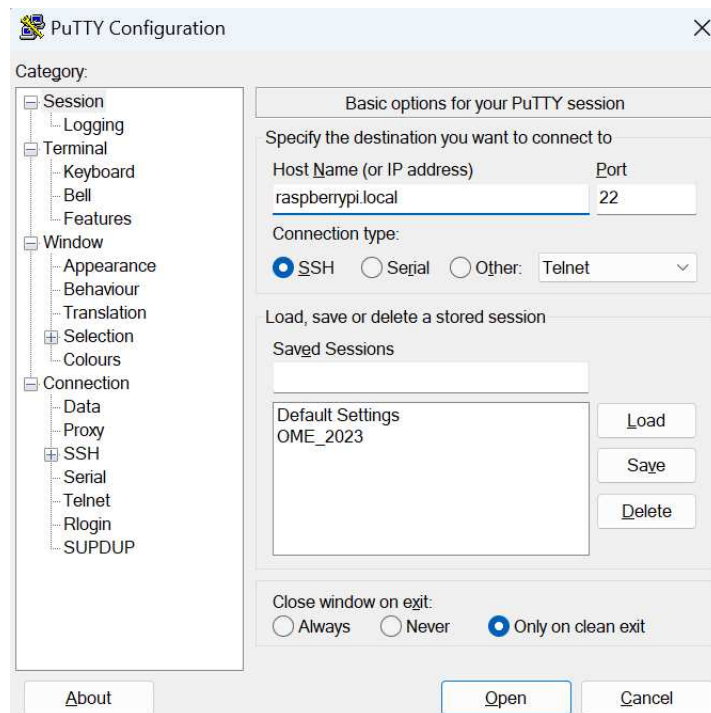
kojega je onda potrebno instalirati i na računalo s kojega se želi upravljati Raspberry-jem i povezati korištenjem njegove IP adrese te korisničkog imena i lozinke. Kako izgleda takav tip spajanja na, u ovom slučaju osobnom laptopu, prikazano je na slici 4.3.



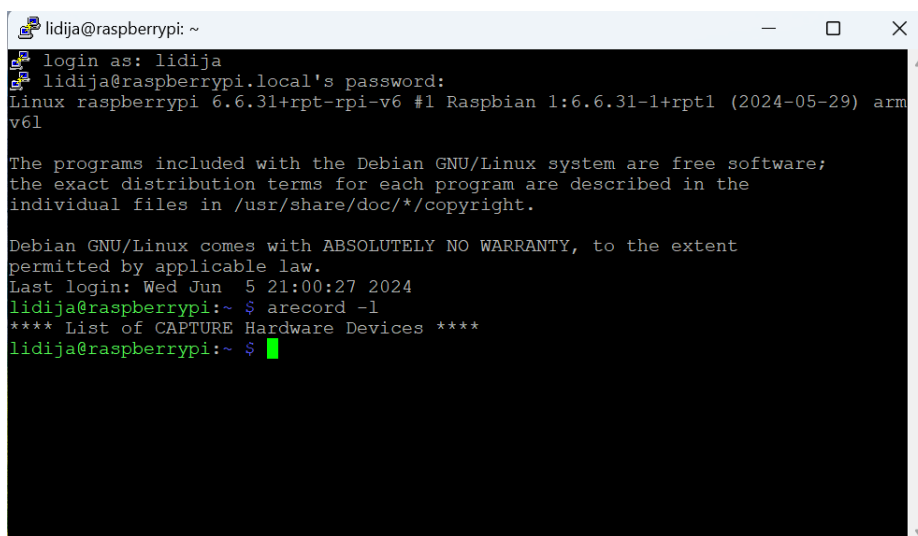
**Slika 4.3:** Prikaz zaslona laptopa i VNC aplikacije

Međutim, ako se Raspberry Pi želi koristiti *headless* i prije prvog boota, potrebno je napraviti još nekoliko koraka prije no što je moguće povezivanje preko VNC aplikacije. Za potrebe ovog rada, ovakav je pristup bio potreban zbog naknadne nabave periferija. Prvo se potrebno pomoću SSH(engl. *Secure Shell*) protokola prijaviti i povezati s terminalom, što je napravljeno uz PuTTY - *open-source* emulator terminala. Proces je prikazan slikama 4.4 i 4.5.

Nadalje je putem naredbe `sudo raspi-config`, koja daje prikaz postavki Raspberry Pi OS-a, potrebno omogućiti VNC pregled. Tek nakon toga je VNC viewer moguće upogoniti.



Slika 4.4: PuTTY



Slika 4.5: Emulirani Raspberry Pi terminal

## 4.2. Interakcija s ChatGPT-om

Programski kod korišten za komunikaciju s ChatGPT-om pisan je u programskom jeziku Python, a korištena je programska okolina Thonny. Thonny je besplatan i *open-source* IDE za Python te je dizajniran za početnike, a na Raspberry Pi OS-u također dolazi predinstaliran.

Prvi korak je učitavanje potrebnih Python biblioteka koje su dane u odsječku 4.1.

```
1 import speech_recognition as sr
```

```

2 import pyttsx3
3 import warnings
4 import sys
5 import keyboard
6 import time
7 import openai
8 from openai import OpenAI
9 import os
10 import platform

```

**Listing 4.1:** Učitavanje Python biblioteka

Odsječkom 4.2 dan je idući dio koda. Prvo, kako bi se omogućilo korištenje OpenAI biblioteke, potrebno je izraditi API ključ preko korisničkog računa na službenim stranicama. On služi za autorizaciju i autentifikaciju prilikom pristupa OpenAI API-ju, a može biti korišten za jedan ili više projekata. Prilikom izrade se on korisniku daje samo jednom, uz naputak da ga se kopira i spremi na sigurno mjesto kako nebi došlo do zlouporabe ključa. U liniji 2 ga je potrebno spremiti u varijablu naziva OPENAI\_KEY. Nadalje, u liniji 5 se inicijalizira prepoznavanje govora uz klasu Recognizer iz `speech_recognition` biblioteke.

