JOONA – A VOICE-CONTROLLED IOT ASSISTANT

REAL-TIME SPEECH-TO-ACTION SYSTEM WITH ESP32

Presented By : Fatima Ahmad(1341)
Maryam Munawar(1344)
Fatima Riaz(1342)
Marriam Sajjad(1353)

🛠️ CORE TECHNOLOGIES:

01 ESP32-S3 (I2S MIC & SPEAKER, RELAYS)

02 🎙️ I2S MICROPHONE (AUDIO CAPTURE)

03 🔈 I2S SPEAKER (AUDIO PLAYBACK)

04 ⚡ RELAY MODULES (DEVICE CONTROL: LIGHT, FAN)

05 FLASK (PYTHON SERVER BACKEND)

06 🗣️ ASSEMBLYAI (SPEECH-TO-TEXT)

07 🤖 OPENAI (INTENT RECOGNITION, RESPONSE GENERATION FALLBACK)

08 🗣️ GOOGLE TTS (TEXT-TO-SPEECH FOR PLAYBACK)

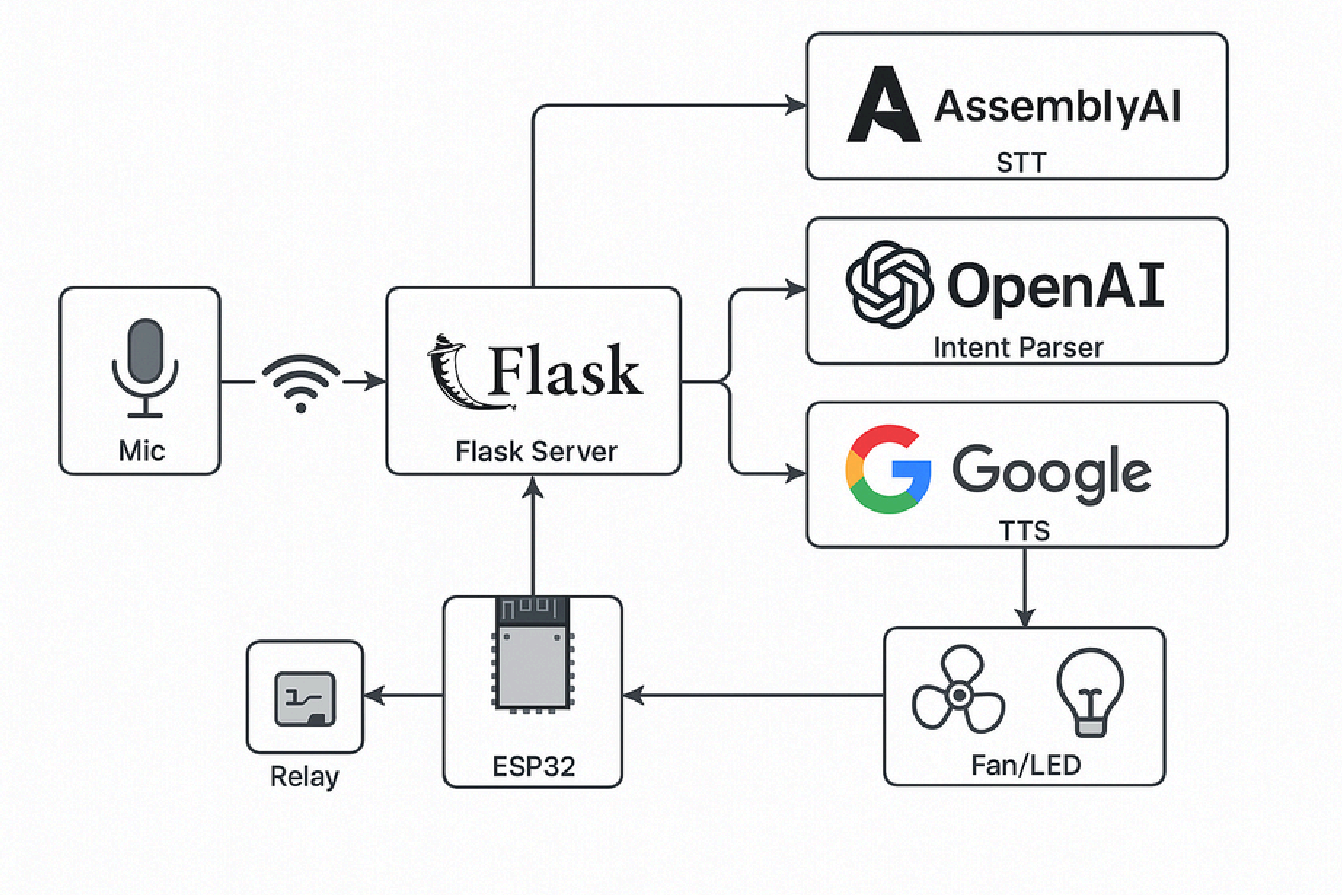
09 ☁️ FIREBASE FIRESTORE

INTRODUCTION

Joona is a real-time voice-controlled IoT assistant powered by the ESP32-S3 microcontroller, capable of interpreting natural language commands to control physical devices like lights and fans via relay modules. It uses I2S audio hardware to capture voice input, which is sent to a Flask-based Python server over HTTP. The server transcribes the audio using AssemblyAI, detects intent via rule-based logic and OpenAI's language model, and responds accordingly. Device control commands are sent back to the ESP32 to trigger GPIO actions, while voice feedback is generated using Google Text-to-Speech and played through an I2S speaker. Joona also supports reminder scheduling using Firebase Firestore and APScheduler, showcasing seamless integration of embedded systems with AI-powered cloud services.

