ECE 445
Spring 2025

Project #77: Portable Offline Translator

Team Members:

Lorenzo Bujalil Silva (lorenzo9)
Joshua Cudia (cudia2)

TA: Chi Zhang
Professor: Arne Fliflet

Introduction

Problem

Traveling is an exciting part of life that can bring joy and new experiences. Trips are the most memorable when everything goes according to plan. However, the language barrier can limit communication with others, causing unnecessary stress on an otherwise enjoyable trip. Although most modern phones provide translation applications, these require a reliable internet connection. In times when the connection is weak or there is no connection at all, translation apps may not be a solution.

Solution

We want to solve this problem by building a portable translator that you can ideally use anywhere in the world without internet connection. The idea is to have a small device that can be programmed to make translations between two different languages, then is able to listen what the person says, converts the speech to text, translates the text to the target language, then converts the translated text back to speech, and drives a speaker with the target translated speech. We want to design our translator to encompass a few subsystems: Main Processing Subsystem (MCU), Secondary Processing Subsystem (Compute Module), Audio Subsystem, User Interface Subsystem, Communication Subsystem, and Power Management Subsystem. Through this design, someone should be able to turn on the device, set the languages up and start talking into the device, and after a few seconds the translated speech will be played. Then, the device can be programmed the other way to have the other party translate. Ideally, this will facilitate communication between people without a common language and make life easier while traveling.

Visual Aid

A common representation of an embedded translation device can currently be seen on the market. However, we have noticed that many times this device will be sold as a package with a variety of functionality including online translation or photo translation.

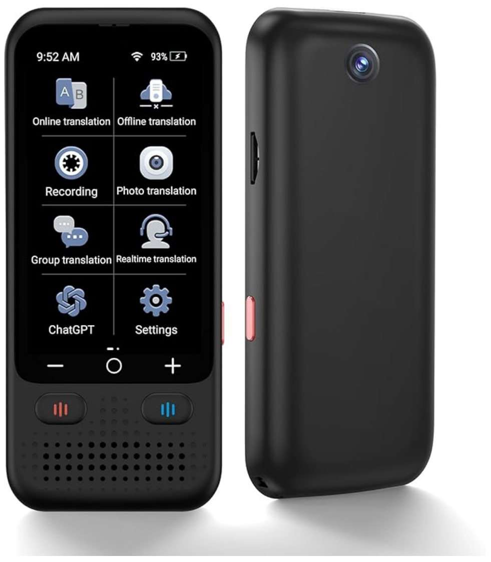

Figure 1: Retail Translator

This inclusion of functionality provides some better usage for the device, but will also significantly increase the cost. Ideally, all of this additional functionality would be provided by a phone that most people already have. The problem that we are setting to solve is one where the user will have no access to internet and need to be able to communicate. We want to design a device that is proficient in simplicity. Being able to clearly understand and translate to ensure proper communication when there is no other resources available.

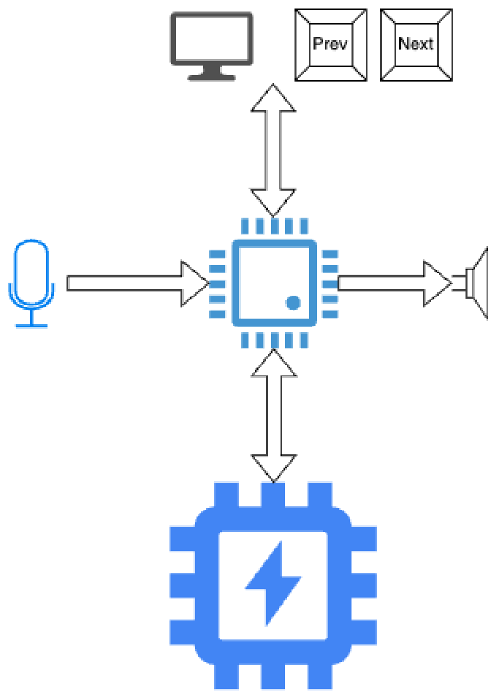

Figure 2: Visual Diagram

Our end product should appear as a hand-held device hosting a microphone to listen to user speech, a user interface with buttons and LCD display, main processing unit managing data flow and I/O, secondary processing unit managing translation, and finally a speaker to listen what was translated.

