COMP3601 – Design Project A
Application Plan

Group 4

Member 1: Stanley Yeung (z5423210)
Member 2: Tsz Fung Chiang (z5449151)
Member 3: Zhihang Huang (z5237095)
Member 4: Ziyao Lu (z5340468)

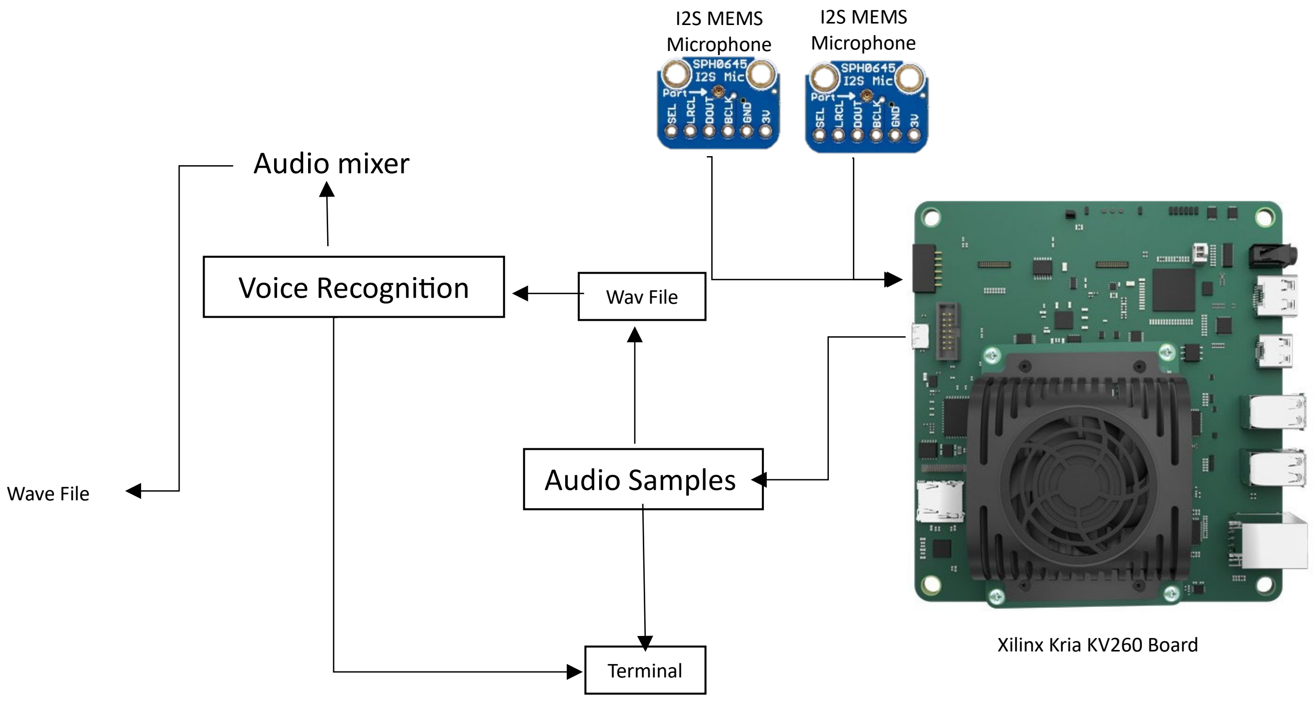
Overview of the project

In this project, our team will design a custom microphone system using the Xilinx Kria KV260 FPGA MPSoC. The microphone system has the ability to allow users to individually customise different aspects of the music captured from the I2S MEMS microphone, such as volume, frequency to suit their needs remotely using a recognised “wake up word” that would then trigger the AI assistant, Melody. These features are particularly useful for a large variety of users, whether they want to relax by listening to music or they want to enjoy music at a party without having the need to invite professional musicians to come and perform.