Ostatak danog koda obavlja postavljanje error handlera za ALSA biblioteku. Prvo što se provjerava je vrti li se kod na Linuxu, a zatim se učitavaju Python biblioteke za definiranje C funkcija i daje se definicija funkcije CFUNCTYPE. Ona odgovara potpisu ALSA error handlera. Nakon toga se opisuje Python error handler koji upravlja ALSA greškama te se on konvertira u pokazivač na C-kompatibilnu funkciju. Na kraju se učitava ALSA biblioteka `asound`, i iz nje se postavlja prilagođeni error handler za greške koje javlja ALSA.

```

1 # Set your OpenAI API key
2 OPENAI_KEY = "sk-proj-tRDVZh4WeADVyqg8jmlVT3BlbkFJrvOFiE49HpbctLqzZbEw"
3 client = OpenAI(api_key=OPENAI_KEY)
4
5 recognizer = sr.Recognizer()
6 if platform.system() == 'Linux':
7     from ctypes import CFUNCTYPE, c_char_p, c_int, cdll
8     error_handler = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int,
9                               c_char_p)
10
11     def py_error_handler(filename, line, function, err, fmt):
12         pass
13
14     c_error_handler = error_handler(py_error_handler)
15     asound = cdll.LoadLibrary('libasound.so')
16     asound.snd_lib_error_set_handler(c_error_handler)

```

**Listing 4.2:** Postavljanje API ključa i error handler



U odsječku 4.3 dana je definicija funkcije `recording` koja kontinuirano čita audio ulaz, prilagođava ga ambijentalnoj buci radi poboljšanja točnosti prepoznavanja i pomoću recognizer-a osluškuje mikrofona. Kada se audio detektira s mikrofona, pokušava se prepoznati što je izgovoreno slanjem signala u oblak putem Google Web Speech API-ja nakon čega se prepoznati govor ispisuje na ekranu. Ako prepoznavanje nije bilo uspješno, ispisuje se poruka koja korisnika obavještava o tome i pita ga da pokuša ponovno.

```
1 def recording():
2     while True:
3         try:
4             with sr.Microphone() as mic:
5                 recognizer.adjust_for_ambient_noise(mic, duration=0.2)
6                 print("Listening...")
7                 audio = recognizer.listen(mic)
8                 text = recognizer.recognize_google(audio).lower()
9                 print(f"Recognized: {text}")
10                return text
11        except sr.UnknownValueError:
12            print("Sorry, I did not understand that. Please try again.")
```

**Listing 4.3:** Funkcija za obradu audio inputa

Definicija funkcije `send_to_GPT` nalazi se u odsječku 4.4 i u njoj se koristi funkcija OpenAI API-ja s parametrima:

- `model` - koji se model GPT-ja koristi, u ovome slučaju je to drugi najnoviji dostupni modela, GPT-3.5-turbo
- `messages` - povijest korisničkih upita kao i odgovora na iste
- `max_tokens` - granica maksimalnog broja tokena koje odgovor može sadržavati, koristi se kako bi se regulirala potrošnja
- `n` - specificira koliko se completionsa može generirati
- `stop` - koje sekvence zaustavljaju generiranje odgovora
- `temperature` - kontrola slučajnosti izlaza modela, manja vrijednost poput zadane 0.5 osigurava veći determinizam izlaza

Funkcija vraća varijablu `message` u koju se sprema generirani odgovor GPT-ja.

```
1 def send_to_GPT(messages, model="gpt-3.5-turbo"): # Use correct model
2     name
3     response = client.chat.completions.create(
4         model=model,
```

```

4     messages=messages,
5     max_tokens=100,
6     n=1,
7     stop=None,
8     temperature=0.5
9 )
10 message = response.choices[0].message.content
11 return message
12
13 messages = []

```

**Listing 4.4:** Funkcija za slanje upita na GPT

Odsječak 4.5 prikazuje glavni dio koda gdje se osluškuje mikrofoni, snimljeni audio šalje na prepoznavanje, te se vraća prepoznati govor u tekstualnom obliku i ispisuje na zaslonu, kao što je opisano u definiciji funkcije 4.3. Zatim se korištenjem funkcije `send_to_GPT` tekst šalje GPT-ju i vraća se odgovor također u tekstualnom obliku koji se ispisuje na zaslonu.

