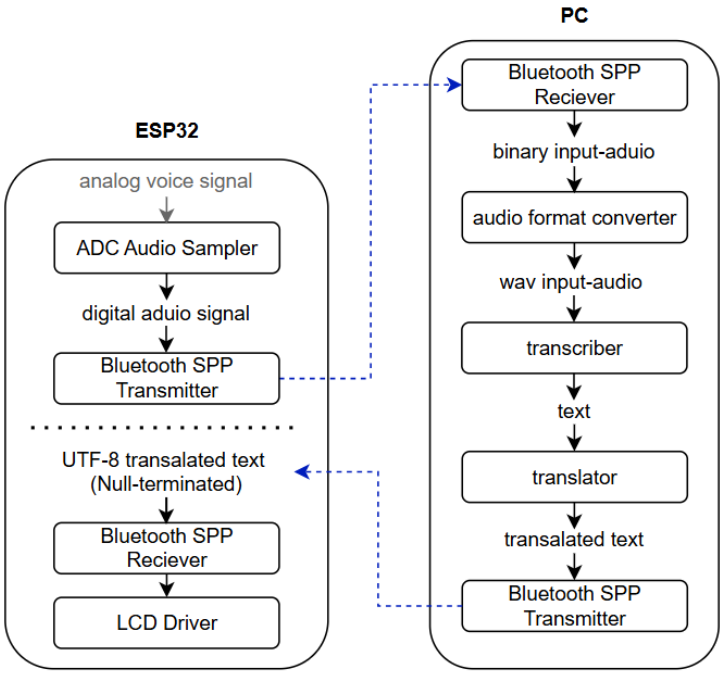
Translator Design & Schedule

Object-Oriented Design

Principle Use Case

Speech translated & displayed as text.

1. A single speaker speaks into the microphone (MAX9814).
2. The ESP32 samples the analog audio signal, samples it through its built-in ADC, and sets the digital audio sample via Bluetooth SPP to the PC.
3. The PC receives the raw audio data, and converts it to a processable format.
4. The PC translates the text into the target language (currently English only) and sends it to the ESP32 over Bluetooth SPP.
5. The ESP32 receives the translated text and displays it on the LCD screen.



Design structure inspired by Tamas-Selicean & Lorincz (2023)
https://www.mdpi.com/3075472

Principle Objects

Interface Key:
- Input
- Output

ESP32:

- **ADC Audio Sampler:** Captures analog signal from the microphone (MAX9814) and converts into digital audio samples using ESP's ADC hardware.
- **Bluetooth SPP Transmitter:** Wirelessly sends digital audio samples from the Audio Sampler as binary data over Bluetooth Classic Serial Port Profile (SPP) to the PC.
- **Bluetooth SPP Receiver:** Forwards received (translated) text received to the LCD driver.
- **LCD Driver:** Receives translated UTF-8 text and displays it on attached LCD screen.

PC:

- **Bluetooth SPP Receiver:** Receives incoming Bluetooth SPP data stream containing ESP32's audio samples. Passes binary audio to the audio format converter.
- **Audio Format Converter:** Converts raw binary audio samples into a processable format like WAV for speech recognition.
- **Transcriber:** Performs speech-to-text on the WAV input.
- **Translator:** Translates the transcribed text into the target language, also in text (just English for now).
- **Bluetooth SPP Transmitter:** Sends translated text over Bluetooth SPP to the ESP32 as null-terminated text.

Project Management

Deliverables

Effort is estimated from 1 (low) – 5 (high).
Asterisks mark tasks that are excluded from weekly milestones. Those will be prioritized before stretch goals.

