ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

AMADEUS

Augmented Modular AI Dialogue and Exchange User System

Team No.33

Qiran Pang (qpang2@illinois.edu)

Chengyuan Peng (cpeng14@illinois.edu)

Ryan Fu (ryfu2@illinois.edu)

TA: Jason Zhang

Professor: Cunjiang Yu

October 2, 2024

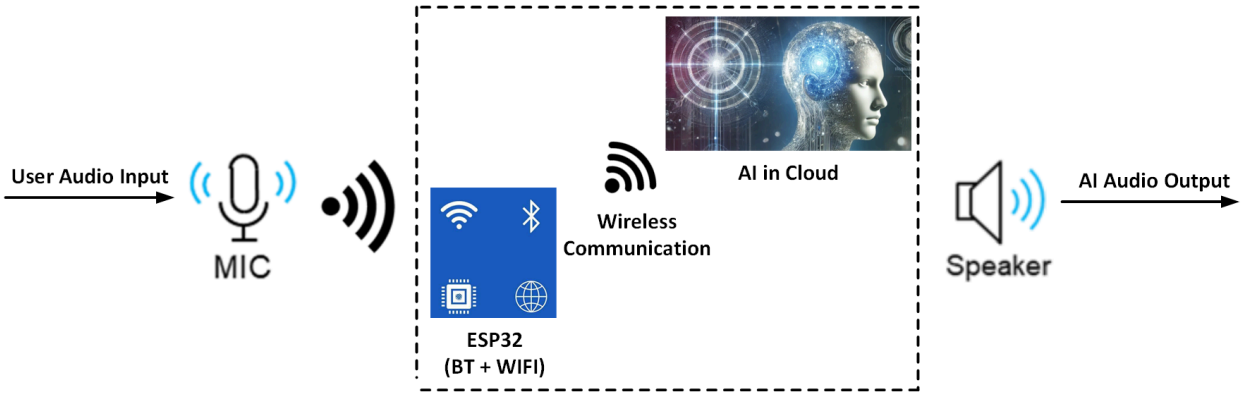
1. Introduction

Problem:

People have envisioned engaging in natural, conversational interactions with robots for many years to fulfill emotional and lifestyle needs. However, most current interactive AI systems are either too bulky or rely heavily on smartphones, detracting from the organic nature of such interactions. A more tangible, interactive medium—such as a child talking to a familiar toy or a headset with built-in AI—would offer a more immersive experience, which corresponds to an increasing market need[1]. Embedding a trained AI model in each toy or device would be cost-prohibitive since it would require powerful and expensive embedded computers. To address this, we propose leveraging cloud-based AI models, such as ChatGPT or similar character-driven AI, in our embedded system, which can process data remotely and send responses back to the device in real time.

Solution:

We aim to develop an AI-based audio interactive interface, housed on a custom-designed PCB. This system will capture audio from the user, transmit it via Wi-Fi to a cloud-based AI model for processing, and play the AI's response back to the user. The ESP32 microcontroller, equipped with Wi-Fi and audio input/output capabilities, will serve as the core of our system.



High-Level Requirements:

- **Response time:** The AI model should receive audio input from the user within **5 seconds**, process it, and send a response back to the PCB within **5 seconds** (response time may vary depending on the chosen AI model and internet speed).