Nakon toga slijedi inicijalizacija TTS-a koristeći `pyttsx3` biblioteku te se daju postavke poput jezika i brzina izgovora u riječima po minuti. Naredbama u linijama koda 16 i 17 se odgovor postavlja u red za sintezu u govor, red se obrađuje i TTS engine sintetizira govor kojeg je moguće čuti kroz audio izlaz Raspberry Pi-ja(zvučnike).

```

1 while True:
2     try:
3         with sr.Microphone() as mic:
4             recognizer.adjust_for_ambient_noise(mic, duration=0.2)
5             print("Listening...")
6             audio = recognizer.listen(mic)
7             text = recognizer.recognize_google(audio, language="sr-SP").
lower()
8             print(f"Recognized: {text}")
9             messages.append({"role": "user", "content": text})
10            response = send_to_GPT(messages)
11            messages.append({"role": "assistant", "content": response})
12            print(f"Response: {response}")
13            engine = pyttsx3.init()
14            engine.setProperty(name='voice', value='croatian')
15            engine.setProperty(name='rate', value=140)
16            engine.say(response)
17            engine.runAndWait()
18        except sr.UnknownValueError:
19            print("Sorry, I did not understand that. Please try again.")

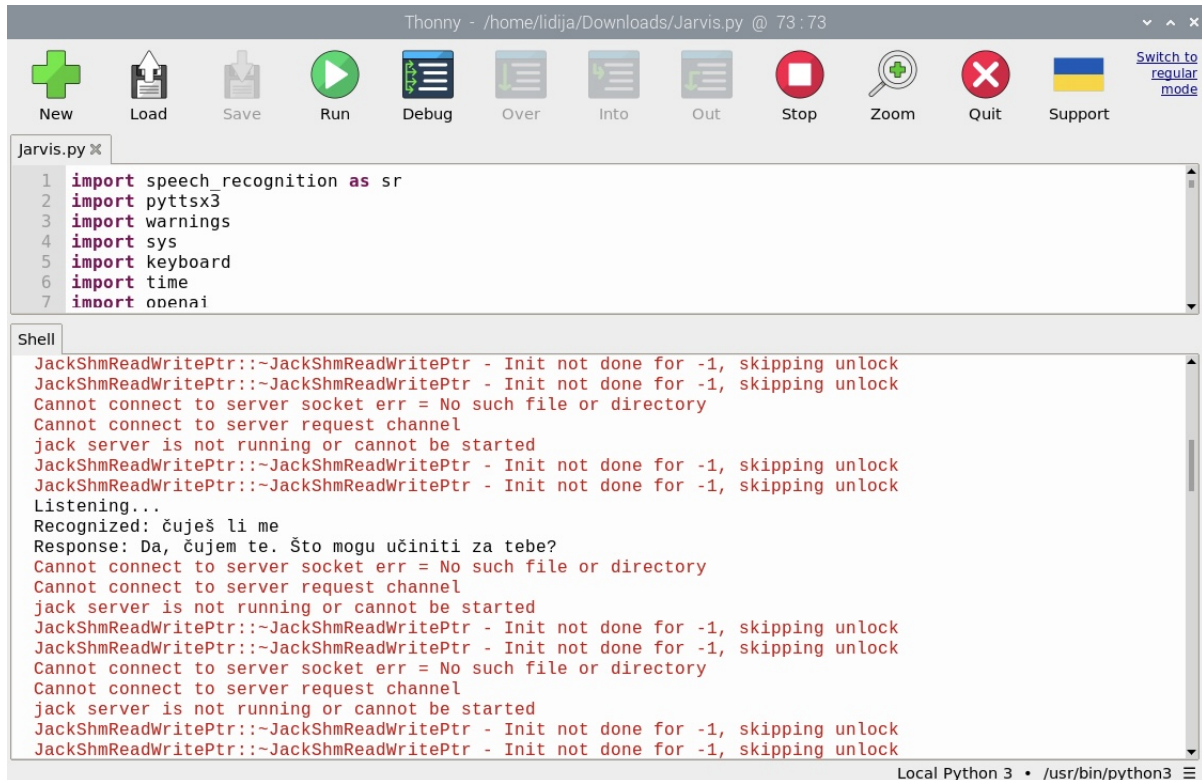
```

**Listing 4.5:** Glavni program



Opisana Python skripta izvršava funkcionalnost glasovne interakcije s GPT modelom, postavljenu na hrvatski jezik (prepoznavanje preko Google Web Speech API-ja je postavljeno na srpski jezik jer hrvatski nije radio).

Na slikama 4.6 i 4.7 prikazani su izlazi Thonny IDE-a prilikom pokretanja opisane skripte. Errori koji se pojavljuju ne utječu na pravilno izvođenje skripte pa su zanemarivi.



```
Thonny - /home/lidija/Downloads/Jarvis.py @ 73:73
New Load Save Run Debug Over Into Out Stop Zoom Quit Support
Jarvis.py
1 import speech_recognition as sr
2 import pyttsx3
3 import warnings
4 import sys
5 import keyboard
6 import time
7 import openai

Shell
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Listening...
Recognized: čuješ li me
Response: Da, čujem te. Što mogu učiniti za tebe?
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Local Python 3 • /usr/bin/python3
```

Slika 4.6: Izlaz skripte

```
Thonny - /home/lidija/Downloads/Jarvis.py @ 73 : 73
New Load Save Run Debug Over Into Out Stop Zoom Quit Support Switch to regular mode

Jarvis.py
1 import speech_recognition as sr
2 import pyttsx3
3 import warnings
4 import sys
5 import keyboard
6 import time
7 import oenai

Shell
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Listening...
Recognized: što je raspberry pi
Response: Raspberry Pi je mali, pristupačan jednodimenzionalni računalni sustav koji je popularan među hobi
stima, inženjerima, edukatorima i entuzijastima za tehnologiju. Raspberry Pi je razvijen u Velikoj Britan
iji kao jeftina platforma za učenje programiranja i eksperimentiranje s elektronikom. Ima sve pot
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Local Python 3 • /usr/bin/python3
```