High-level Requirements

1. Translation Latency This system should be capable of translating spoken input to text and vice versa within 3 seconds to ensure real-time usability.
2. Translation Accuracy: This system should be capable of maintaining an accuracy of at least 90% for common phrases and vocabulary.
3. Speaker Noise: The speaker output should be clear and audible within typical decibel ranges of normal conversation.
4. Microphone Data Retrieval: The microphone should capture audio input within distances of 1 meter.
5. Data Integrity Across Pipeline Stages: This system should ensure efficient movement of data throughout the pipeline to to ensure at 100% of original meaning.
6. Translation Capabilities: This device should support up to 2 different language pairs.

Design

Block Diagram

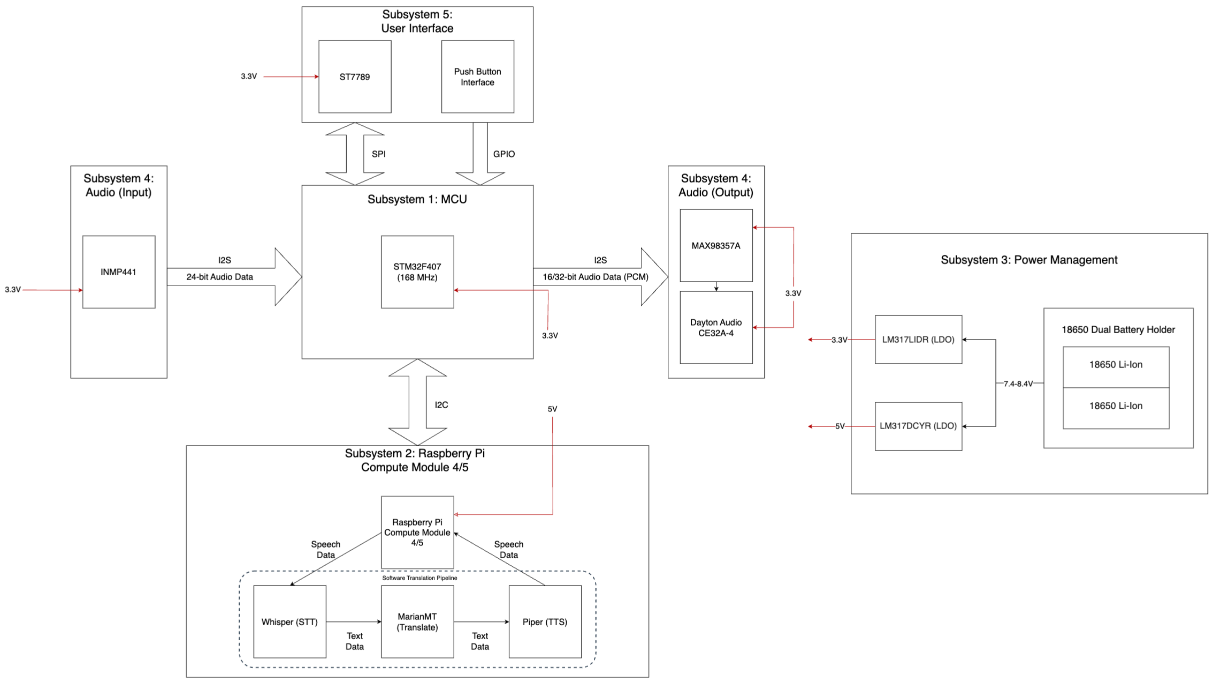

Figure 3: Block Diagram

Subsystem Overview

MCU (Main Processing Subsystem)

