*Davis Rippon, Kody Appleton, John Curran, Dina Mohottalalage*
*Semester 1, 2024 - 05/05/2024 - Team E21*

# Executive Summary

**Introduction:**

This project aimed to create an innovative audio processing environment for Monash Embedded System Solutions (MESS). It primarily utilizes two components, a STM32 microcontroller for recording audio and a CLI interface for receiving data, alongside facilitating processing and viewing of the recorded data.

**Overview:**

The embedded hardware consists of a 3.5mm jack for monophonic audio input, a level adjust circuit that shifts the voltage input so it can be read by the microcontroller Analogue-to-Digital-Converter (ADC) and an Ultrasonic Sensor (US) that detects when the user is present near the recording device. The CLI includes a python script which allows user interaction and commands, a serial receiving mode where data sent over UART is received and recorded, and the ability to plot the waveform of the data processed as a .wav file. A separate C processing program scans the data, smoothing the raw ADC data using a moving average filter and creating .WAV files from both the raw and filtered values. US measurements are written to a .CSV file.

**Outcomes:**

During normal usage, our proposal functions by powering up the embedded system and running the CLI, where the embedded system already is providing ADC values over serial UART at just over 6kS/s, interspersed with US measurements. The CLI then begins recording these ADC values to a raw .DATA file when the US values are less than the user specified distance or manually begun by the user. Depending on how the user configured the settings, it continues recording until the user manually stops, or the user is outside the US set distance. The user can then run the C processing script, which reads the .DATA file and manually allocates system memory in the creation of arrays which store: 1. the US sensor data for writing to a .CSV file, 2. the ADC data for writing to a .WAV file, 3. that same ADC data after it is first passed through a moving average filter for writing to a separate “smoothed” .WAV file. The audio waveforms contained in the .WAV files are able to be graphically shown. The user can complete this process indefinitely, conduct many recordings and modify US distance and sampling rate.

**Challenges:**

Our embedded system operates as it does due to difficulties faced in implementing the communication of data between the CLI and STM32. Our STM32 continuously outputs data instead of starting/stopping when required as we faced issues implementing a check for reading the UART for a manual user stop message while also not blocking data transfer and ADC reading. To avoid blocking data transfer while reading from ADC, a hardware timer is used to generate a PWM signal which triggers the ADC in interrupt mode for continuous output. This single data point is immediately sent over UART using Direct Memory Access (DMA) to allow the CPU to move on to the next task. We experimented with continuous ADC recording using DMA and a circular buffer system taking advantage of half and complete callbacks but could not get this reliably functional without filling the buffer, skipping data, or blocking other functionality. The US value is also sent immediately over UART DMA with 10000 added to it, so it can be filtered out by the serial handler/processing program.

**Conclusion:**

In summary, our solution provides a fully functional system capable of receiving, recording, processing and analyzing audio/spatial data. With further development, recording could be implemented on the STM32 side using methods described above to further offload tasks from the CLI.