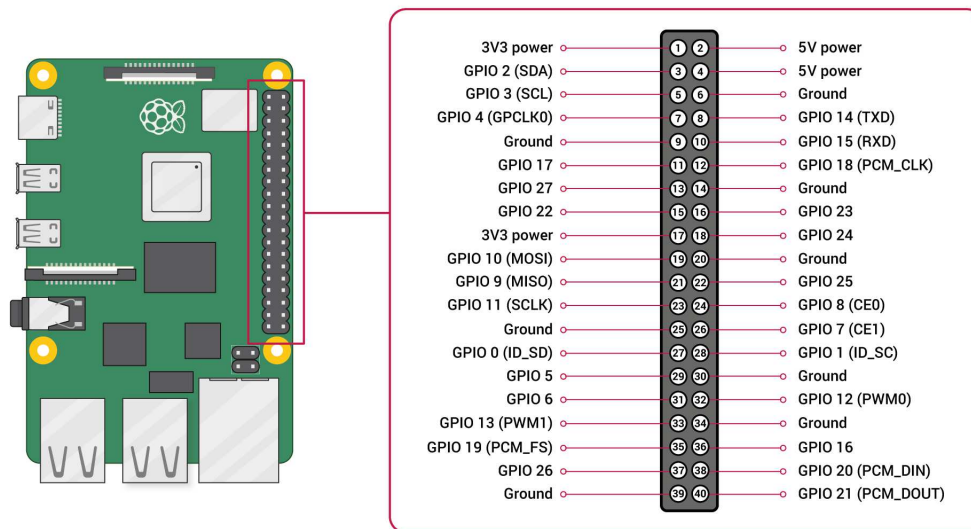
Slika 4.7: Izlaz skripte

### 4.3. Dodavanje LED diode

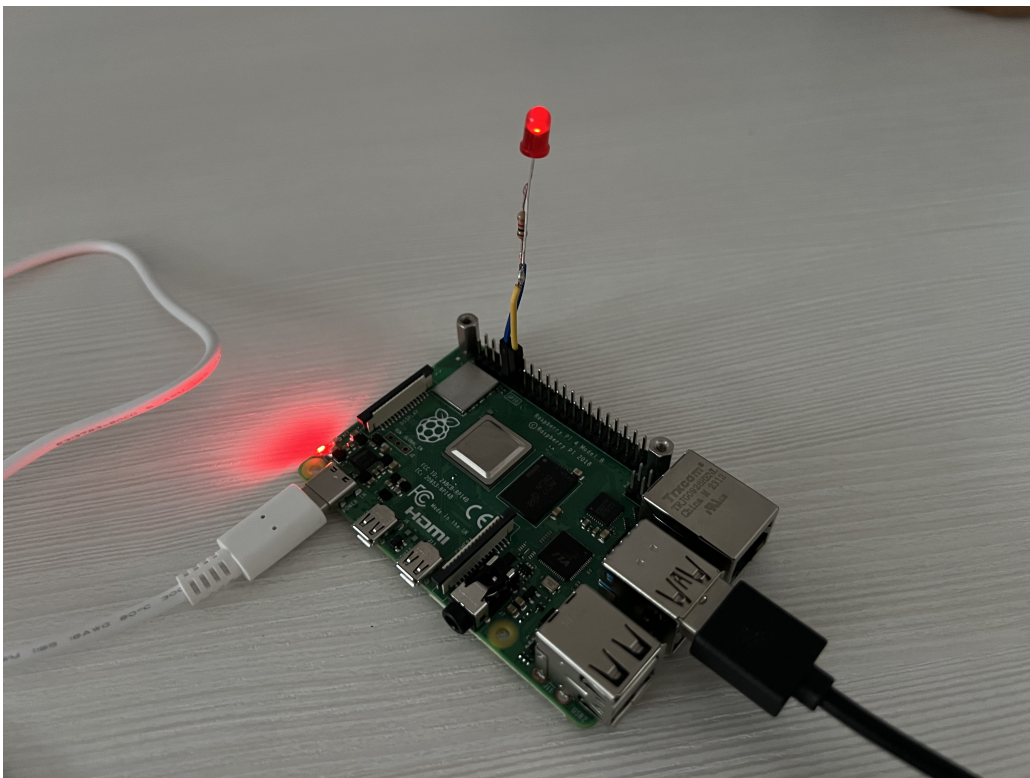
U cilju proširenja mogućnosti opisanog softverskog rješenja, koje u ovom trenutku implementira glasovnu interakciju s ChatGPT-om, kako bi u konačnici rješenje bilo sličnije funkcijama prvotno zamišljene Alexe, dodana je LED dioda. To je jednostavan način ilustracije mogućnosti koje se naknadno mogu ugraditi u program, drugim riječima dokaz da Raspberry Pi može poduzeti određenu akciju ako ga se isprogramira na određeni način.

S obzirom da Raspberry Pi 4B interno ima samo dvije LED diode od kojih je jedna za napajanje, a druga statusna, bilo je potrebno eksterno dovesti LED diodu na GPIO pinove. GPIO pinout Raspberry Pi-ja 4B dan je na slici 4.8. Crvena LED dioda u seriji s otpornikom od 1 kΩ je spojena na GPIO17(pin broj 11) i masu(pin broj 9), kao što je vidljivo na slici 4.9.

Programsko upravljanje LED diodom je izvedeno na način da se u transkriptu govornog upita traži ključna riječ "upali" te ako je ona detektirana, GPIO17 pin se postavlja u stanje HIGH što rezultira paljenjem LED diode u trajanju od 3 sekunde, nakon čega se ona gasi. Ako ključna riječ nije detektirana, ispisuje se sukladna poruka i LED dioda se ne pali. Odsječak 4.6 sadrži finalni kod, a na slici 4.10 je prikazano paljenje LED diode.



