Portable Offline Translator

Team 77

Lorenzo Bujalil Silva, Joshua Cudia

Presentation Overview

- **Project Overview**
- **Motivation**
- **Project Design**
- **Problems Encountered**
- **Video Demonstration**
- **Results**
- **Conclusion**

Project Overview

Project Overview

Offline Speech Translator on Embedded Hardware

- Translates spoken language without an internet connection
- Built on ESP32-S3 and Raspberry Pi CM5
- Inference Pipeline
  - Speech Recognition: Whisper.cpp
  - Speech Translation: Llama.cpp
  - Text-to-Speech: Piper
- Audio I/O via I2S
- SPI LCD Language Selection Menu
- Designed for portability and real time usage

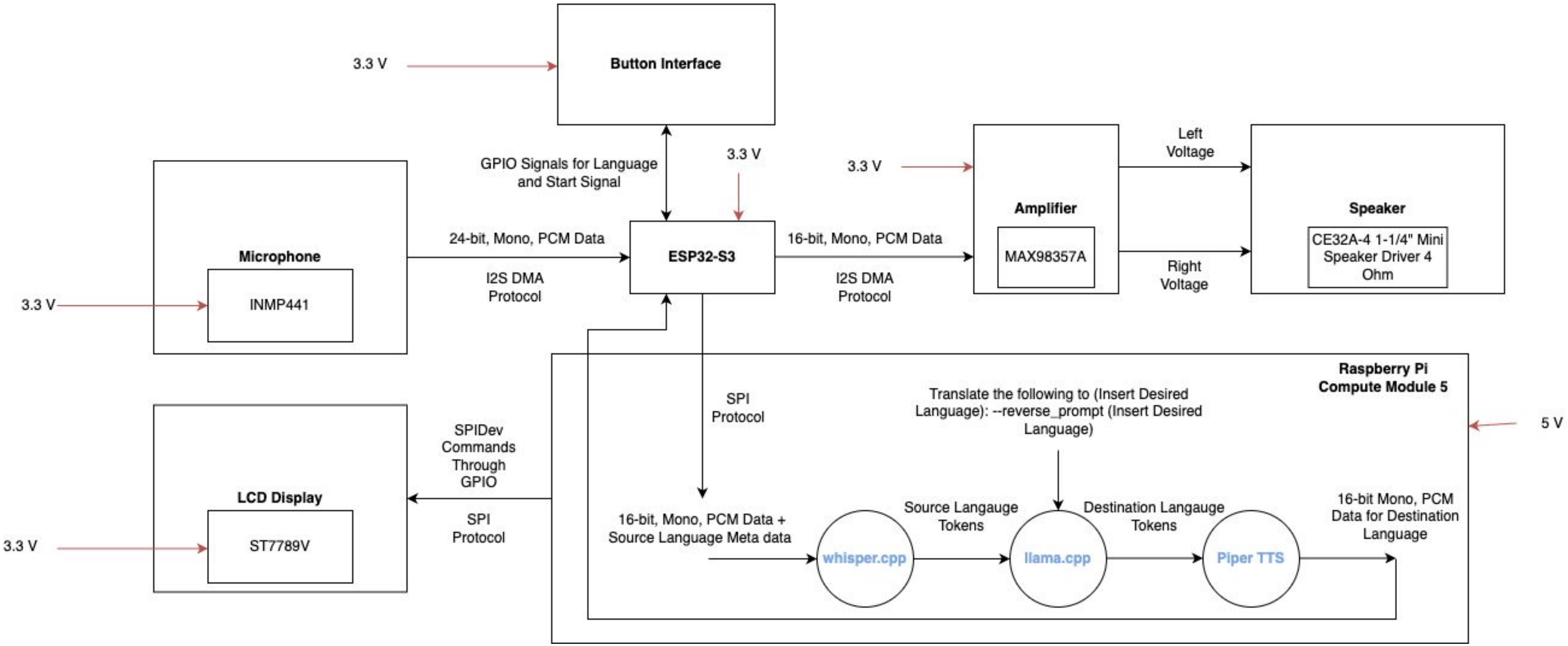
Motivation

Motivation

- We love to travel and explore new cultures
- Language barriers make communication difficult
- In the case of emergency situations, reliable communication is essential
- Online connection isn’t always available abroad
- We wanted a device that can work anywhere, anytime

Project Design

Block Diagram



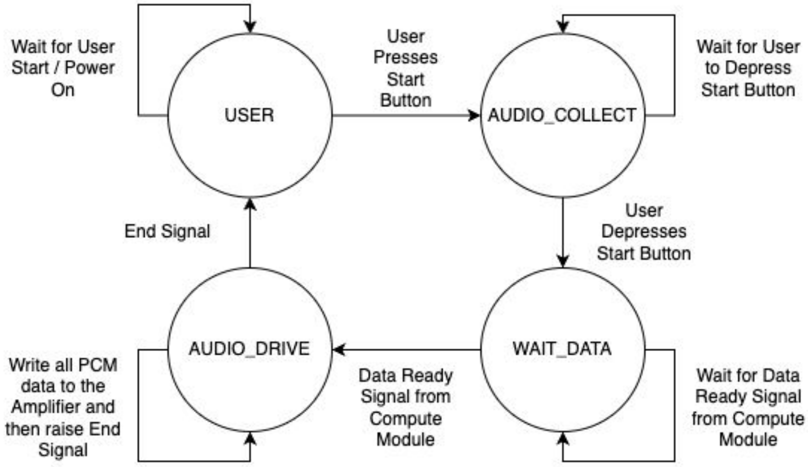
Design Requirements

- **Microcontroller Subsystem (ESP32-S3)**
  - Interfaces with audio I/O
  - Manages audio data buffering and SPI communication with the CM5
- **Compute Subsystem (Raspberry Pi Compute Module 5)**
  - Runs STT, translation, and TTS models locally
  - Manages UI logic and controls display updates
- **Audio I/O Subsystem**
  - Reads PCM data from a digital-output MEMS microphone
  - Drives digital PCM input Class D amplifier
- **User I/O Subsystem**
  - LCD shows language options (Source/Destination)
  - GPIO buttons enable language selection and starting inference
- **Power Subsystem**
  - Regulates 5V and 3.3V rails for MCU, Pi, and peripherals