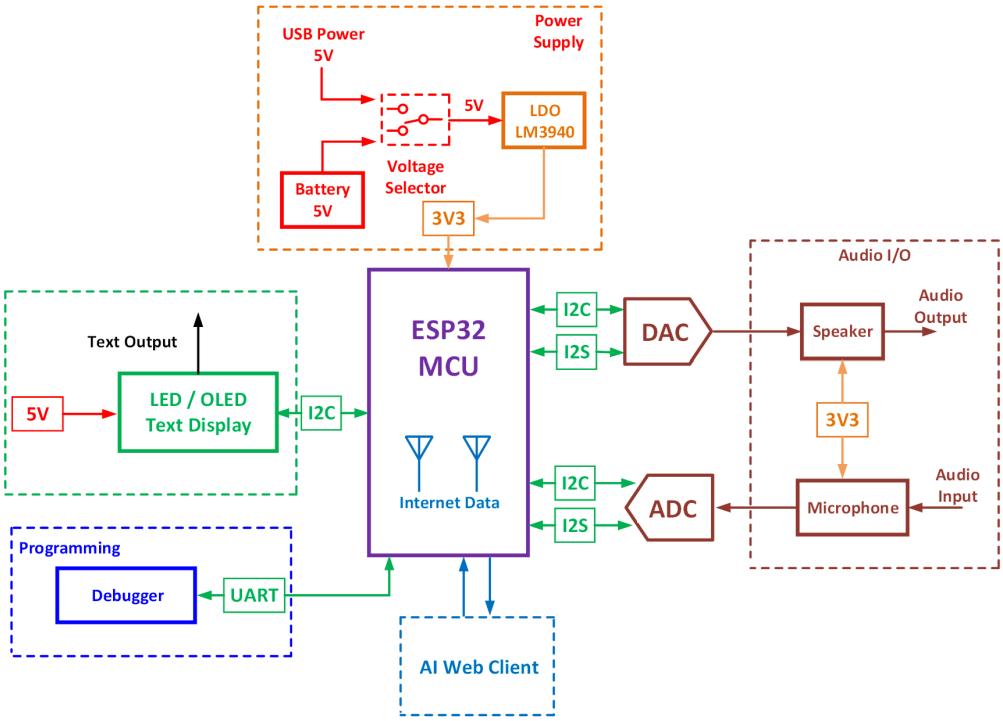
- **Voice Clarity**: The AI-generated audio must be **clear and audible** to the user, with a Signal-to-Noise Ratio (SNR) larger than 20 dB.
- **Multi-language support:** The system will support voice input in **three different languages**: **Chinese, English, and Japanese**.

Additional features:

- **Indoor and outdoor modes**: In outdoor mode, audio input will be processed only when a button is pressed, and the system will apply noise reduction to improve voice clarity.
- **Headphone/Bluetooth integration**: This feature will allow users to interact with the device using wireless headphones or earbuds.
- **Text Display**: The PCB will include a small display to show transcribed audio, offering a visual representation of conversations for users.

2. Design

Block Design:



Subsystem Overview & requirements:

Subsystem 1: AI Web Client

Overview:

Our language model will be hosted on a cloud-based server. The local MCU will transmit audio to the server via a WiFi module. We are collaborating with a local start-up that will provide some AI models[2] for us. However, we also have the option to train our own AI model to create additional characters using their interface or connect with other available AI models online such as ChatGPT.

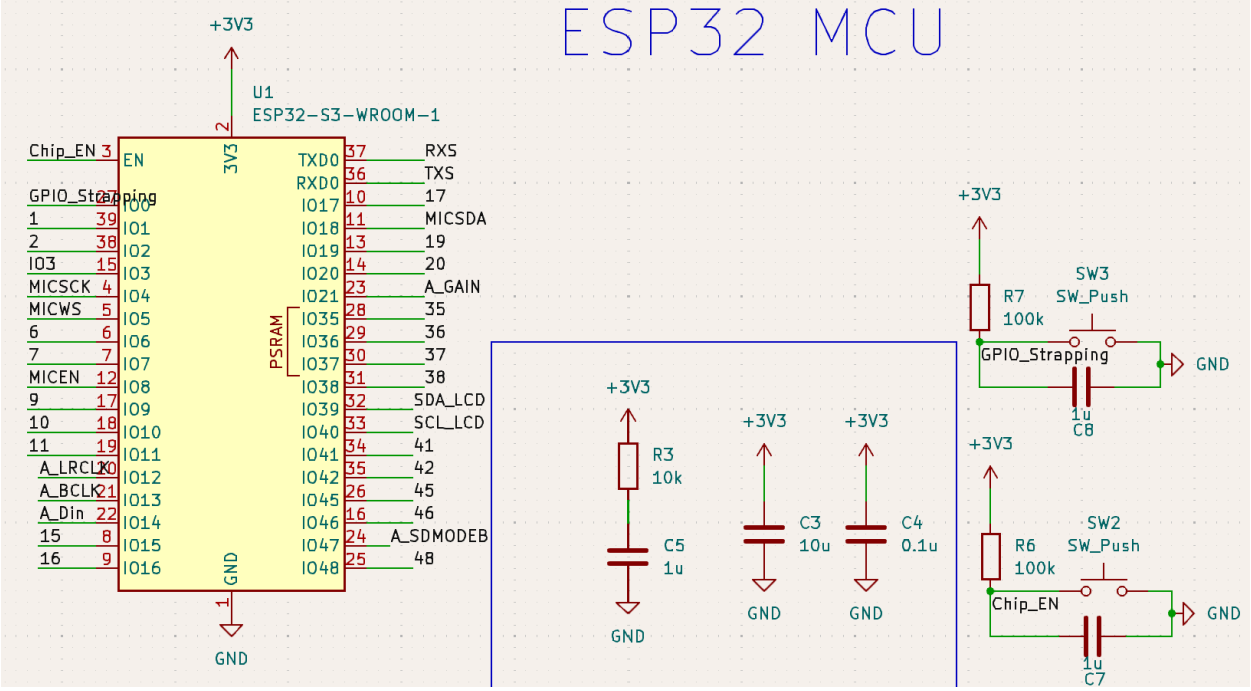
Interaction:

The AI web client mainly interacts with the WiFi module of the ESP32 board to receive and send audio and text signals.

Requirement:	Verification:
- The AI should respond with an average latency of **no more than 10 seconds** for a 10-second audio input, ensuring the user receives the reply promptly.	1. Send **various 10-second** audio inputs to the AI web client. 2. **Measure the time** from when the audio is sent to the AI until the response is received. 3. Conduct **at least 10 tests**, calculate the **average** response time, and ensure it meets the 10-second limit.
- AI models should support language inputs in **English, Chinese, and Japanese**.	1. Perform tests with audio inputs in **English, Chinese, and Japanese**. 2. Confirm that the AI accurately **understands** the input audio in different languages and responds **accurately**.

Subsystem 2: ESP32 with Wifi Capability

Overview:



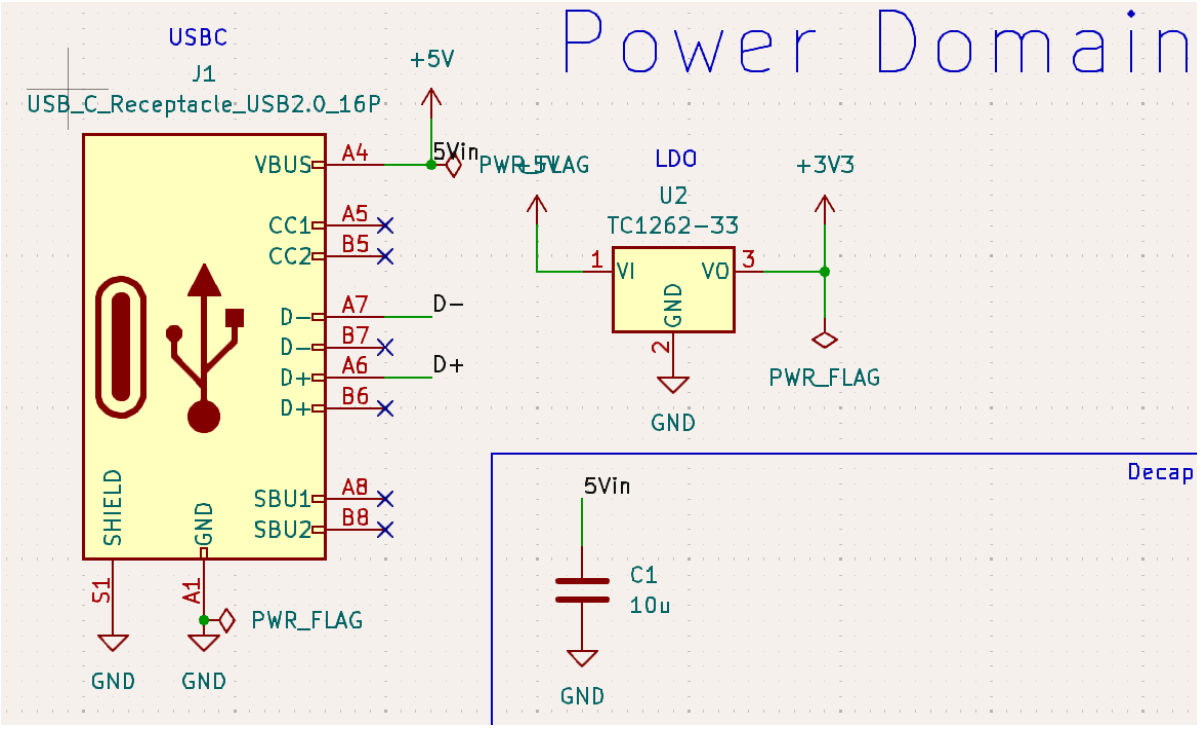
The ESP32 with Wi-Fi capability serves as the core processing unit for the entire system. It receives audio input from the microphone via the ADC, transmits the audio to the cloud-based AI model via Wi-Fi, receives the processed audio response from the AI model, and then sends the output to the audio codec for playback. This subsystem is also responsible for processing the buffer data in serial communication and compressing them according to the requirement of the server.