Slika 4.8: GPIO pinout Raspberry Pi 4B modela



Slika 4.9: Spajanje i paljenje LED diode

```

1 import speech_recognition as sr
2 import pyttsx3
3 import warnings
4 import sys
5 import keyboard

```

```

6 import time
7 import openai
8 from openai import OpenAI
9 import os
10 import platform
11 import RPi.GPIO as GPIO
12
13 # Set your OpenAI API key
14 OPENAI_KEY = "sk-proj-tRDVZh4WeADVygg8jmlVT3BlbkFJrvOFiE49HpbctLqzZbEw"
15 client = OpenAI(api_key=OPENAI_KEY)
16
17 recognizer = sr.Recognizer()
18 if platform.system() == 'Linux':
19     from ctypes import CFUNCTYPE, c_char_p, c_int, cdll
20     error_handler = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int,
21                               c_char_p)
22
23     def py_error_handler(filename, line, function, err, fmt):
24         pass
25
26     c_error_handler = error_handler(py_error_handler)
27     asound = cdll.LoadLibrary('libasound.so')
28     asound.snd_lib_error_set_handler(c_error_handler)
29
30 def send_to_GPT(messages, model="gpt-3.5-turbo"): # Use correct model
31     name
32     response = client.chat.completions.create(
33         model=model,
34         messages=messages,
35         max_tokens=100,
36         n=1,
37         stop=None,
38         temperature=0.5
39     )
40     message = response.choices[0].message.content
41     return message
42
43 messages = []
44
45 # Set up GPIO
46 GPIO.setmode(GPIO.BCM)
47 GPIO.setwarnings(False)
48 GPIO.setup(17, GPIO.OUT)
49
50 while True:
51     try:

```

```

49     with sr.Microphone() as mic:
50         recognizer.adjust_for_ambient_noise(mic, duration=0.2)
51         print("Listening...")
52         audio = recognizer.listen(mic)
53         text = recognizer.recognize_google(audio, language="sr-SP").
lower()
54         print(f"Recognized: {text}")
55         messages.append({"role": "user", "content": text})
56         keyword = "upali"
57         if keyword in text:
58             print("Prona ena klju na rije : upali")
59             print ("LED on")
60             GPIO.output (17,GPIO.HIGH)
61             time.sleep(3)
62             print ("LED off")
63             GPIO.output (17,GPIO.LOW)
64         else:
65             print("Nije prona ena klju na rije .")
66         response = send_to_GPT(messages)
67         messages.append({"role": "assistant", "content": response})
68         print(f"Response: {response}")
69         engine = pyttsx3.init()
70         engine.setProperty(name='voice', value='croatian')
71         engine.setProperty(name='rate', value=140)
72         engine.say(response)
73         engine.runAndWait()
74     except sr.UnknownValueError:
75         print("Sorry, I did not understand that. Please try again.")

```

**Listing 4.6:** Finalni kod

Ovakav pristup predstavlja temelj za daljnji razvitak sustava u kojemu bi se potencijalno detekcijom ključne riječi "upali" upalilo svjetlo ili klima unutar *smart home* sustava.

```
49     with sr.Microphone() as mic:
50         recognizer.adjust_for_ambient_noise(mic, duration=0.2)
51         print("Listening...")
52         audio = recognizer.listen(mic)
53         text = recognizer.recognize_google(audio, language="sr-SP").lower()
54         print(f"Recognized: {text}")
55         messages.append({"role": "user", "content": text})
56         keyword = "upali"
```

```
Shell
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Listening...
Recognized: upali svjetlo
Pronađena ključna riječ: upali
LED on
LED off
Response: Upalio sam svjetlo.
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
```

Slika 4.10: Izlaz skripte s prepoznatom ključnom riječi



## 5. Vlastita izvedba sustava

### 5.1. Minimum potrebnog hardvera

S obzirom da je opisani ugradbeni računalni sustav izveden korištenjem Raspberry Pi 4B modela, za kojega je već rečeno da je vrlo snažno računalo primjereno i mnogo zahtjevnijim primjenama od implementirane, postavlja se pitanje može li se ista funkcionalnost izvesti na platformi slabijih performansi kako bi se smanjili zahtjevi na hardver i minimizirala cijena uređaja.

S tim ciljem je iznova pokušano koristiti Raspberry Pi Zero W i ReSpeaker 2-Mics Pi Hat koji su opisani u poglavlju 3, no ovaj put uz stariju verziju OS-a koja podržava drivere za modul. Odabrana je verzija iz svibnja 2023. godine i microSD kartica od 32GB. Nakon što je ustanovljeno da na toj verziji HAT uistinu radi i prima audio ulaz, isproban je rad Python skripte dan u odsječku 5.1. Umjesto audio ulaza, program prima tekstualni input kojeg šalje preko OpenAI API-ja GPT-ju na obradu, te on vraća odgovor u obliku teksta koji se ispisuje na zaslonu i to se izvodi u beskonačnoj petlji. Prikaz izvođenja danog programa je vidljiv na slici 5.1.

```
1 import openai
2 from openai import OpenAI
3
4 # Set your OpenAI API key
5 OPENAI_KEY = "sk-proj-tRDVZh4WeADVygq8jmlVT3BlbkFJrvOFiE49HpbctLqzZbEw"
6 client = OpenAI(api_key=OPENAI_KEY)
7
8 def send_to_GPT(messages, model="gpt-3.5-turbo"): # Use correct model
9     name
10     response = client.chat.completions.create(
11         model=model,
12         messages=messages,
13         max_tokens=100,
14         n=1,
15         stop=None,
16         temperature=0.5
```

```

16     )
17     message = response.choices[0].message.content
18     return message
19
20 messages = []
21
22 while True:
23     try:
24         text = input("Enter your request: ")
25         print(f"Recognized: {text}")
26         messages.append({"role": "user", "content": text})
27         response = send_to_GPT(messages)
28         messages.append({"role": "assistant", "content": response})
29         print(f"Response: {response}")
30     except sr.UnknownValueError:
31         print("Sorry, I did not understand that. Please try again.")