[D] Documentation	**Priority**	**Effort**
1. Hardware/software decisions.*	LOW	2
2. User Guide.	MED	2
3. Weekly Log.*	LOW	1
4. Bluetooth protocol.*	LOW	2
5. Demonstration	HIGH	4

[H] Hardware Setup	**Priority**	**Effort**
1. ESP32 wiring w/ MAX9814 microphone and LCD screen.	HIGH	3
2. Power supply.	HIGH	1

[E] Embedded Programming	**Priority**	**Effort**
1. Use ESP32's ADC to sample audio from the mic.	HIGH	3
2. Transmit audio over Bluetooth SPP to the PC.	HIGH	4
3. Receive translated text (currently English-only over Bluetooth SPP.	HIGH	4
4. Display translation on the LCD screen.	HIGH	2

[C] PC Software	**Priority**	**Effort**
1. Receive binary audio samples over Bluetooth SPP.	HIGH	3
2. Convert raw audio to processible format (WAV).	MED	4
3. Transcribe speech-to-text (use libraries).	MED	3
4. Translate transcribed text (use API).	MED	2
5. Send translated text to ESP32 over Bluetooth SPP.	HIGH	4

[S] Stretch Goals	**Priority**	**Effort**
1. On/Off button.*	NO	1
2. A2DP streaming test.	LOW	5
3. Language selection option.*	NO	3
4. Speaker differentiation.*	NO	5

Weekly Milestones

Tasks are ordered by dependencies (planning before execution) and priority (hardware & embedded before PC-side software). They are distributed by estimated effort. 41 effort points over a ~4-week period means a goal of ~10 points per week. The first and last weeks will be assigned less because of cutoff. There is no room to provide a safety buffer at the end.

Week 1 (May 11 – 17):

- Milestones: Planning and hardware setup.
- Deliverables: D2, H1, H2
- Points: 2, 3, 1 = **5**

Week 2 (May 18 – 24):

- Milestones: ESP32 Audio sampling and binary transmission over Bluetooth.
- Deliverables: E1, E2, C1
- Points: 3, 4, 3 = **10**

Week 3 (May 25 – 31):

- Milestones: ESP32 receiving text over Bluetooth and displaying on LCD.
- Deliverables: C5, E3, E4 (PC just sends confirmation that data has been received, no processing yet)
- Points: 4, 4, 2 = **10**

Week 4 (June 1 – 7):

- Milestones: PC-side processing and translation. Replace transmitted confirmation with translation.
- Deliverables: C2, C3, C4
- Points: 4, 3, 2 = **9**

Week 5 (June 8 – 9):

- Milestones: Wrap-up and Demonstration.
- Deliverables: D5
- Points: **4**

Contingency Plan

The project priorities are the hardware and embedded programming aspects. In the event of the project falling behind, the PC side software will not be completed. The contingency plans depend on the issue:

- Light schedule delay: The PC will not translate the text and just return the transcription.
- Med schedule delay: The text won't be transcribed or converted into a processable format. The PC just returns confirmation message that data has been received.
- Poor audio quality: Skip data processing and translation attempts. Still implement ESP32<->PC communication, such as a "ping"-"pong".
- PC does not receive data: Periodically transmit "ping" messages.
- ESP32 does not receive data (severe): Try replacing the PC with a Jetson Nano or RPi OR use USB serial instead as a backup transfer method.

In the scenario that the project is ahead of schedule, the next task will be started.

Resources

Components

- ESP32 DEVKIT V1 - on hand
- MAX9814 Microphone Module - on hand
- LCD Display module - on hand
- (Windows) PC - personal property
- USB-A to Micro-USB Cable - on hand
- Jumper Cables - on hand
- Breadboard - on hand

Software

- Arduino IDE
- Python or C/C++ for PC-side programs
- Speech-to-text APIs (to be identified by Week 4)
- Translation APIs (to be identified by Week 4)

Labor / Schedule

~10 effort points/week are assigned to 1 person. Main bottleneck will be time.
Since the project has a linear task schedule and no division of labor, the schedule is fully provided in the Weekly Milestones (it's already ordered by dependencies, priorities, and then effort).

User Manual

The project is an absolute-minimum viable translator, and will not require any UI or configurations.

To use the system, the user will need a computer that supports Bluetooth Classic, and connect the ESP32 to the computer via USB to provide power. Once the ESP32 is powered on, the user runs the PC-side program using the executable, and the system will begin automatically.

To use the translator, the user needs to speak into the microphone. The ESP32 chunks the audio, which the PC assembles and is responsible for transcribing it and detecting a sentence end. Once it does, it sends the translated (English) sentence, which the ESP32 will receive and display on the connected LCD screen.

In the case that the screen is full, lines will be shifted up to make room for the new text.