Interaction:

The ESP32 is the core of the system so it needs to interact with all other subsystems: audio I/O system by receiving and sending audio signal through I2C and I2S, AI web client through WiFi, LCB display to display text through I2C, debug module to be debugged through UART, and power supply system to receive 3.3 V power.

Requirement:	Verification:
- The ESP32 must establish a Wi-Fi connection **within 10 seconds of receiving valid credentials** and maintain a stable connection with >99% uptime during operation.	1. Input valid Wi-Fi credentials into the ESP32. 2. **Measure the time** from when the credentials are sent to when the ESP32 successfully connects. 3. **Repeat this test 10** times to ensure consistency. The average time must be ≤ **10 seconds**.
- The ESP32 must maintain a stable Wi-Fi connection with **>99% uptime** during operation.	1. Run the system continuously for at least **24 hours while connected to Wi-Fi**. 2. **Monitor** and log any connection drops during this period. 3. Calculate the total uptime and ensure it **exceeds 99%.**
- The ESP32 must transmit and receive audio data at **a minimum bitrate of 64 kbps** to ensure acceptable audio quality.	1. Measure the actual **bitrate** during audio transmission using Wi-Fi. 2. Perform **multiple tests** with varying audio inputs. 3. Confirm that the bitrate **consistently meets or exceeds 64 kbps**.

Subsystem 3: Power System



Overview:

The system can be powered through either a USB connection or a 5V battery. The 5V supply directly powers the I/O devices and the programming module. To provide 3.3V power for the microcontroller and audio processing module, an LDO voltage regulator is used to step down the voltage.

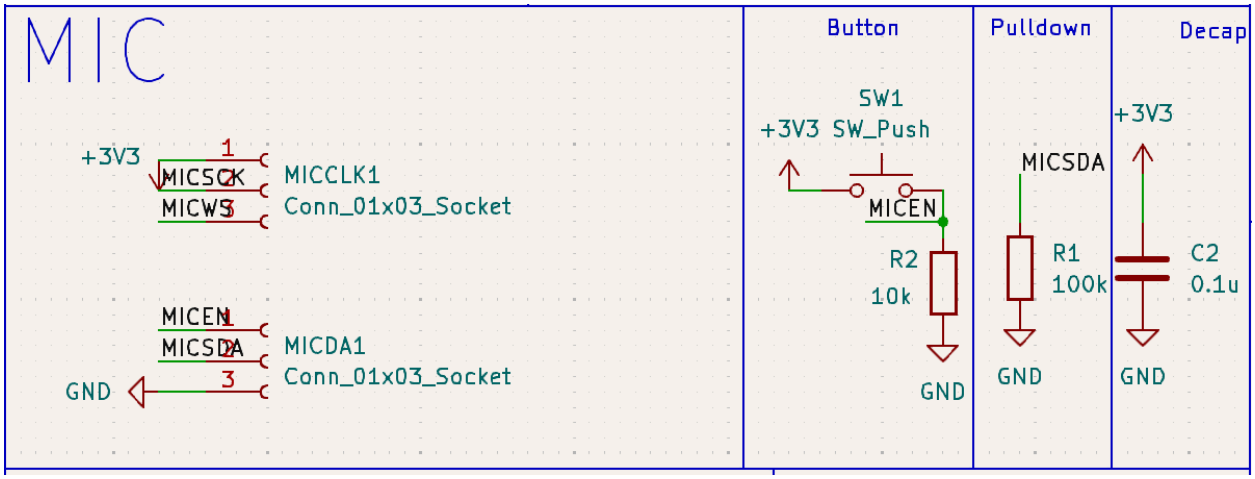
Interaction:

The power supply system interacts with other subsystems by providing Vcc power to other components, such as the ESP32 core and audio I/O processor.