```

**Listing 5.1:** Program za povezivanje s GPT-om

The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script with the following code:

```

1 import openai
2 from openai import OpenAI
3
4 # Set your OpenAI API key
5 OPENAI_KEY = "sk-proj-tRDVZh4WeADVygg8jmLVT3B1bkFJrv0FiE49HpbctLqzZbEw"
6 client = OpenAI(api_key=OPENAI_KEY)
7
8 def send_to_GPT(messages, model="gpt-3.5-turbo"): # Use correct model name
9     response = client.chat.completions.create(
10         model=model,
11         messages=messages,
12         max_tokens=100,
13         n=1,
14         stop=None,
15         temperature=0.5

```

The Shell window at the bottom shows the execution output:

```

>>> %Run jarvis_test_clean.py
Enter your request: when did raspberry pi first launch
Recognized: when did raspberry pi first launch
Response: The Raspberry Pi was first launched on February 29, 2012.
Enter your request: |

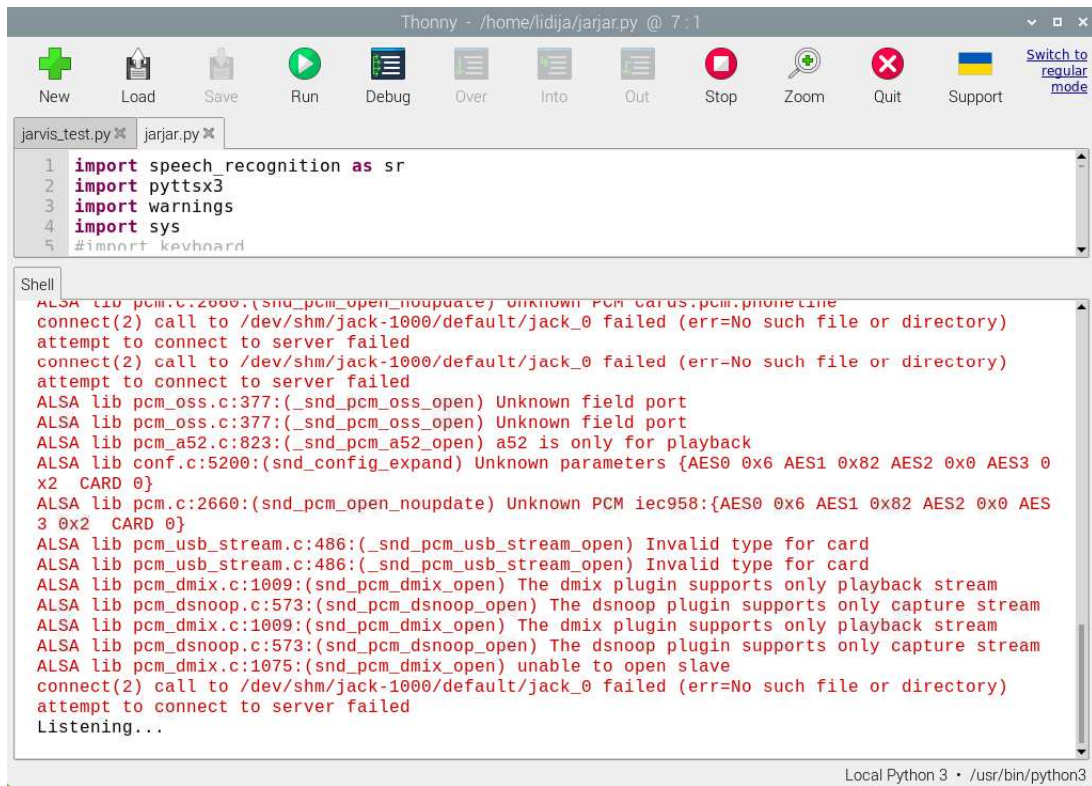
```

**Slika 5.1:** Izlaz skripte

Potom je nadodan dio koda koji detektira i prepoznaje govor opisan u odsječku 4.3, pri čemu varijabla `text` više nije definirana kao tekstualni input kojeg unosi korisnik, već kao prepoznati govor s audio ulaza pretvoren u tekst, te je i ta funkcionalnost implementirana bez grešaka. Problem je nastao pri pokretanju cijele skripte, odnosno kad je nadodan TTS



dio koda dan odsječkom 4.5. Javilo se mnogo ALSA errora, no isprva je program radio bez obzira na njih i implementacija cjelokupnog sustava je bila uspješna. Međutim, već po idućem bootu program više nije radio. Prikaz izvođenja cijele skripte je dan na slici 5.2.



```
Thonny - /home/lidija/jarjar.py @ 7:1
New Load Save Run Debug Over Into Out Stop Zoom Quit Support
Switch to regular mode

jarvis_test.py jarjar.py
1 import speech_recognition as sr
2 import pyttsx3
3 import warnings
4 import sys
5 #import keyboard

Shell
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneLine
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_a52.c:823:(snd_pcm_a52_open) a52 is only for playback
ALSA lib conf.c:5200:(snd_config_expand) Unknown parameters {AES0 0x6 AES1 0x82 AES2 0x0 AES3 0
x2 CARD 0}
ALSA lib pcm.c:2660:(snd_pcm_open_noupdate) Unknown PCM iec958:{AES0 0x6 AES1 0x82 AES2 0x0 AES
3 0x2 CARD 0}
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1009:(snd_pcm_dmix_open) The dmix plugin supports only playback stream
ALSA lib pcm_dsnoop.c:573:(snd_pcm_dsnoop_open) The dsnoop plugin supports only capture stream
ALSA lib pcm_dmix.c:1009:(snd_pcm_dmix_open) The dmix plugin supports only playback stream
ALSA lib pcm_dsnoop.c:573:(snd_pcm_dsnoop_open) The dsnoop plugin supports only capture stream
ALSA lib pcm_dmix.c:1075:(snd_pcm_dmix_open) unable to open slave
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
Listening...

Local Python 3 · /usr/bin/python3
```