Current vision for application extension

Our team is considering the option to add another layer of complexity to the custom microphone system by implementing the sound mixing feature. In particular, this system will be integrated with our unique AI Assistant, Melody, allowing customers to remotely access the sound mixer with maximal comfort. These features are chosen in particular because of their extensive applications in real world, meaning that plenty of resources are available Internet (which is extremely useful given the limited time frame to deliver the project). More importantly, the AI Assistant also gives us the potential to explore how our Kria FPGA board can be a part of a larger eco system of products if time permits, making our product more unique and stand out from other similar competitors. At this stage, our team have decided that the system will first capture audio from the i2s microphone using the FPGA and save it as a wav file on another device. This will then become the library of songs that user can choose to listen to on their own device. Together with the AI assistant and sound mixer implemented using Python, user will then be able to control different aspects of the music and stop whenever they want by simply saying a particular command word.

System Diagram



Things in Brackets are inside the board

Architecture Diagram

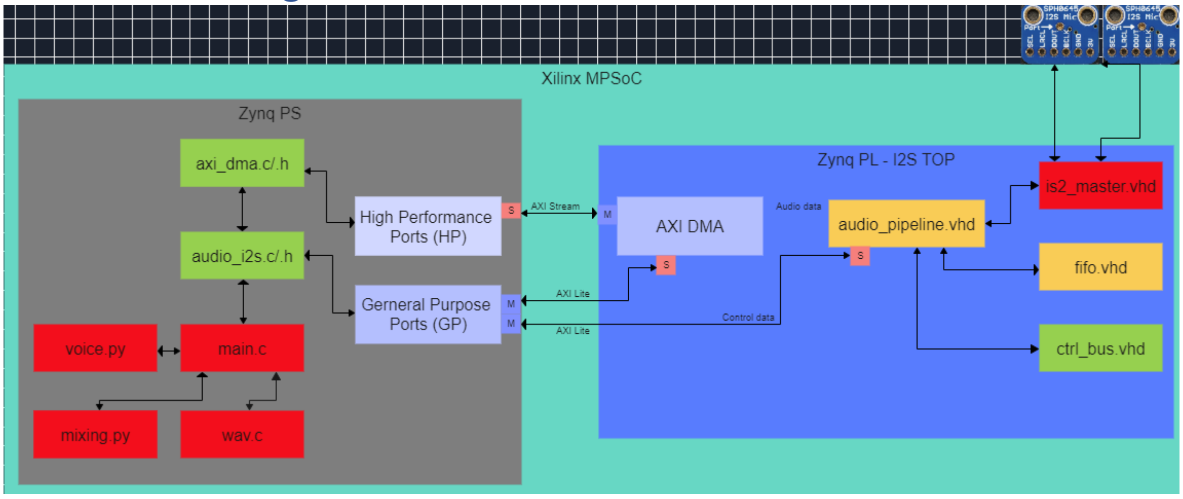


Figure 1 – Architecture Diagram of the microphone system

Explanation of the Project Plan Diagram

As shown from the system level diagram of the project, the KV260 FPGA board is connected to 2 i2s microphones, both of which will capture audio from an external source and transmit the audio to the FPGA board through the i2s protocol. As shown from the architecture diagram (Figure 1), the captured audio will then be passed to the FIFO for temporary storage via the audio pipeline of the board to ensure that data is only transmitted on the rising edge of the left/right select clock to comply with the i2s protocol. In addition, there is also the control bus which controls the mechanism of data transfer under the i2s protocol. The captured data from FIFO will then be transmitted to the Zynq PS device's general and high performance ports which allows us to easily write software program to perform actions with the capture data, such as saving it as a wav file to play on another device as well as adjusting the tone of the audio to extend the functionalities of the microphone system.

From a software perspective, the header files (which are labelled in green on the above architecture diagram) includes any functions that are used to memory map the physical addresses of different ports on the FPGA board to create virtual addresses which can easily be referenced to in the software files (which are labelled in red on the architecture diagram). The main program is then branched out to different subprograms that perform specific tasks, such as speech recognition, sound mixing and generating the wav file. This is done intentionally to make the code easier to maintain. Once completed, the program will firstly use the captured audio to generate a wav file and save it onto a local computer for later use. Then, if an appropriate "wake word" is detected from the user, the file that has been saved onto the local computer will then be played and the tone will be adjusted in real time to suit the users' preferences.