Design Verification

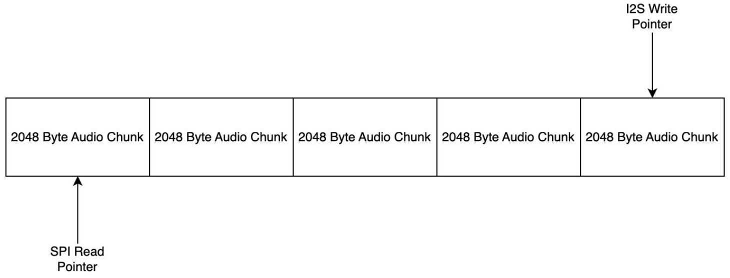
- **Microcontroller Subsystem (ESP32-S3)**
  - Verified I2S audio capture and delivery
  - SPI communication tested using pre-recorded data, achieving <500 ms end-to-end response
- **Compute Subsystem (Raspberry Pi Compute Module 5)**
  - Ran various simulations on models to validate 90% accuracy
  - Button language selection verified via LCD updates
  - Verified SPI reception with waveform comparison with reference audio file
- **Audio I/O Subsystem**
  - Verified audible and intelligible speech I/O
- **User I/O Subsystem**
  - Verified button inputs change language selection
  - Validated SPI commands to update display
- **Power Subsystem**
  - Validated voltage levels at 3.3V and 5V rails
  - Tested under full workload to ensure voltage stability

ESP32-S3 Firmware

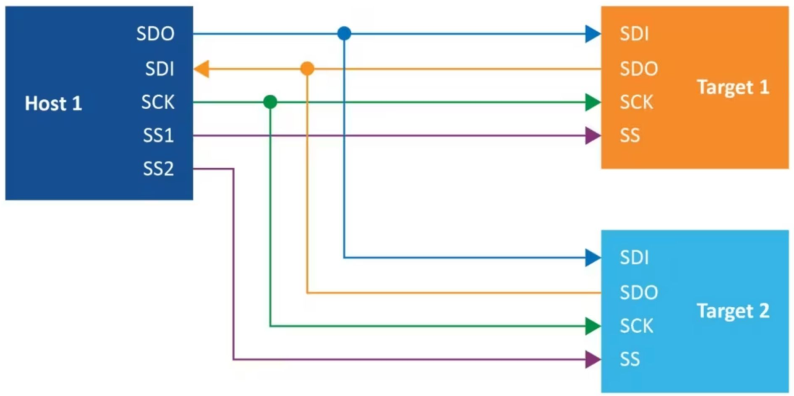


Overview:
- Captures I2S Audio data
- Streams raw PCM data over SPI to CM5
- Receives translated audio over SPI
- Drives amplifier with the translated PCM data via I2S

System Components:
- ESP32-S3: Acts as SPI slave and I2S master for both mic and speaker
- Raspberry Pi: SPI master (sends translated audio back)
- I2S Mic: Captures 16-bit mono audio at 16 kHz
- Amplifier/Speaker: Plays back 16-bit mono audio via I2S



SPI Interface (CM5 Software)



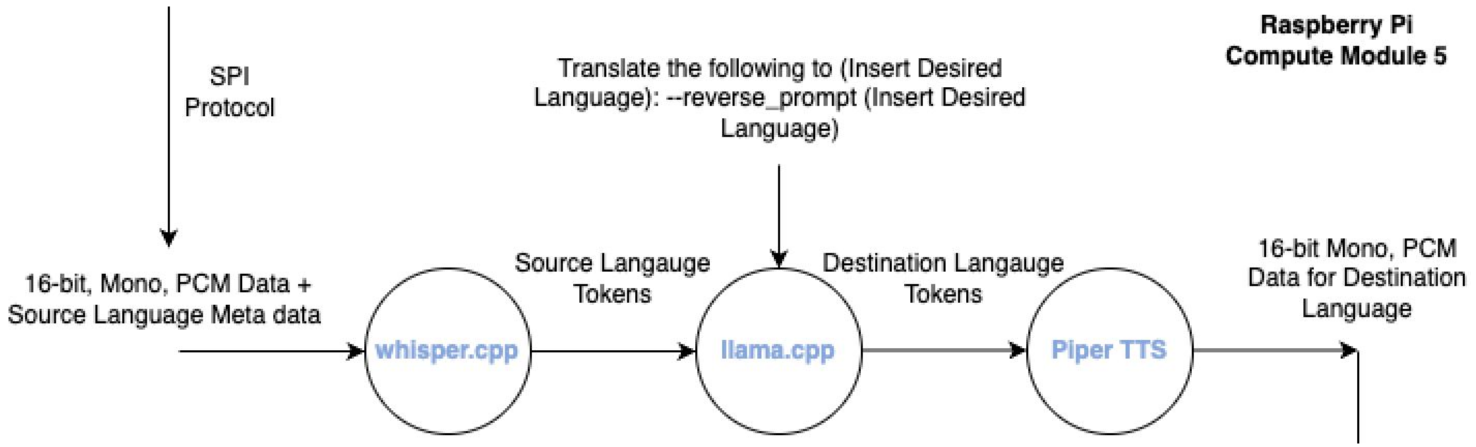
Requirements:
- LCD SPI Display
- Audio Data Transfer via SPI

Configuration:
- SPI Network with multiple peripherals

Audio Data Transfer Protocol:
1. ESP32 triggers GPIO interrupt
2. CM5 polls for audio data from ESP32 via SPI
   a. Creates 16-bit PCM data file
   b. Waits time slice to allow for buffer to populate
3. CM5 triggers signal (SIGUSR1) to start translation
4. CM5 waits for pipeline signal for translated data ready
5. CM5 transmits translated PCM data back to ESP32 va SPI

Inference Pipeline (CM5 Software)

Speech-to-Text: Whisper.cpp converts PCM to source language text tokens
Translation: LLaMA.cpp translates tokens to target language using prompts
Text-to-Speech: Piper TTS converts tokens to 16-bit PCM audio



Speech-to-Text (Whisper.cpp)