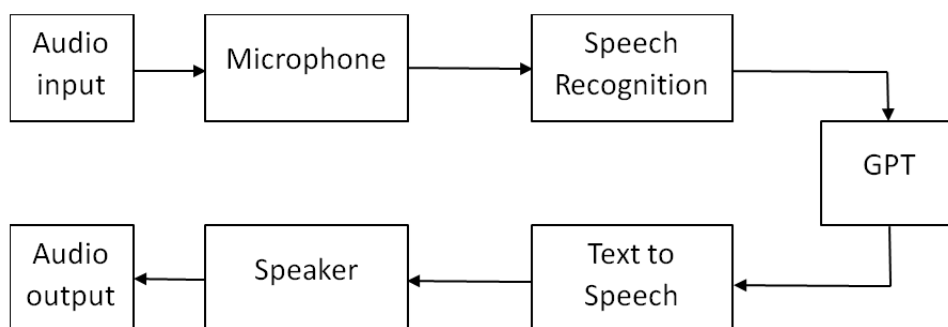
Slika 5.2: Izlaz skripte

Bez obzira na ultimativne rezultate izvođenja skripte, uz utrošak vremena na otklanjanje ALSA errora ili pak korištenja jednostavnijih komponenti poput USB mikrofona i zvučnika koji su korišteni u kombinaciji s Raspberry Pi 4B, zaključak je da funkcionalnost sustava može biti ostvarena uz korištenje jednostavnijih platformi. Takvo rješenje minimizira dimenzije sustava, otklanja dio komponenti i portova na PCB-u koji za ovu primjenu nisu potrebni, ima puno manju potrošnju te smanjuje cijenu. Raspberry Pi Zero W je u odnosu na model 4B 2-5 puta jeftiniji, ovisno o varijanti 4B modela. Korištenje ovakve ekonomičnije varijante dolazi uz mnogo *trade-off*-ova, a to su procesorska moć (*single-core* vs *quad-core*), veličina RAM-a (512 MB vs 1 - 8GB, ovisno o verziji), slabija bežična povezanost kao i slabiji podatkovni portovi itd. Zbog velike razlike u performansama između dva korištena Raspberry Pi-ja, razlika u vremenu izvođenja danih programa je osjetna. No, s obzirom da vrijeme izvođenja implementacije glasovnog asistenta nije od kritične važnosti u ovome slučaju, to u ovome trenutku niti nije toliko bitno.

## 5.2. Ideja vlastitog PCB-a

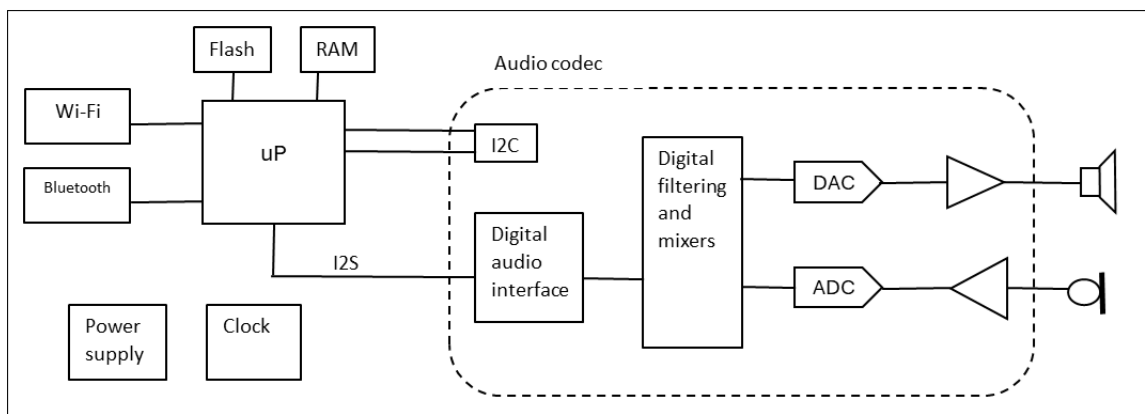
Kako bi se postigla neovisnost o dobavljalivosti korištenih platformi, smanjili zahtjevi na potreban hardver te time potencijalno smanjila cijena, mogućnost izrade sustava u vlastitoj izvedbi je dovedena u razmatranje. Cjelokupna izvedba bi uključivala istraživanje potrebnih električkih komponenti, izradu električne sheme te dizajn i izradu PCB-a (engl. *Printed Circuit Board*). Ovdje će biti opisana načelna ideja i prikaz u obliku blok sheme.

Na slici 5.3 prikazan je tok signala, a na slici 5.4 principijelna blok shema PCB-a. Prepoznavanje govora i sinteza govora se, kao što je opisano u poglavlju 4, odvijaju u oblaku. Iako je u prethodnom potpoglavlju prikazana izvedba sustava pomoću Raspberry Pi Zero W platforme, za ostvarivanje zadovoljavajuće brzine odaziva sustava i obradu zahtjeva u realnom vremenu, potrebno je odabrati dovoljno snažan mikroprocesor s barem 1 GB RAM memorije koji uz to nudi i bolju i bržu bežičnu povezanost. Na blok shemi je nešto detaljnije prikazan integrirani audio kodek. Ulazni audio signal dolazi preko mikrofona i predpojačala do A/D konvertera, nakon čega se u digitalnom obliku obrađuje u jedinici za digitalno filtriranje i miksiranje. Kodek putem I2S protokola komunicira s mikroprocesorom, gdje se digitalni audio signal šalje na obradu u oblaku. Nakon obrade se istim putem izlazni audio signal vraća na kodek, pomoću D/A konvertera se vraća u analogni oblik te se pojačava i pušta na zvučnik. Mikroprocesor također koristi I2C protokol za konfiguraciju kodeka i pružanje informacija o *clock*-u, formatu i tipu audio podataka koji će biti korišteni.



