# Capita Selecta Project Description

Student: Ahmed Al-Ganad – 528820

## Contents

Project Overview: ........................................ 1
Project Objectives: ........................................ 1
Design Aspects: ........................................ 1
Hardware Components and interaction: ........................................ 1
Software Libraries and platforms: ........................................ 2
Design Decisions and Justifications ........................................ 2
Project Scheduling (Starting from this week): ........................................ 2
MoSCoW: ........................................ 3
Current progress: ........................................ 4

## Project Overview

The project is about designing and In-Vehicle Infotainment system using a microphone and a speaker with ESP32 microcontroller and implementing an AI voice recognition to allow a user to control various aspects of the infotainment systems. Additionally, the system will support OTA(Over the Air) Updates. Enabling AI module updates without physical intervention.

## Project Objectives

- **Implement I2S Protocol**: Understanding and implementing I2S protocol to enable microphone input and speaker output.
- **Voice Recognition and NLP**: Implement Voice Recognition and Natural Language Processing using AI to process voice commands on the ESP32
- **OTA Firmware Updates:** Enable OTA firmware updates to securely update the software.
- **RTOS Task Management**: Ensure the system can handle multiple tasks such as voice processing, internet connectivity, output control, and OTA updates simultaneously using Real-Time Operating System tasks management.

## Design Aspects

### Hardware Components and interaction:

- **ESP32-S3-WROOM Microcontroller**: Selected for its dual-core processor, built-in Wi-Fi and Bluetooth capabilities, and 8MB SRAM. The ESP32-S3 provides sufficient processing power and memory for handling multiple tasks, including voice recognition, OTA updates, and task scheduling via RTOS.

- **INMP441 I2S Microphone**: digital microphone uses the **I2S protocol** to interface with the ESP32. I2S is common for handling audio data, ensuring good-quality sound capture. The ESP32 processes this audio input for voice recognition.
- **LEDs**: To demonstrate an output that an audio is processed.
- **PAM8302A Amplifier and 8-Ohm Speaker(Maybe used)**: For audio feedback, the PAM8302A amplifier is connected to the ESP32 DAC output, which can produce sound feedback for command confirmations.

Software Libraries and platforms:

- **Machine learning Libraries:** TensorFlow Lite will be used to get or import a pre-trained module that recognize natural voice commands, because of it's light weight and the ability to run on microcontrollers.
- **Real-Time task management:** FreeRTOS will be used to mange tasks within the running environment due to it's simple functions and full compatibility with ESP32.
- **Audio signal processing:** Fast Fourier transfer will be used to process the signals and libraries such as ArduinoFFT will be used.
- **OTA Updates:** The ESP32 OTA library will be used to manage secure, wireless updates of the firmware.

---

## Design Decisions and Justifications

This section explains the justification of choosing ESP32-S3-WROOM, INMP441 microphone, and TensorFlow Lite as an AI framework.

- **Microcontroller Selection**: The ESP32-S3-WROOM was selected over other microcontrollers due to its dual-core processing capabilities, relatively high RAM 8MB, integrated Wi-Fi and Bluetooth, and compatibility with machine learning models on TensorFlow Lite. This module ensures processing captivity for handling multiple audio, network, and control tasks in real-time.

- **Microphone Selection (INMP441)**: The INMP441 microphone was chosen because it uses the I2S protocol, which is natively supported by the ESP32. This minimizes the need for additional processing and ensures high audio quality, which is essential for accurate voice recognition.

- **Machine Learning Framework**: **TensorFlow Lite** for Microcontrollers was selected because it's optimized for embedded systems, allowing run a compact neural network for voice recognition on the ESP32. Also, for its huge documentation, communities, and pre-trained modules.

## Project Scheduling

Sarting from this week 11th of November the following table will be the ideal plan

| | |
|---|---|
| **Week 1** | - Research deeper into TensorFlow lite and software libraries of choice.<br>- Connect the hardware to test connections |
| **Week 2** | Integrate INMP441 Microphone and Capture Audio Input<br>- Use I2S library to capture audio from Microphone<br>- Write a simple software to verify audio data capture and test different sampling rates. |

| | |
|---|---|
| | - Ensure the audio quality is sufficient for accurate voice recognition. |
| **Week 3** | Audio processing and feature extraction<br>- Use ArduinoFFT (Fast Fourier Transform) library to feature extraction on audio input. |
| **Week 4/5** | Load and integrate Pre-Trained Model<br>- Select a pre-trained TensorFlow Lite model for keyword spotting and/or voice command recognition<br>- Quantize the model to reduce memory usage, if needed<br>- Use TensorFlow Lite for Microcontrollers to load the model on the ESP32<br>- Test model integration by feeding pre-processed audio features into the model and observing output |
| **Week 6** | Command recognition and output implementation:<br>- Map each recognized command to specific actions, such as controlling LEDs or initiating audio feedback (if time allowed)<br>- Program the ESP32 to respond to recognized commands by executing outputs<br>- Test voice commands to ensure accurate recognition and appropriate responses or output |
| **Week 7** | Implement OTA updates & task scheduling with FreeRTOS<br>- Set up OTA (Over-the-Air) updates for the ESP32 using the Arduino OTA library<br>- - Use FreeRTOS to manage tasks: audio input, voice processing, command execution, audio feedback, and OTA updates. |
| **Week 8** | Final Test, Debugging, and Documentation:<br>- Test the whole system to ensure everything is working.<br>- Debug if needed.<br>- Finalize the final document. |

## MoSCoW

MoSCoW method helps to clarify and prioritize project requirements.

**Must Have**:

- **Voice Recognition and NLP**: The ability to recognize and process voice commands is the key feature of this In-vehicle Infotainment system. The voice recognition system must accurately convert speech to text and respond to basic commands related to IVI controls (e.g., music playback, volume adjustment, etc..).
- **LEDs showing an output**: This feature will demonstrate that the audio is processed correctly and there is an output accordingly.
- **OTA Firmware Updates:** Secure OTA updates for software is critical to ensuring that the system remains up-to-date without requiring physical access to the vehicle.
- **RTOS for Task Management**: Real-time task scheduling is essential to handle simultaneous processes like voice input, voice output, output feedback, and OTA updates.

**Should Have:**

- **Speaker for Audio Feedback**: This feature will enhance user experience by allowing the system to confirm the users command with audio output.

- **Natural Language Understanding (NLU):** Expanding the voice recognition system to include more advanced NLP features that can understand complex natural language commands and respond intelligently.
- **Error Handling and Recovery:** The system should handle errors, especially during OTA updates, to revert to the previous stable version if an update fails.

**Could Have:**

- **Output Indicators**: Basic LED indicators are needed to provide visual feedback on system status (e.g., voice command recognition, system errors, OTA update progress).
- **User Authentication via Voice**: Implementing a system that can recognize different users based on their voice and provide personalized responses or settings.
- **AI-Based Sentiment Analysis**: An advanced feature where the system detects the emotional state of the driver (e.g., frustration, calm) and adjusts the In-Vehicle Infotainment features accordingly (e.g., playing soothing music).

**Won't Have**:

- **Facial recognition**: This requires a camera and needs a high computational power.
- **Offline Natural Language Processing**: Running advanced NLP models entirely offline is too resource-intensive for the ESP32's capabilities in this project version.

## Current progress

I have searched into the project through many aspects such as components suitable, libraries needed, AI platforms, etc..

I also ordered all the components needed to start the implementation phase.