Requirement:	Verification:
- The power system must be able to use a **5V power supply** source and **power all the modules** in our system with their	1. Connect the **5V power** supply to the system. 2. **Measure the output voltage** provided to each module (ESP32, audio I/O, etc.).

respective power requirements (5V or 3.3V).	3. Confirm that the appropriate modules receive **5V or 3.3V** as per their power specifications.
- The power system must keep a **stable** power output for all the modules under any working environment	1. Use an oscilloscope to monitor the **output voltage** provided to the modules. 2. Measure the output voltage of the power system after programming the board with programs with different loads. 3. Reading oscilloscope to make sure it remains stable within ±5% of the target voltage.

Subsyste 4: Audio I/O & Processing



Overview:

The Audio I/O & Processing subsystem is responsible for capturing the user's voice through the microphone, converting the analog audio signals into digital form using an ADC, processing these signals, and then sending the digital audio to the ESP32 for transmission. Once the AI response is received, it is converted from digital to analog form using a DAC or Audio Codec, and the output is played through a speaker.

Interaction:

The Audio I/O & Processing subsystem interacts on one side with the user by receiving audio from the microphone and playing audio through the speaker and interacts with the ESP32 microcontroller through I2C and I2S to communicate and send signals. In addition, it also receives power from the power supply module.

Requirement:	Verification:
- The microphone must have a sensitivity of **at least -42 dBV** and a **frequency response range of 20 Hz to 20 kHz** to capture a full range of human speech clearly.	1. Measure the microphone sensitivity using a calibrated sound source and verify it **meets or exceeds -42 dBV.** 2. Use a signal generator to input audio frequencies across the **range of 20 Hz to 20 kHz.** 3. Record the output and verify that the microphone captures the **full range of frequencies** with consistent performance.
- The audio processing circuit must maintain an **SNR of at least 20 dB** to ensure clear audio input and output.	1. Input a known audio signal into the system and **measure** the **signal-to-noise ratio (SNR)** at the output. 2. Ensure that the measured SNR is **at least 20 dB** during normal operation. 3. Perform this test under **different environmental conditions** (quiet and noisy environments) to confirm consistent performance.

Subsystem 5: Text Display

Overview:

If we have more time after finishing the baseline, an additional feature of our project will be a text display LCD. After the audio input / output are converted into texts, the LCD screen attached to the microprocessor will display the text output. It will ideally display both the input from user and output from AI on the LCD screen so users can make sure their audio is identified correctly while reading the response from AI when they did not hear the audio clearly.

Interaction:

Through I2C, the LCD text display interacts with the ESP32 microcontrollers by receiving a text to be displayed on the screen for the user to see,

Requirement:	Verification:
- The LCD resolution must be at least **128x64 pixels** so the user can see the text	1. Verify the resolution of the LCD by checking the **datasheet and specifications** provided by the manufacturer. 2. Display a predefined test pattern on the screen and confirm that the resolution is **at least 128x64 pixels.**
- The LCD should display the text output within **2 seconds** of receiving data, ensuring synchronization with audio playback.	1. Input text data into the system and start **a timer** when the data is sent to the LCD. 2. **Measure the time** it takes for the text to appear on the LCD screen. 3. **Perform at least 10 tests** to confirm that the text is consistently displayed **within 2 seconds**. 4. Verify that the LCD text display is **synchronized with the corresponding**

	audio playback.

Subsystem 6: Debug Module

Overview:

The debug module will consist of a debugging serial port and a programmer. The serial port will be temporarily integrated into the PCB for debugging the output from the ESP32 processor through UART. Additionally, a programmer will be connected to the MCU for programming purposes through USB.

Interaction:

The debug module is used by the user to debug and program the ESP32 microcontroller through UART and USB.