SYSTEM OVERVIEW / WORKFLOW

[Button Press → ESP32 Starts Recording via I2S Mic]
→ [HTTP POST WAV to Flask Server]
→ [STT: AssemblyAI]
→ [Intent Detection: OpenAI / Rule-Based Parser]
→ [Response: Google TTS → WAV file]
→ [ESP32 Downloads wav file → Playback via I2S Speaker]
→ [If Device Command → Trigger GPIO Relays (Light/Fan)]
→ [If Reminder Intent → Store in Firebase Firestore
→ Triggered by APScheduler]



ESP32 FIRMWARE (AUDIO CAPTURE & HTTP SEND)

LIBRARIES USED:

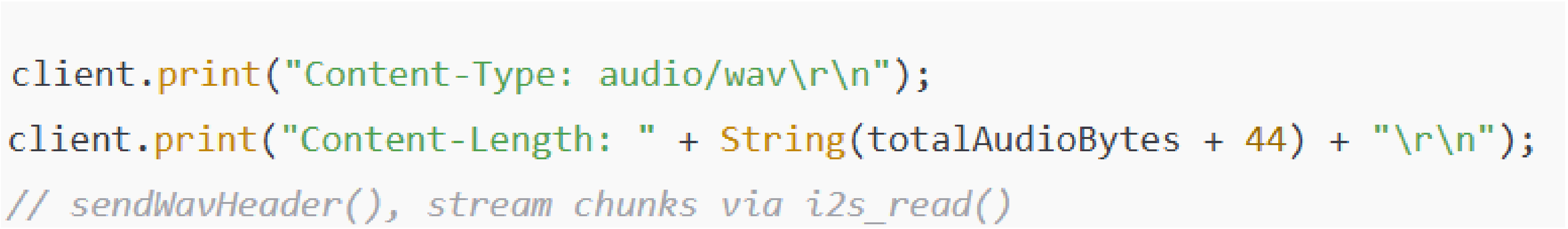
WiFi.h, HTTPClient.h, ArduinoJson.h, driver/i2s.h

CODE RESPONSIBILITIES:

- Setup I2S mic + speaker
- Capture 3 seconds of audio
- Send to server as audio/wav with custom WAV header
- Await server JSON response

KEY FUNCTION:

streamAudioToServer()



```
client.print("Content-Type: audio/wav\r\n");
client.print("Content-Length: " + String(totalAudioBytes + 44) + "\r\n");
// sendWavHeader(), stream chunks via i2s_read()
```

PYTHON FLASK SERVER – AUDIO ENDPOINT (/UPLOAD-AUDIO)

DETECTS CONTENT-TYPE:

- multipart/form-data (for Web UI uploads)
- audio/wav (for ESP32)

TRANSCRIBES USING:

```
text = transcribe_audio(filepath)["text"]
```

SAVES AUDIO TO DISK:

```
with open(filepath, "wb") as f:
    f.write(request.get_data())
```

INTENT DETECTION:

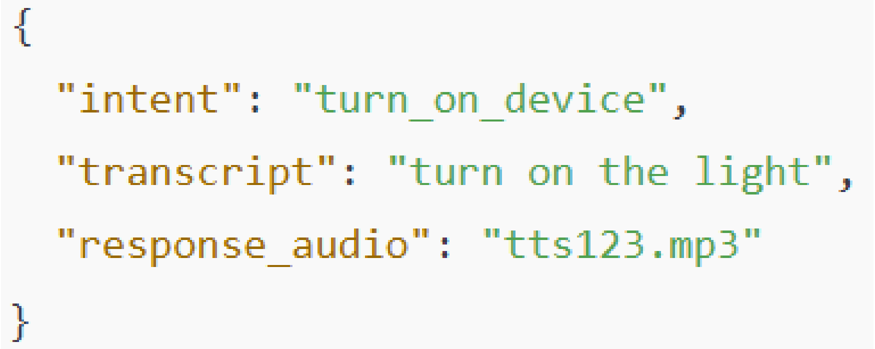
- Rule-based (intent_parser)
- Fallback: OpenAI (generate_chat_response)

ADDS “**JOONA**” KEYWORD CONDITIONALLY TO PERSONALIZE RESPONSE.