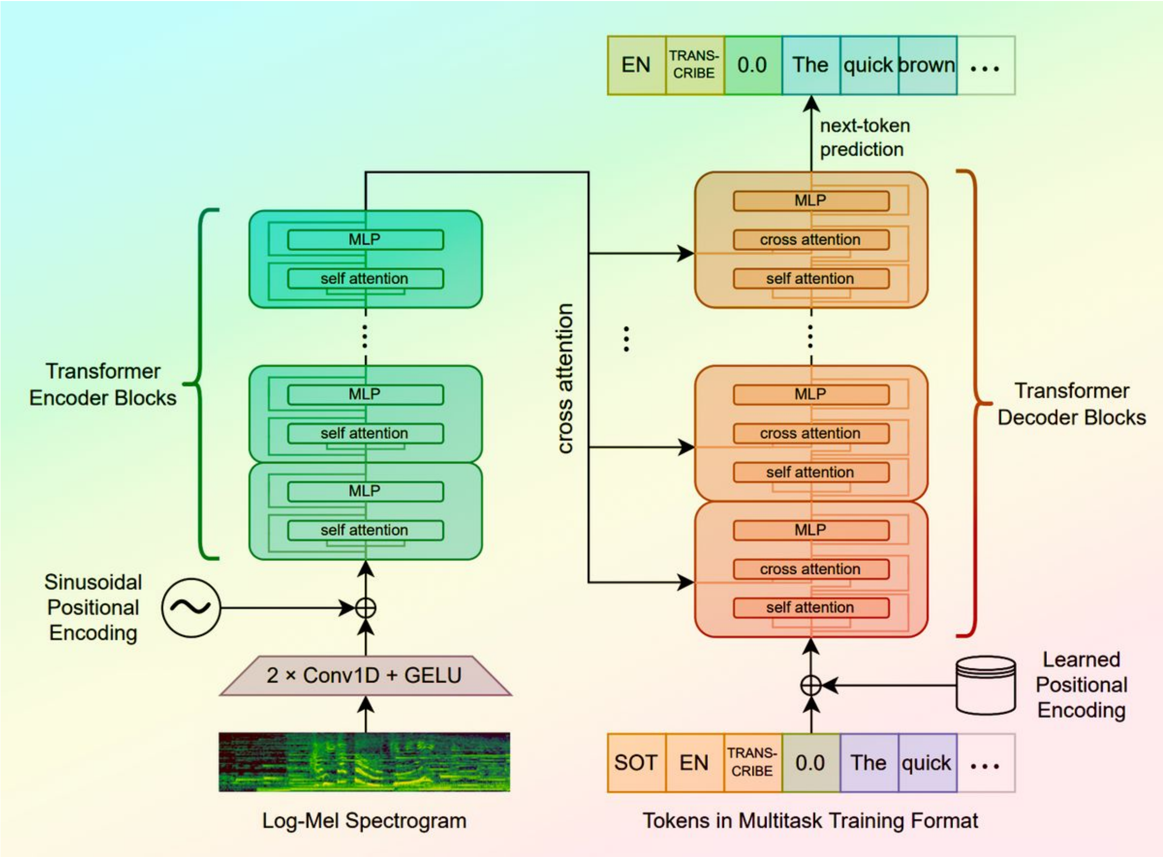
Overview: Whisper.cpp is a C++ implementation for high-performance inference of OpenAI's Whisper automatic speech recognition model

Model Type: Transformer-based encoder-decoder

How it works:
- Converts 16-bit mono PCM audio, sampled at 16kHz, into a log-Mel spectrogram
- Feeds spectrogram into an encoder by mapping audio to latent features
- Decoder autoregressively generates text tokens, using past outputs and attention over audio features

Why Use it?:
- Multilingual Support
- Scalable Model Sizes (75 MiB -> 834 MiB)
- Resilient to noise or accents
- Efficient Local Inference + Lightweight

Speech-to-Text (Whisper.cpp)



Whisper Architecture:

- Log-Mel Spectrogram Input
- 2 x Conv1D + GELU Layers
- Transformer Encoder Blocks
- Transformer Decoder Blocks
- Multitask Prompt Tokens
- Learned Positional Encoding
- Autoregressive Decoding
- Unified Architecture

Translation (LLaMA.cpp)

Overview: LLaMA.cpp is a C++ implementation for inference of Meta's LLaMA model

Model Type: Decoder-only Transformer (GPT)

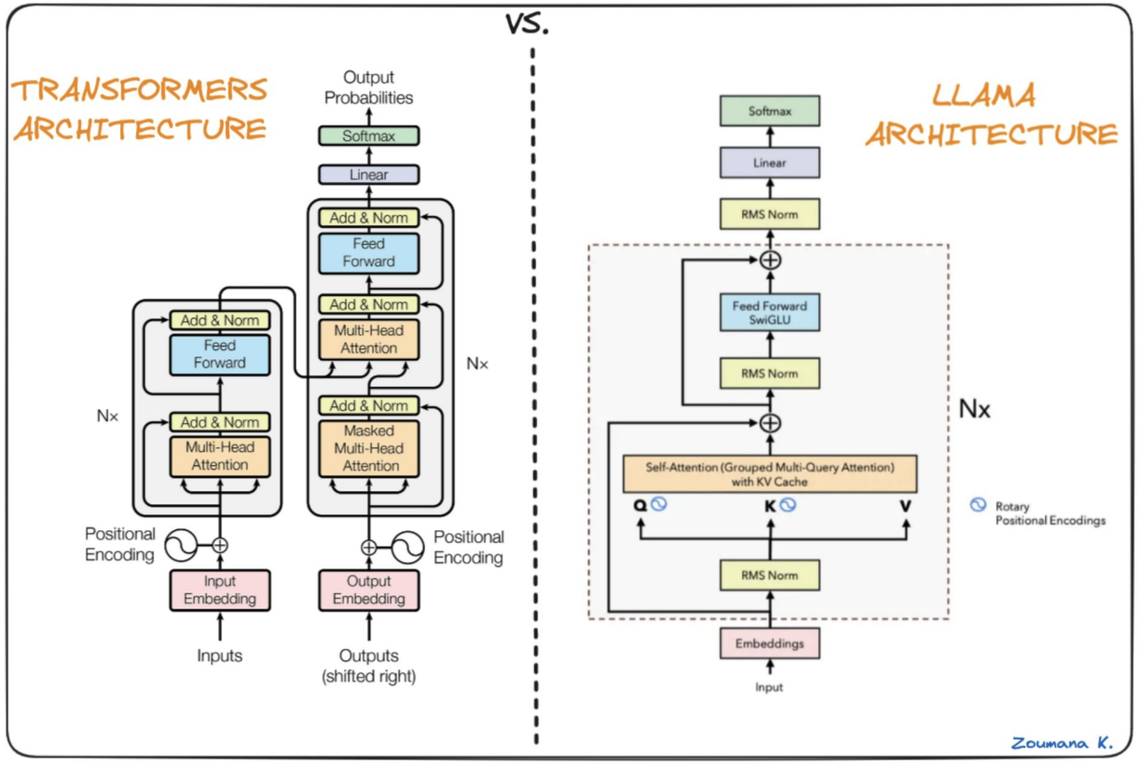
How it works:

- Accepts source language tokens from Whisper
- Pipeline prompts model : *"Translate the following to Spanish: <Whisper output>"*
- LLaMA then auto completes the response as translated text, generating target language tokens

Why Use it?:

- Efficient Local Inference
- Highly Flexible - can do more than simply translation
- Simply prompt the model (GPT-like)

Translation (LLaMA.cpp)



LLaMA Architecture:

- Self-attention with causal masking
- Rotary positional embeddings
- Token-by-token generation
- Key/Value Caching
- Optimized for fast inference on edge devices

Text-to-Speech (Piper TTS)

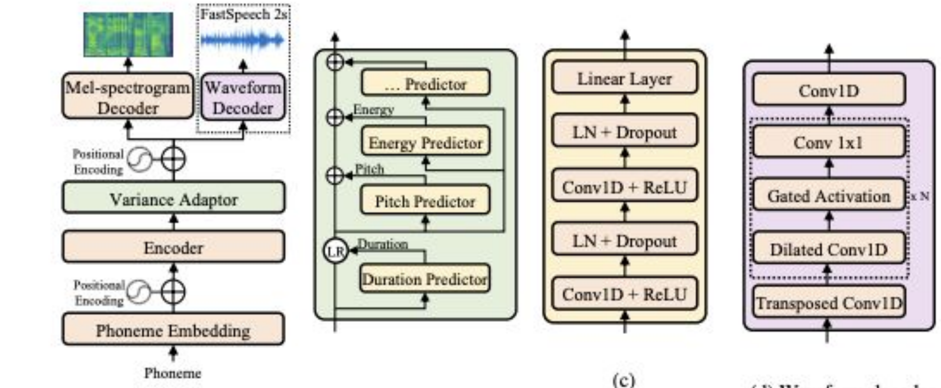
Overview: Piper TTS is a fast, local neural text to speech system

Model Type: FastSpeech2 (non-autoregressive Transformer) + HiFi-GAN vocoder

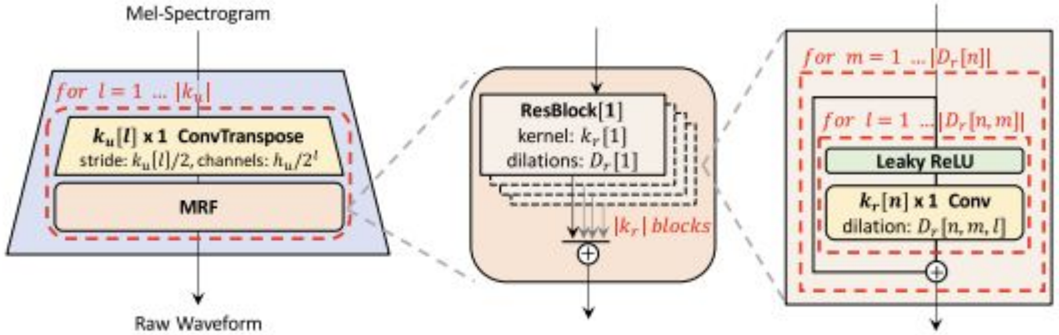
How it works:
- Accepts destination language tokens
- FastSpeech2 predicts mel spectrograms using phoneme-level input
- HiGi-GAN converts spectrograms into 16-bit mono PCM audio

Why Use it?:
- Efficient Local Inference
- Multilingual Support
    - **Must store many voices to use multiple languages**
- High Audio Quality
- CLI Integration

Text-to-Speech (Piper TTS)



(a) FastSpeech 2 (b) Variance adaptor (c) Duration/pitch/energy predictor (d) Waveform decoder