Requirement:	Verification:
- The debug module must support flashing programs to the board through a USB connection.	1. Connect the ESP32 board to a computer via the USB interface. 2. Attempt to flash a sample program (e.g., a simple LED blink program) onto the board using the flashing software. 3. Verify successful flashing by checking the functionality of the flashed program (e.g., confirm LED is blinking as programmed). 4. Repeat the flashing process **at least 5 times** to ensure consistent and reliable programming capability through USB.
- The debug module must support UART communication at **115200 bps**, with error	1. Set the UART communication speed to **115200 bps** on both the ESP32 and the PC. 2. **Transmit a series of test data packets**

detection and handling to ensure reliable data transmission.	through the UART interface. 3. Verify that the data is **received correctly** on the PC without errors using a terminal emulator. 4. **Perform multiple transmission** tests over a long duration to ensure consistent and reliable communication.

Tolerance Analysis:

A common challenge that embedded system designers frequently encounter is insufficient storage space of the processor, especially if the system is related to acoustics. The audio files will typically be large enough to occupy a large portion of the flash memory.

Suppose a 30s audio data is sampled at a rate of 44.1kHz, memory overflow can easily happen:

Along the 30s duration, the total number of samples will be:

$$30 \times 44.1k = 1323$$

Also, suppose each sample is of type int (4 bytes), the total number of bytes occupied by this sample will be:

$$1323 \times 4 = 5292 = 5.16 \text{ MB}$$

As there will be both audio inputs and outputs that are processed by the MCU, the total size of the two audio samples will be $5.16 \times 2 = 10.32$MB. 10MB is an incredibly large size to be processed – even a modern laptop cache can hardly meet this requirement. Not to mention ESP32 processors are much less powerful than a complete computer system.

Given that the best ESP processors only have an internal memory of 4MB with half of the storage already occupied by built-in libraries, we will have very limited space to store the audio inputs and outputs. As such, we must have a smarter implementation to reduce the sizes of the audio samples. We have come up with the two following solutions:

1. Instead of storing the entire file in the flash, we could use the flash as a buffer. In particular, we plan to use the flash as a buffer, only to store 5ms of the audio each time and send it to the cloud. In that case, the buffer only needs to use about 30b for the audio storage.
2. A 32-bit audio sample is way more than enough to produce an audible audio output, hence we should not waste our memory on unnecessary data. It is likely that we can compress all int32

audio samples into int8 forms, thus decreasing the data size from 10.32MB to 2.58MB. 2.5MB is sufficient to be stored into a MCU without any external memory components.

The potential risk of this project is that ESP32's flash size is 2 - 4 mb depending on the model of the chip, which is relatively small for storing audio files locally. For example, it takes 600kb to store a 10s wav file, which infers that if we receive a 1-minute audio file, its size will exceed the storage of the flash. Consequently, instead of storing the entire file in the flash, we use the flash as a buffer. In particular, we plan to use the flash as a buffer, only to store 5ms of the audio each time and send it to the cloud. In that case, the buffer only needs to use about 30b for the audio storage.

3.Cost and Schedule

After reviewing salaries for internships and roles in embedded systems development, we determined an hourly rate of approximately $40. With 10 weeks spent on the project at 6 hours per week, the total labor cost amounts to $7,200.

In addition to the labor cost, the total cost of all components of our design is $51.347:

Part Number	Functionality	Quantity	Unit Price
ESP32-S3-WROOM-1	**ESP32 Microprocessor**	**2**	**$3.20**
TC1262 - 33	**5V - 3.3V LDO**	**3**	**$0.69**
USB4085	**USB-C Receptacle**	**3**	**$1.15**
INMP441	**On-board Microphone**	**4**	**$8.4**
ICS-43432	**On-board Microphone**	**3**	**$8.76**
MAX98357A	**Digital Audio Amplifier**	**1**	**$6.5**
LCD 1602	**16 * 2 LCD Display**	**1**	**$3.03**
OLED 128x64 1.3" I2C	**128 * 64 OLED Display**	**1**	**$7.08**
333028	**UART Serial Port**	**1**	**$5.96**
SP-1504	**Speaker**	**1**	**$2.03**

0805_2012_10u	**10u SMD Capacitor**	**15**	**$0.10**
0805_2012_1u	**1u SMD Capacitor**	**10**	**$0.08185**
0805_2012_0.1u	**0.1u SMD Capacitor**	**10**	**$0.05885**
0805_2012_100k	**100k SMD Resistor**	**5**	**$0.10**
0805_2012_10k	**10k SMD Resistor**	**10**	**$0.10**
0805_2012_4.7k	**4.7k SMD Resistor**	**10**	**$0.10**