The main processing subsystem will manage the workflow for the system, control all I/O, and communicate commands/data to the secondary processing subsystem. When the system powers on a simple interface will be prompted to the user to allow them to select the source and target language to translate. The MCU will support the user inputs through a push button to select the language and will drive the display. Then when the user decides to start translation, through a particular push button, the MCU will change states to start listening on the port for audio data from the microphone. The INMP441 microphone will output a digital signal and communicate over I2S which can be interpreted through our MCU. The MCU will also need to buffer data and need to normalize it to be within the appropriate bit range to be interpreted by the STT model. After preprocessing the data, we will need to set up code to communicate packets of this data over a SPI or I2C protocol to the compute module. We also may need to set up some kind of custom protocol to set the compute module to start listening for a data sequence. Then the compute module will take over and do the translations and conversions to speech and output pulse code modulation data. This data is transmitted again over SPI or I2C to the MCU that is listening. Then the MCU will move to another state to start writing the data to the MAX98357A that will drive the speaker. Then the MCU will move back to a state of user input again to allow the user to translate again. Other than managing the entire workflow for the system, it needs to control the I/O which will include reading inputs from the user on the push buttons and will need to drive an LCD display to show what the user is currently selecting. With enough time, we may also add some status messages onto the LCD display to see what is happening in the system.

We decided to use the STM32F407 for this project because we required high levels of communication between various systems along with the numerous I/O. We also found that it has a LCD parallel interface and JTAG interface. We also have a long reach goal to do some audio manipulation (e.g. filtering, noise reduction) before sending it off to the compute module. We can also expect to support a real time control of the audio and peripheral management.

Subsystem Requirements: The main requirement of the MCU is to orchestrate the state of the system and manage data flow. We can quantify this impact by being able to buffer 10 s of speech data into the pipeline to be translated. Since it is also managing the flow of the system, the end to end performance should be measurable here to be under 500 ms. This would include the time between a user finishing speaking to getting speech out from the speaker.

Raspberry Pi Compute Module 4/5 (Secondary Processing Subsystem)

The main purpose of the compute module is to offload high compute tasks, including speech to text transcription, translation, and text to speech conversion. It would be too computationally complex to host all three models on the STM32 while processing I/O data. We specifically chose the RPI Compute Module because it has the computational power to run the AI models, it can interface over SPI or I2C to the MCU, it runs Linux, and it has eMMC Flash to store the models on board. For this subsystem, we are going to have to build an infrastructure around querying the models, reading and sending data to and from the MCU, and a data processing pipeline to move through different stages of translation. It will also need to have some level of state awareness to know what kind of translation is being requested. We hope to design the PCB so that we can simply plug in the compute module onto the PCB through pinouts.

We need to start to build a software program that will orchestrate the pipeline of data through various ML models that will perform transcription, translation, and speech synthesis. This will also need to support listening to the micro controller port to decide when it should start to process data. We will need to start with a general implementation of this data pipeline where we can initially use APIs to query models. Once we get this to work, we can figure out exactly how much memory and storage we will require from these models to determine if we need to do some quantization or add more storage space. This is to be able to have the models on board and queryable offline.

Subsystem Requirements: The main requirement of this processor would be to ensure appropriate translation of the data. Ideally this processor should have the software infrastructure to translate the data with 90% accuracy across each of the models. This would be measured across an ideal translation. We would also ensure that the data is translated within 300 ms.

Audio I/O Subsystem

The audio subsystem encompasses all of the components managing analog signals. Essentially this will include listening to the user voice then delivering that to the MCU. It will also include the amplifier and speaker that are managing converting the digital signals to an analog signal and then speaking out the translated speech. Additionally, the microphone will do analog to digital conversions and some amplification to the signal. To avoid conflict with noise from the speaker with the microphone, we are going to build a very synchronous system where we can only use the microphone or the speaker at a time.

Subsystem Requirements: The main requirements of this system would be accuracy in retrieving audio data and delivering understandable speech on the speaker. We would hope to have 95% accuracy in retrieving data from the microphone and driving the speaker to be understood by a person at around 60 dB.