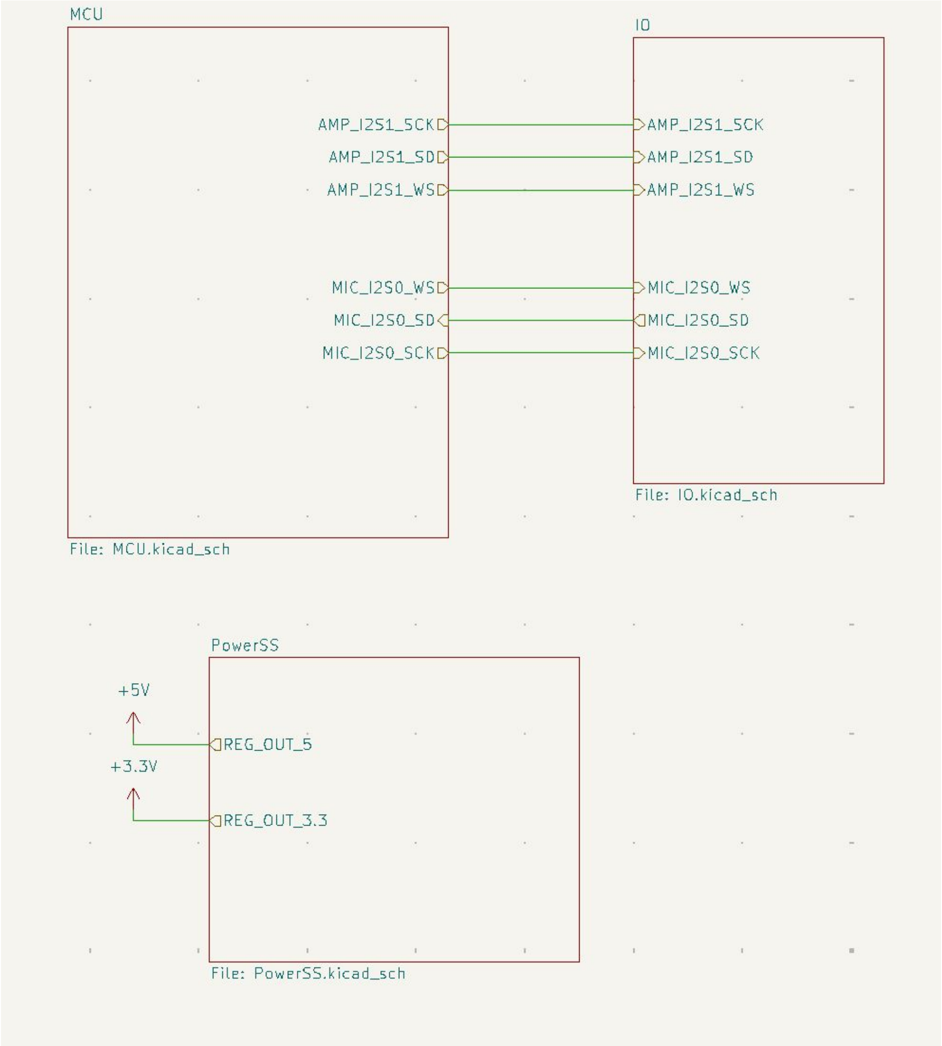
Piper Architecture:

- FastSpeech2
- Variance Adaptor
- HiFi-GAN Vocoder
- Non-Autoregressive Design
- Optimized for fast inference on edge devices

PCB Design - Schematic

Subsystems

- MCU
  - Boot-mode I/O
  - USB-C receptacle
  - Connector Headers
- I/O
  - Audio Amplifier
  - Microphone & Speaker
- Power
  - 3.3V and 5V channel
  - JST connection



Power Subsystem - Schematic

3.3V Regulator Configuration:

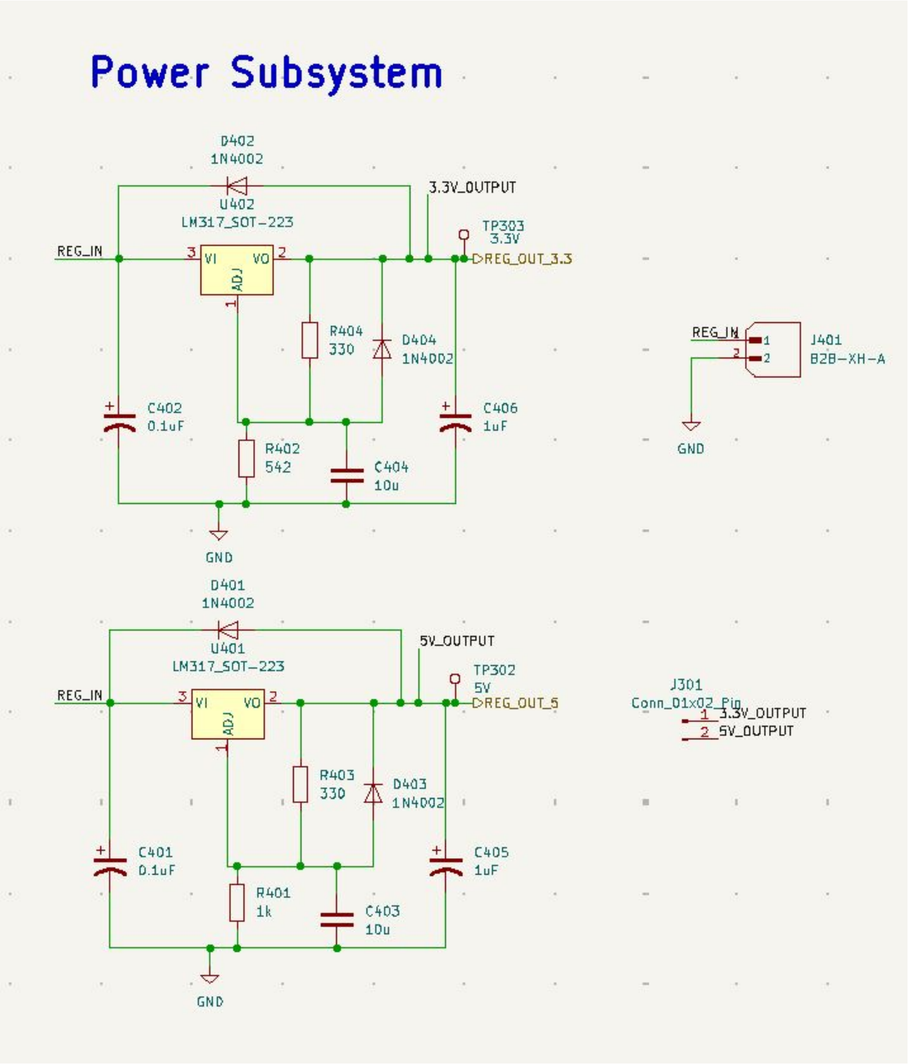
- R1 =330Ω
- R2 = 542Ω

5V Regulator Configuration:

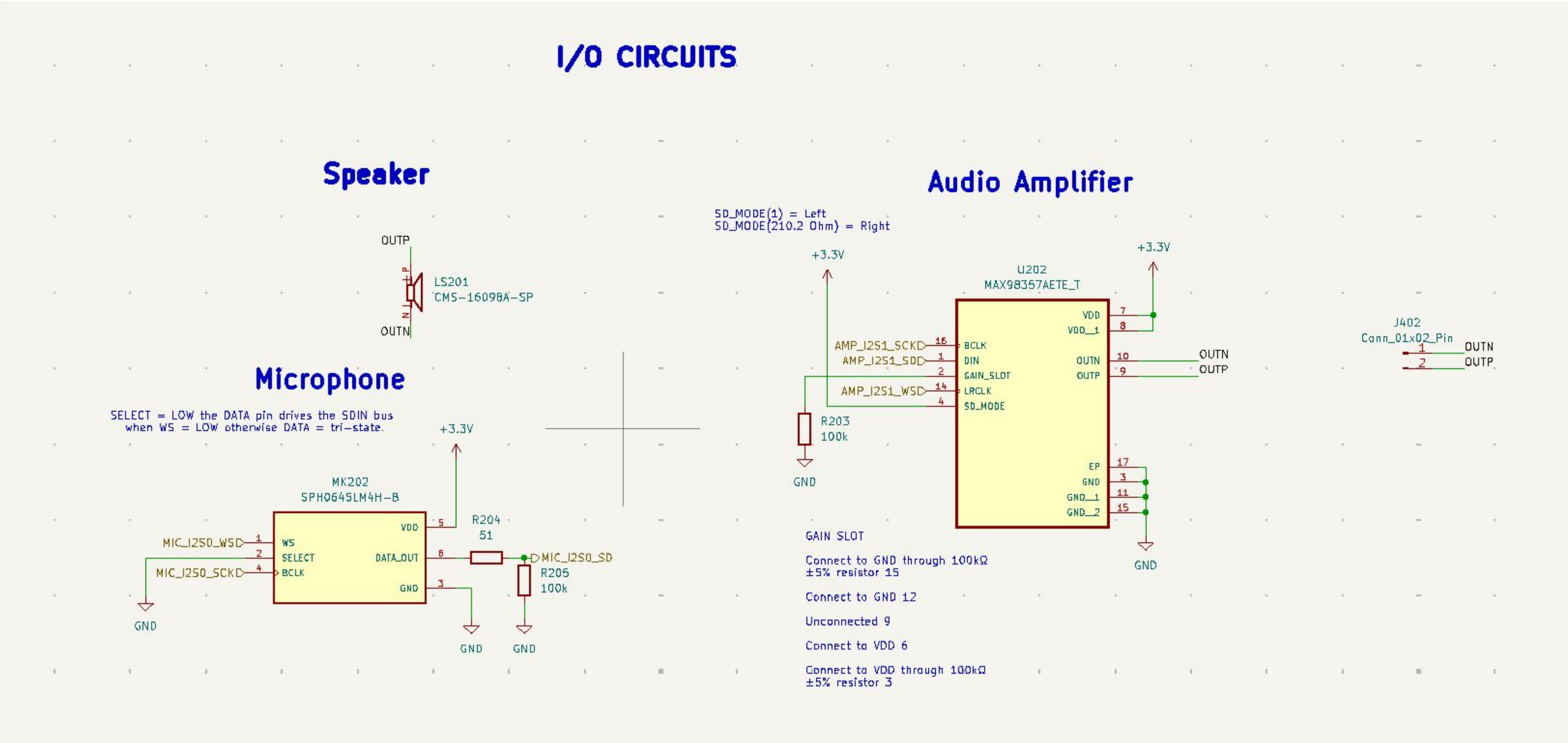
- R1 = 330Ω
- R2 = 1kΩ

Diodes:

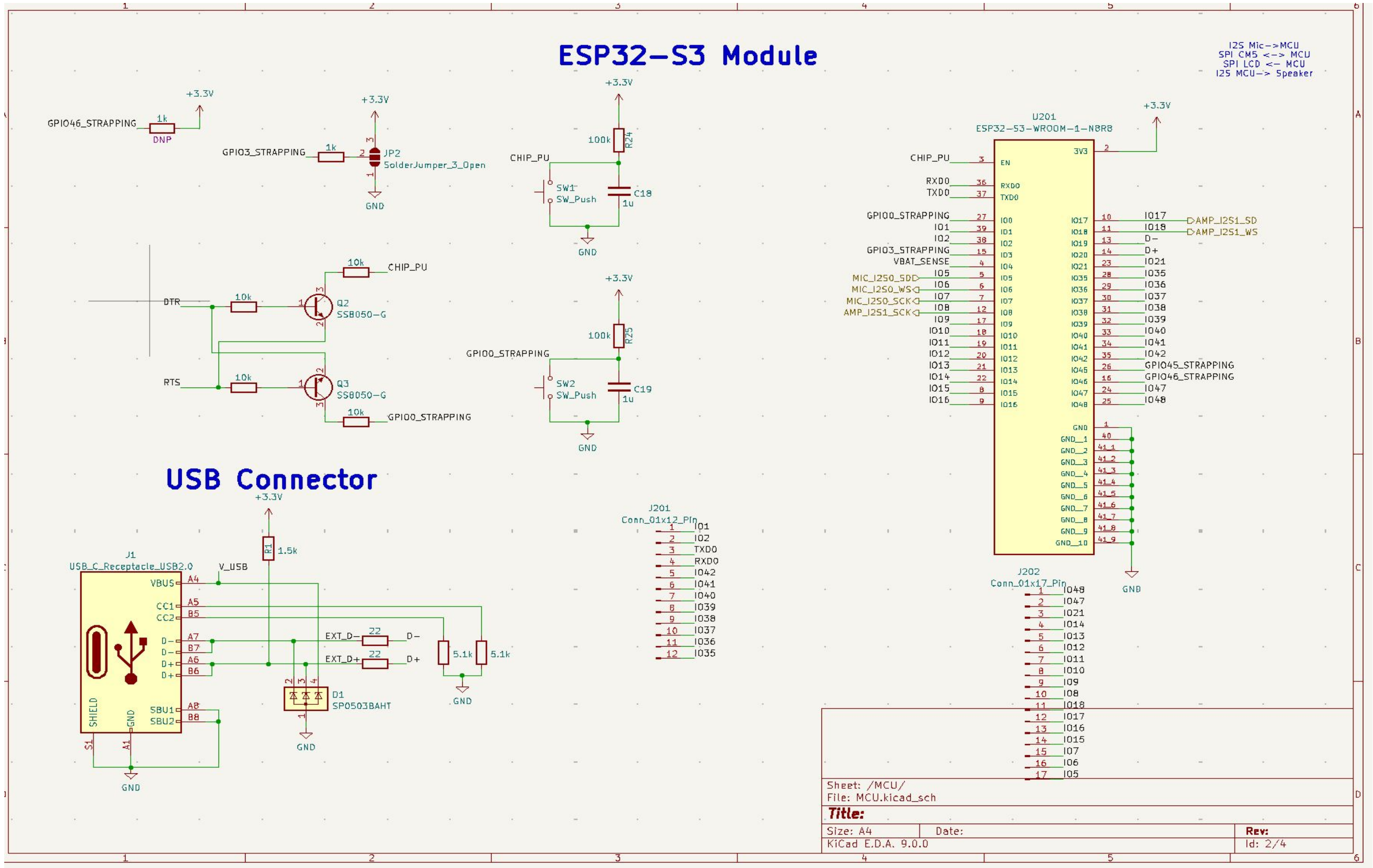
- Protect against reverse polarity
- Prevent damage during power-off conditions