Schedule

Week	Chengyuan	Wesley	Ryan	Due
9/30	**Implement basic communication code with OpenAI API**	**Implement basic communication code with AI API**	**PCB schematic design and component research**	**Design Document - TR Proposal Regrade - F**
10/7	**Setting up the base for the server**	**Testing I2S input on the development board**	**PCB layout design and overall design review**	**Design Review - T PCB Review - F**
10/14	**Server development**	**Testing I2S output on the development board**	**Order PCB parts**	**PCB Order - T Teamwork evaluation - W Last Day for Machine Shop - F**
10/21	**Test communication with the server**	**Finish communication with AI web client**	**PCB testing and debug**	**N.A**
10/28	**Establish the WAV packing and file saving mechanism**	**Testing PCB voice input and output**	**PCB testing and debug**	**N.A**
11/4	**Microphone testing**	**Set up the code for text display**	**Audio basic functionality and PCB reorder**	**Individual Progress reports - W Design Doc Regrade - F**
11/11	**Implement the**	**Add advanced**	**Audio**	**N.A**

	display function	**features such as smart home control**	**Improvement and Text Display**	
11/18	**Prototype debug, review**	**Prototype debug, review**	**Prototype debug, review**	**Mock Demo**
11/25	**Fall break**	**Fall Break**	**Fall Break**	**Fall Break**
12/2	**Final Demo, debug, work on final presentation**	**Final Demo, debug, work on final presentation**	**Final Demo, debug, work on final presentation**	**Final Demo**
12/9	**Wrap up, work on final paper**	**Wrap up, work on final paper**	**Wrap up, work on final paper**	**Final Presentation**

4.Ethics and safety

#User Privacy and Data Security

The AMADEUS project involves the collection and processing of user audio data, which raises privacy and data security concerns. In accordance with the IEEE Code of Ethics, Section I.5, it is our responsibility to ensure that the privacy of the user is protected and that sensitive information is not misused. User data must be securely transmitted and stored, utilizing encryption both during transit (Wi-Fi) and at rest on cloud servers. Compliance with global data privacy regulations must be maintained. To avoid breaches, we will anonymize user data when possible and implement secure protocols for all communications between the ESP32 board and cloud server.

#Bias in AI Models

As the project involves the use of AI, it is important to address the risk of bias in the AI models. According to the ACM Code of Ethics, Section 1.4, we must ensure fairness in algorithmic processes. Any AI system should be free of bias regarding race, gender, or other demographic factors. To mitigate this, we will work with the developers of the AI model to ensure that training data is diverse and representative. Additionally, continuous monitoring and auditing of AI model performance will be established to prevent unfair treatment of users.

ce will be implemented, allowing users to review data policies and make informed decisions.

#Hardware Safety

The AMADEUS system involves the use of an ESP32 microcontroller and audio-related hardware components. According to UL 60950-1 and IEC 62368-1 safety standards for audio-visual and IT equipment, the PCB must be designed to avoid electrical hazards such as short circuits or overheating. Additionally, any exposed parts of the system must be properly insulated to protect users from accidental electrical shocks.

#Power System Safety

The power supply system uses both USB and battery-powered configurations. It is crucial to ensure that these power sources are properly regulated to avoid potential fire hazards or battery explosions. The IEEE Code of Ethics, Section I.1, requires us to prioritize public safety and welfare. Therefore, we will conduct rigorous testing of the power supply circuit, ensure compliance with FCC Part 15 regulations regarding electromagnetic interference, and adopt safety protocols for battery usage, such as overvoltage and temperature protection circuits.

5. Citation

[1] Marr, B. (2024, July 2). Generative AI is coming to your home appliances. *Forbes*. https://www.forbes.com/sites/bernardmarr/2024/03/29/generative-ai-is-coming-to-your-home-appliances/(visited on 10/1/2024)

[2] FalcoTK. (n.d.). GitHub - FalcoTK/character-ai: Unofficial API for character.ai, Support chat v2, support voice module (BETA TES). GitHub. https://github.com/FalcoTK/character-ai

[3] IEEE. “”IEEE Code of Ethics”.” (2016), [Online]. Available:IEEE - IEEE Code of Ethics(visited on 9/17/2024).