REAL-TIME VOICE. REAL-WORLD ACTIONS

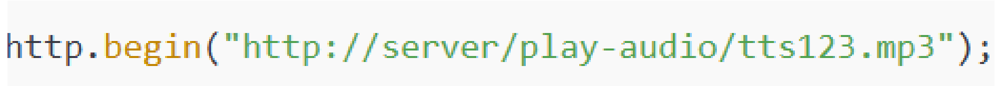
ESP32 RESPONSE HANDLING & PLAYBACK

RECEIVES JSON RESPONSE:



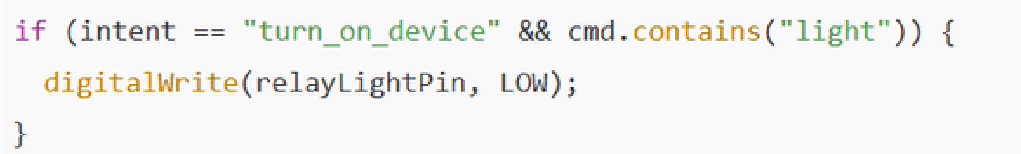
```
{
  "intent": "turn_on_device",
  "transcript": "turn on the light",
  "response_audio": "tts123.mp3"
}
```

DOWNLOADS MP3 VIA HTTP:



```
http.begin("http://server/play-audio/tts123.mp3");
```

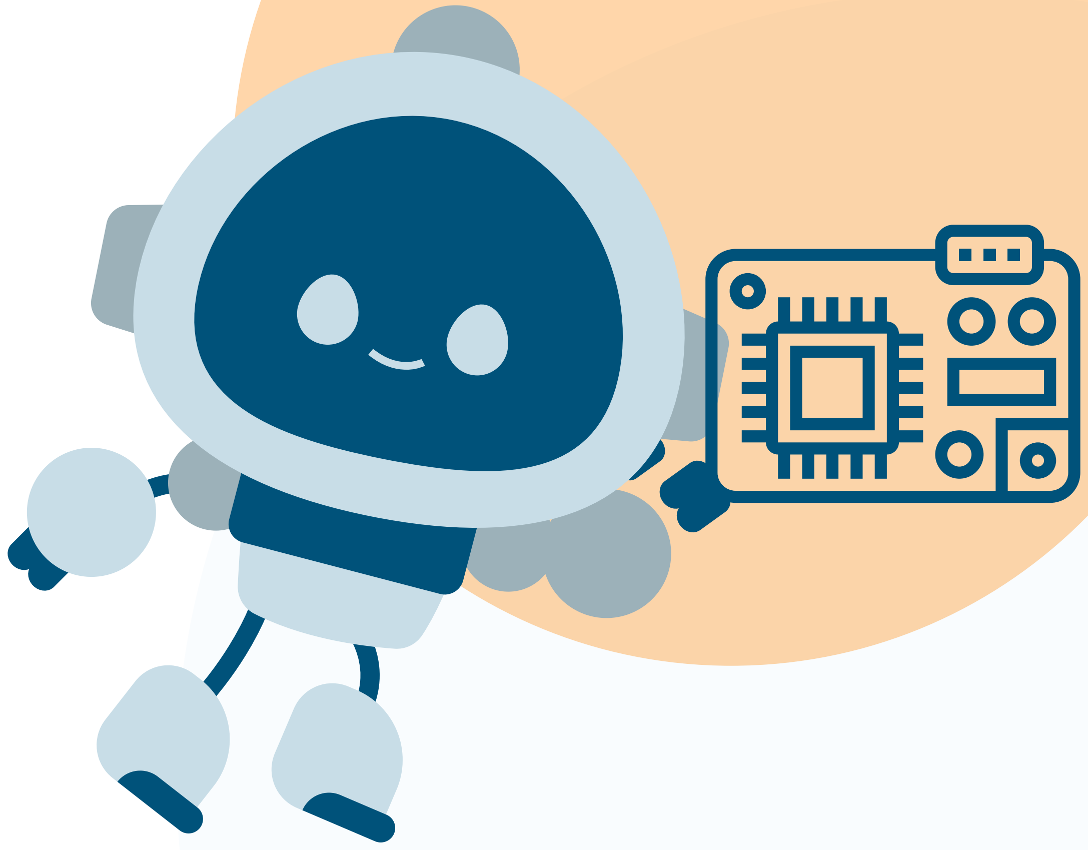
PARSE JSON, TRIGGER GPIO:



```
if (intent == "turn_on_device" && cmd.contains("light")) {
  digitalWrite(relayLightPin, LOW);
}
```