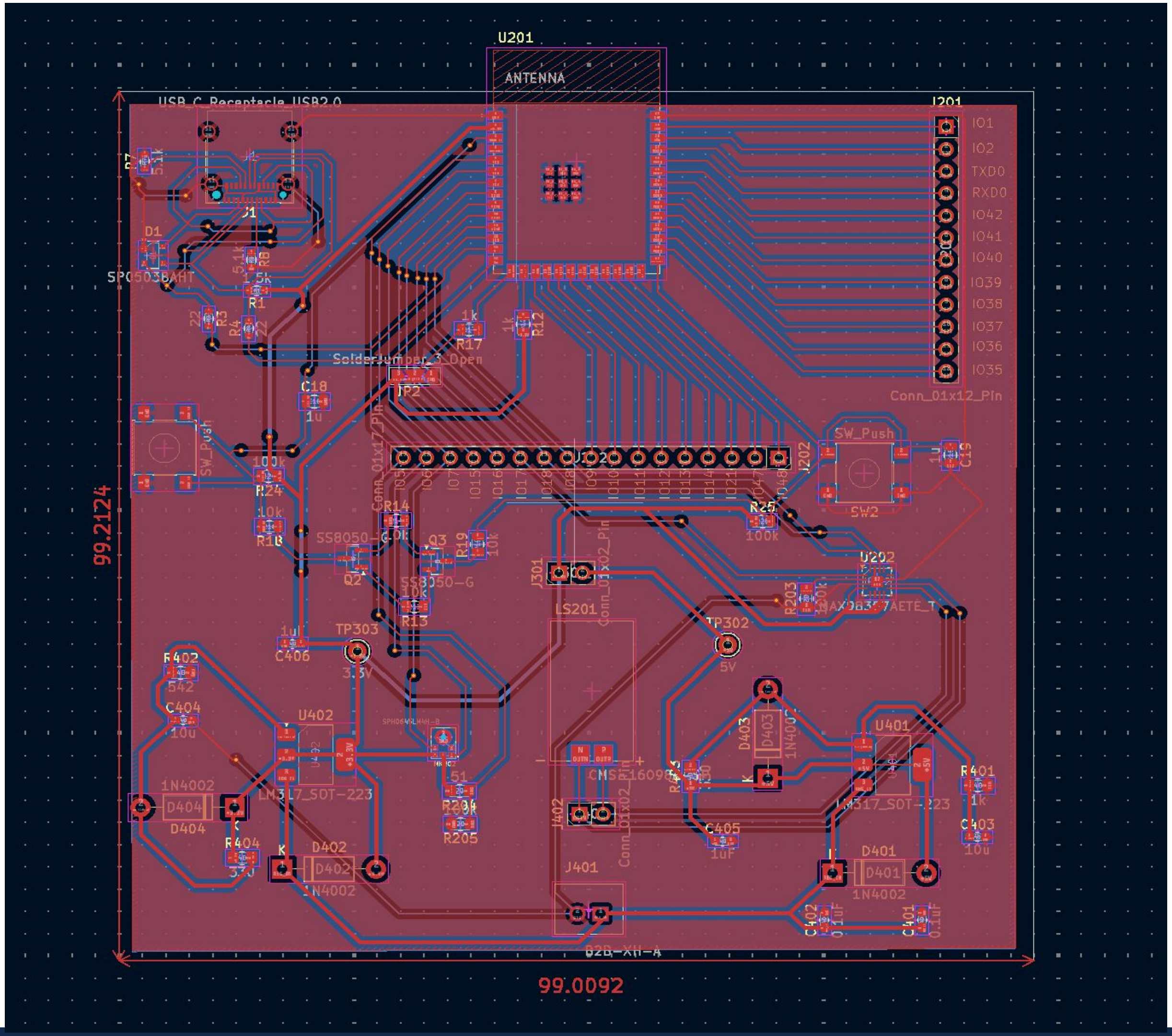
I/O (Audio) - Schematic



MCU Subsystem- Schematic



PCB Editor View



Problems Encountered

Problems Encountered

PCB Design/Setback:
- Boot-mode difficulties (Not Resolved)
- Issues using USB-C programming
- Resolved by using USB-UART bridge

Translation Latency:
- Prototyped Inference Pipeline on M2 Mac
  - End-to-End Latency: ~800 ms
- Initial Port of Pipeline on CM5
  - End-to-End Latency: ~2 minutes
- Resolved Issue by adjusting transcription model