**Slika 5.3:** Tok signala

S cjenovnog stajališta, s obzirom da se radi o osobnom projektu a ne dizajnu komercijalnog proizvoda, procjenjeno je da bi cijena za vlastito rješenja bila viša nego uz korištenje postojećih platformi s obzirom da bi samo mikroprocesor koštao jednako ili više no cijeli Raspberry Pi Zero W.



**Slika 5.4:** Blok shema PCB-a

## 6. Zaključak

U ovome diplomskom radu opisana je izvedba govorom upravljano ugradbenog računalnog sustava uz dvije različite Raspberry Pi platforme. Implementirana je funkcionalnost prepoznavanja govora pomoću Google Web Speech API-ja, obrađivanja transkripta u GPT modelu putem OpenAI API-ja te sinteze govora. Također je dodana funkcionalnost paljenja eksterne LED diode pri prepoznavanju ključnih riječi. U jednoj je varijanti korišten Raspberry Pi 4 Model B s USB mikrofonom i zvučnicima, dok u drugoj Raspberry Pi Zero W s audio kodekom ReSpeaker 2-Mics Pi HAT i zvučnicima, čime je istraženo korištenje minimuma potrebnog hardvera za ostvarivanje navedene funkcionalnosti. Na kraju je dana principijela blok shema PCB-a koji bi predstavljao vlastitu izvedbu sustava uz pretpostavku da bi takvo rješenje iziskivalo veću cjenovnu potrošnju, ali dalo mogućnost prilagodbe konkretnoj primjeni.

# LITERATURA

<https://www.techradar.com/news/the-evolution-of-speech-recognition-technology>  
<https://www.smartsheet.com/voice-assistants-artificial-intelligence>  
<https://en.wikipedia.org/wiki/WaveNet>  
<https://www.amazon.science/blog/alexa-at-five-looking-back-looking-forward>  
[https://en.wikipedia.org/wiki/Amazon\\_Alexa](https://en.wikipedia.org/wiki/Amazon_Alexa)  
<https://www.tomshardware.com/features/raspberry-pi-zero>  
<https://www.seeedstudio.com/ReSpeaker-2-Mics-Pi-HAT.html>  
[https://wiki.seeedstudio.com/ReSpeaker\\_2\\_Mics\\_Pi\\_HAT/](https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/)  
<https://sg.element14.com/raspberry-pi/rpi-zero-w-v2/raspberry-pi-board-arm-cortex/dp/3838499>  
<https://www.amazon.in/seeed-Studio-ReSpeaker-2-Mics-HAT/dp/B07CXSW6LB>  
[https://wiki.seeedstudio.com/ReSpeaker\\_2\\_Mics\\_Pi\\_HAT\\_Raspberry/](https://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT_Raspberry/)  
<https://datasheets.raspberrypi.com/audio/codec-zero-phat-product-brief.pdf>  
<https://thepihut.com/products/iqaudio-codec-zero>  
<https://www.raspberrypi.com/documentation/accessories/audio.html#configuration>  
<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>  
<https://www.trust.com/en/product/21674-primo-desk-microphone-for-pc-and-laptop-leaflets>  
<https://www.raspberrypi.com/documentation/computers/os.html>  
<https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pi-gpio-pins>  
<https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio-and-the-40-pin-header>  
<https://www.analog.com/media/en/technical-documentation/data-sheets/MAX9860.pdf>  
<https://pcbartists.com/products/es8388-module/esp32-es8388-audio-codec-interfacing/?currency=USD>

# **Govorom upravljani ugradbeni računalni sustav temeljen na Amazon Alexa Voice Service tehnologiji**

## **Sažetak**

S obzirom na nemogućnost implementacije virtualnog asistenta Alexa tvrtke Amazon, u ovome radu opisana je izvedba ugradbenog računalnog sustava upravljanog govorom koji implementira govornu interakciju s ChatGPT platformom tvrtke OpenAI, kao i prikaz prepoznavanja ključnih riječi i mogućnost izvršetka radnji na temelju toga. Za implementaciju softverskog dijela sustava korišten je operacijski sustav Raspberry Pi OS temeljen na Linuxu te programski jezik Python, a hardverskog dijela platforma Raspberry Pi 4 Model B s potrebnim periferijama. U svrhu istraživanja minimalnog potrebnog hardvera za ostvarivanje opisane funkcionalnosti korištena je platforma Raspberry Pi Zero W. Na kraju je principijelno objašnjena izgradnja sustava u vlastitoj izvedbi.

**Ključne riječi:** Ugradbeni računalni sustav, Raspberry Pi, ChatGPT, prepoznavanje govora, sinteza govora

## **Title**

## **Abstract**

Given the inability to implement Amazon's Alexa virtual assistant, this paper describes the implementation of an embedded system controlled by speech, which enables voice interaction with the ChatGPT platform by OpenAI, as well as the demonstration of keyword recognition and the ability to perform actions based on it. The Raspberry Pi OS, based on Linux, and the Python programming language were used for the software part of the system, while the Raspberry Pi 4 Model B platform with the necessary peripherals was used for the hardware part. For the purpose of researching the minimal necessary hardware for achieving the described functionality, the Raspberry Pi Zero W platform was used. Finally, the construction of the system in its own implementation is principally explained.

**Keywords:** Embedded system, Raspberry Pi, ChatGPT, speech recognition, text-to-speech