PLAYS VIA I2S SPEAKER:

i2s_write(buffer, len, &written)



INTEGRATION & TESTING

✅ **All components tested together:**

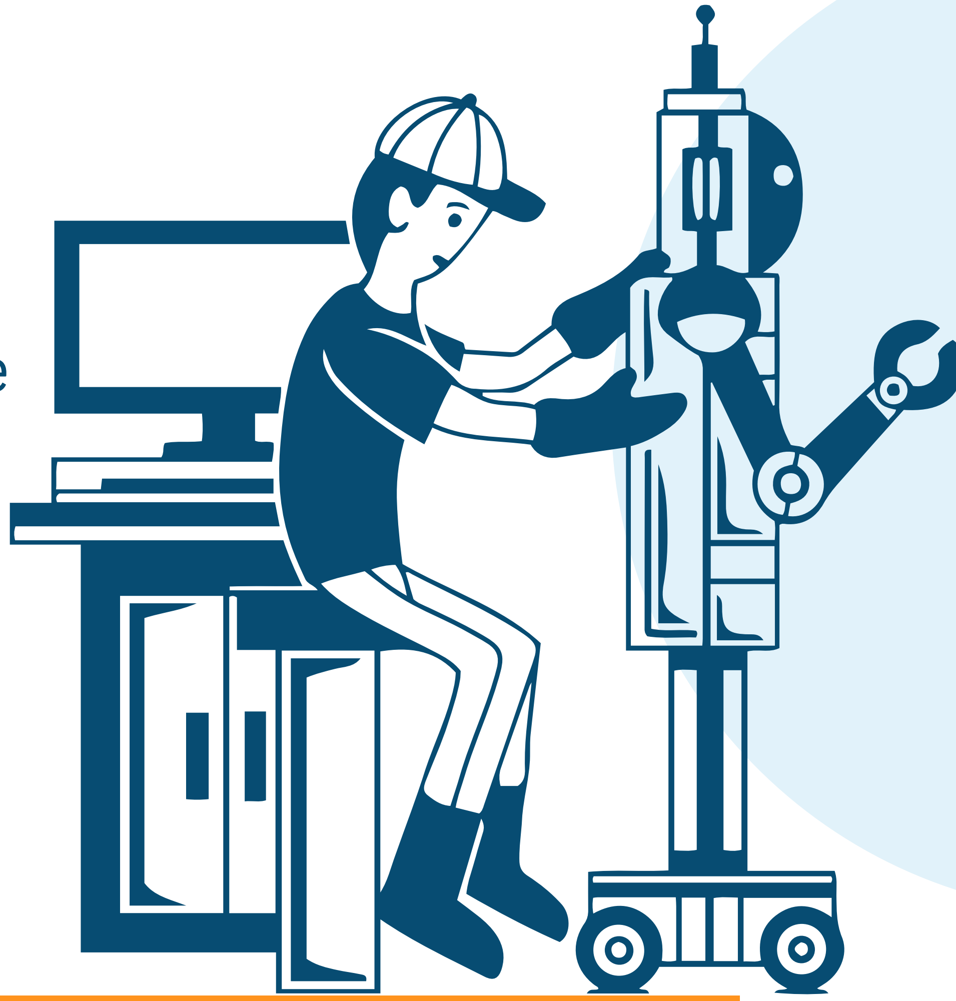
- Real-time voice commands to control LED and Fan
- Reminders set via natural language
- Voice feedback played back successfully

🧪 **Testing Process:**

- Verified STT accuracy with ambient noise
- JSON parsing on ESP32 validated
- Relay toggling validated through Serial Monitor + physical device

⚠️ **Limitations:**

- Dependency on WiFi + server availability
- Button not yet implemented (future**)**



CONCLUSION & FUTURE ENHANCEMENTS

Key Achievements:

- Integrated ESP32-S3 with cloud-based AI
- Real-time voice control of physical devices
- Speech processed via AssemblyAI & OpenAI
- TTS responses generated using Google TTS
- Reminders managed through Firebase & APScheduler

Future Enhancements:

- Add offline wake-word detection
- Improve latency via local caching or faster STT
- Display responses on OLED / LED matrix
- Expand to control more smart devices
- Goal: Transform Joona into a fully-featured voice assistant like Alexa