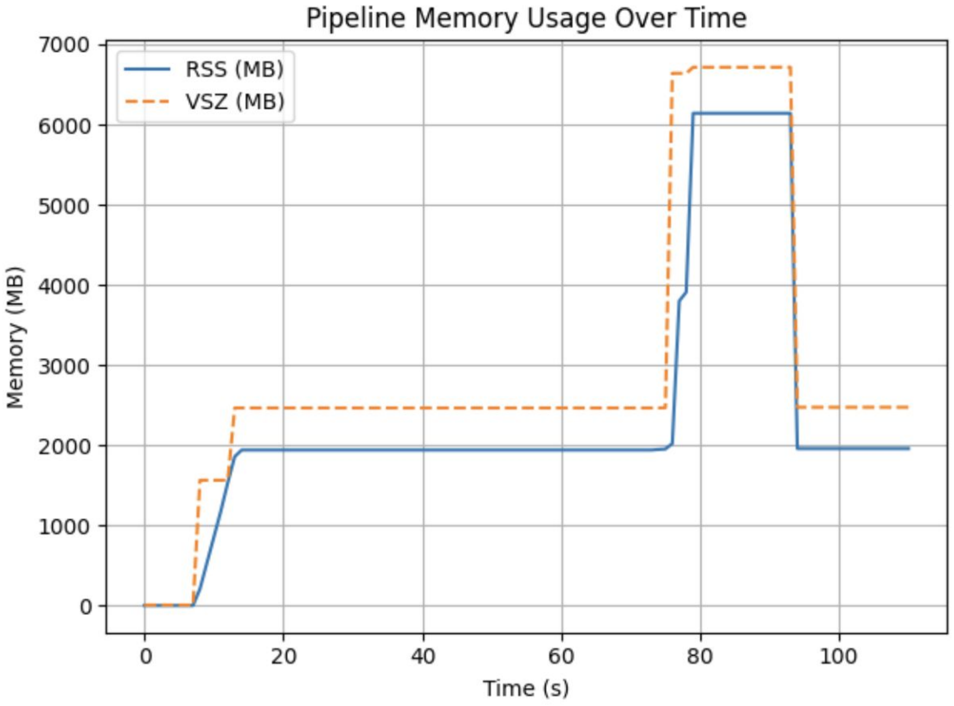
SPI Data Transfer:
- Reconstruction of audio file failed after hardware setup for ESP32 and CM5
- Speaker audio output noise was glitchy
- Resolved problems by adjusting SPI transfer time delay to align I2S and SPI

Video Demonstration

Video Demonstration

Results

Translation Latency - Memory Bottleneck



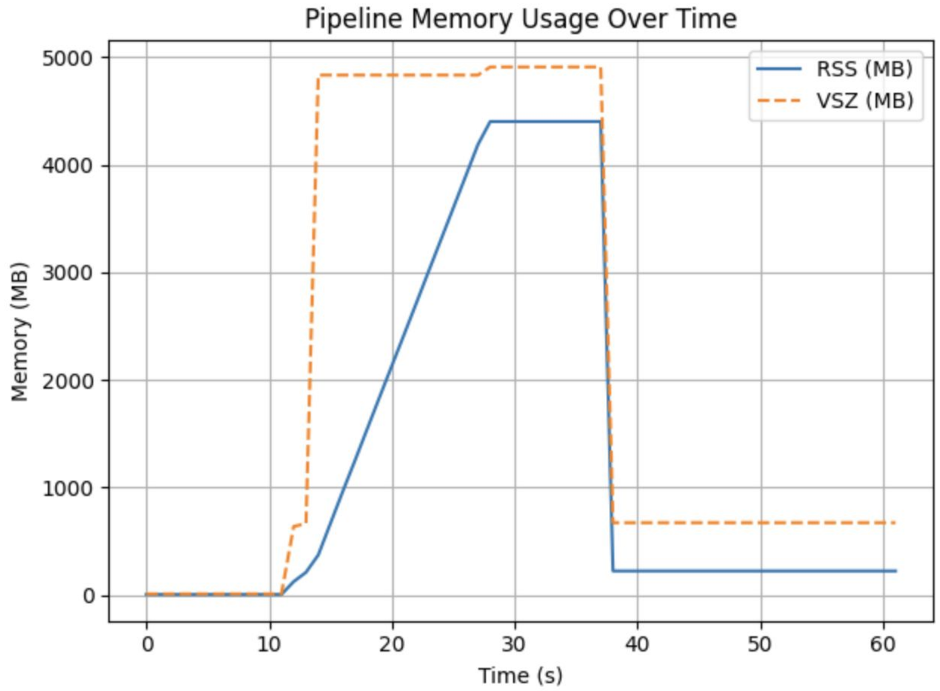
Configuration:

- llama.cpp: mistral-7b.Q4_K_M.gguf
- whisper.cpp: ggml-large-v3-turbo.bin

Results:

- Total Runtime: 90s
- Whisper Inference Duration: 55s
- Translation + TTS Duration: 15s
- Peak RSS: ~6300 MB
- Peak VSZ: ~6700 MB

Translation Latency - Memory Bottleneck



Configuration:

- llama.cpp: mistral-7b.Q4_K_M.gguf
- whisper.cpp: ggml-tiny.bin

Results:

- Total Runtime: 38s **(2.37x Faster)**
- Whisper Inference Duration: 19s **(2.89x Faster)**
- Translation + TTS Duration: 6s **(2.5x Faster)**
- Peak RSS: ~4500 MB **(28.6% Less)**
- Peak VSZ: ~4900 MB **(26.9% Less)**

Translation Accuracy

Configuration:

- llama.cpp: mistral-7b.Q4_K_M.gguf
- whisper.cpp: ggml-tiny.bin

Sample Text:

- *“Where is the library?”* (English -> Spanish)
  - **Transcribed Text:** *“Where is the library?”* (Levenshtein Similarity: 100%)
  - **Translated Text:** *“¿Donde está la biblioteca?”* (Semantic Similarity: 99.08%)
- *“What are the directions to the Illini Union?”* (English -> French)
  - **Transcribed Text:** *“What are the directions to the Illynei Union? ”* (Levenshtein Similarity: 95.5%)
  - **Translated Text:** *“Quelles sont les directions vers l'Union Illynei ?”* (Semantic Similarity: 90.9%)

Conclusion

Conclusion

- Built a portable, offline translator that performs real-time speech to speech translation
- Integrated embedded systems (ESP32-S3, Raspberry Pi CM5)
- Created inference pipeline (Whisper -> LLaMA -> Piper TTS)
- Resolved real-world bottlenecks with embedded machine learning
- **Next Steps:**
  - Rebuild translation code to work around a non-prompt based model
  - Expand storage capacity on CM5 to store more TTS models
  - Resolve PCB bring up issues


ILLINOIS