User I/O Subsystem

We are going to have a LCD display that can let the user decide what languages to translate between and some push buttons to be able to decide. We are also going to have a push button that will start listening on the microphone, then stop listening so we can ensure that all of the data has been stored. In the case that the translation lasts too long, we may add some feature to automatically stop the input of speech so we make sure not to have too much data to translate. This UI subsystem will essentially make it so that this is a usable product.

Subsystem Requirements: The main requirements of this device would be able to have a usable interface to be able to select across languages and see the selection on the screen. We hope to achieve a refresh rate on this LCD display of at least 30 hz.

Power Management Subsystem

This portable power management system will use a Samsung 25R 18650 2500mAh 20A rechargeable Li-Ion battery to supply stable voltage to two power rails. We will use the 5V power rail for the Raspberry Pi Compute Module (CM) and 3.3V power rail for the MCU, LCD, and audio subsystem. This system includes two LM317DCYR adjustable LDO regulators. One will be used to step down the battery voltage to 5V and the another to step down the voltage to 3.3V. Lastly, the 18650 battery holder securely holds the battery and enables easy replacement.

Subsystem Requirements:

- Voltage Stability:
  - The 5V and 3.3V power rail should maintain ±0.1V tolerance to their respective subsystems
- Current Supply:
  - The 5V regulator should supply at least 1.5A continuously to support worst case current draw from the Raspberry Pi CM4/5.
  - The 3.3V regulator should supply at around 400mA to ensure the functionality of the MCU, LCD, and audio subsystem.
- Thermal Management:
  - The LM317 regulators should not exceed 85°C during normal operation and heat dissipation methods should be used to prevent overheating.
- Protection Mechanisms:
  - Overcurrent protection should be in place to prevent system damage if a component draws excessive power (ie. PCB fuses).

Tolerance Analysis

Translation Accuracy

Within our translation accuracy, we hope that our system is able to fully translate full sentences with a confidence of accuracy that is higher than 90%. This will be the entire pipeline that needs to process this data and maintain data integrity to reach the output. This also needs to be driven to a decibel level that is able to be interpreted by an end user. We also need to account for possible noise that we may pick up at the input and filter out that data to be more tolerable. We expect that this filter will ensure our accuracy.

Translation Latency

We hope to have our translation latency to be within 500 ms to ensure that the system is able to be simply used by a person in a remote location with simple and quick use. We will be limiting buffers within our design to ensure that data will be processed within the time frame and reduce possible noise in latency spikes.

Power Management

In order to ensure efficient power management, we have included "worst case" current draw for the different voltage domains.

Component (3.3V)	**Peak Current Draw (mA)**
STM32F407IGH7 (MCU)	240.0
INMP441 (I2S Microphone)	1.6
ST7789 (LCD Display)	90.0
Total	331.6
Component (5V)	**Peak Current Draw (mA)**
Raspberry Pi Compute Module 4	1400.0
MAX98357A (Amplifier)	4.0
Dayton Audio CE32A-4 (Speaker)	2.75
Total	1406.75

Ethics and Safety

Open Source Usage

Our project is based on various open source projects. We have adapted these projects to meet our specific needs. We acknowledge our responsibility to properly attribute the original authors and comply with licensing requirements. We will ensure that all intellectual property is properly cited and comply with their licenses.

Battery Safety

To comply with industry standards, including IEEE 1725-2021 for rechargeable battery safety, we will incorporate protection mechanisms against overcharging, overheating, and short circuiting.

Translation Misuse

Our project can be prone to the sensitive nature of translating sensitive language which could cause harm to others. We intend to censor this language as much as possible to prevent from any possible misuse of this device. We intend to develop a table of commonly used sensitve words and censor appropriately if a particular word is found.

References

- Marian MT Translation Model
- Piper TTS Model
- Whisper STT Model
- Raspberry Pi Compute Module 4/5 Datasheet
- STM32F407 Datasheet
- INMP441 Datasheet
- MAX98357A Datasheet
- Dayton Audio CE32A-4 Datasheet
- ST7789 Datasheet