Plan to Complete Project

In order to ensure that our team is able to deliver a working prototype of the project with some application extension features shown in a limited time frame, the application extension milestone has been divided into the following tasks which is assigned to each of our individual team members shown below:

- Research and implementing the speech recognition feature (Stanley)
- Implementing the sound mixer feature (Asher and Calvin)
- Introducing a second channel audio input from the microphone (Ziyao)

Task 1: Speech recognition feature

This task will involve the use of a module named speechRecognition in the Python library together with the Google Text to Speech (gTTS) function to receive simple voice command input from users and perform certain actions. Once completed, users will then be able to hear what has been captured from the i2s microphone onto the FPGA in a wav file format without having the need to access the device in which the audio is saved onto, allowing us to achieve the buttonless user interface for the microphone system. Additionally, the voice assistant will also be able to adjust different aspects of the music (e.g. turn up the bass, turn off a particular channel) based on voice commands received from the user in real time using the sound mixer algorithm developed by our team. Furthermore, the speech recognition feature should also be optimised such that the impact of environmental noise on the ability to pick up the commands is minimised.

Task 2: Sound mixer

Based on the 2 channel audio received from the FPGA and saved in a wav file, a program should be written in Python to read the file and adjust different aspects of the audio in real time. This involves the implementation of an audio equalizer to adjust the tone and volume of the audio. More importantly, this algorithm will then be incorporated with the speech recognition program, such that whenever a command word is heard, the program is able to adjust a particular aspect of the streaming audio with minimal delay. To do this, research is required to be done on how to implement audio equalizers using digital filters that decompose the captured audio into different frequency ranges and real time audio processing in Python. Once completed, testing must also be conducted to ensure that all requirements listed above are satisfied before combining with the speech recognition algorithm for testing. Given that this may involve more work, two team members are allocated for this task to ensure its completion within the given time frame.

Task 3: Additional microphone input for audio

This task involves modifying the VHDL project code as well as the C files that memory maps the device for processing and generating the wav file to include a second channel input from another external i2s microphone. Once completed, the FPGA board will be able to capture audio input from both the left and right channel and save them onto a wav file on another device, which can then be streamed and processed using our audio processing and speech recognition algorithms. It is important to ensure that the data from both channels are transmitted properly without overlapping and the noise in captured audio is minimised. The aim of doing so is to more adjustable functionalities and complexities to the sound mixer (as supposed to the current one channel version) which gives user more flexibility to adjust the audio to suit their preferences.

Justification of Plan

Introducing MelodyMate – your gateway to unparalleled personalised audio experience. As the world's first smart audio mixer system featuring cutting edge technologies, including smart DJ Melody, who knows more about your musical taste and preferences than anyone else on this planet, you can easily enjoy music customed to your preferences from the comfort of your home with just a simple voice command. There is no need to worry about the complex structure of traditional sound mixers anymore! Together with the expansion of the MelodyMate ecosystem in the future, a world class theatre is just minutes away from the comfort of your bedroom.

There is no doubt that funding should be given to develop MelodyMate. According to a recent research from the World Health Organisation, by 2030, 1 in 6 people globally will be aged 60 and over. This means that there will be more people requiring mobility support, limiting their ability to complete daily tasks independently. The buttonless, voice over control feature of MelodyMate ensures that high quality audio experience is accessible independently for the aging population, giving them a form of entertainment, which helps reduce a sense of boredom. In addition to this, investing in the smart technology industry is expected to grow rapidly over the coming years despite the slow recovery economy post Covid-19, making it a safe investment option. According to a study from Fortune Business Insights, the global smart home market size is valued at USD 94 billion in 2023 and by 2030, the size would increase by 20.1%, to more than 328 billion. This, together with the change in lifestyle of the population due to the pandemic, with more people intending to relax at home during weekends, makes our product more appealing compared with other competitors. Thus, investing in MelodyMate now would guarantee huge profits.

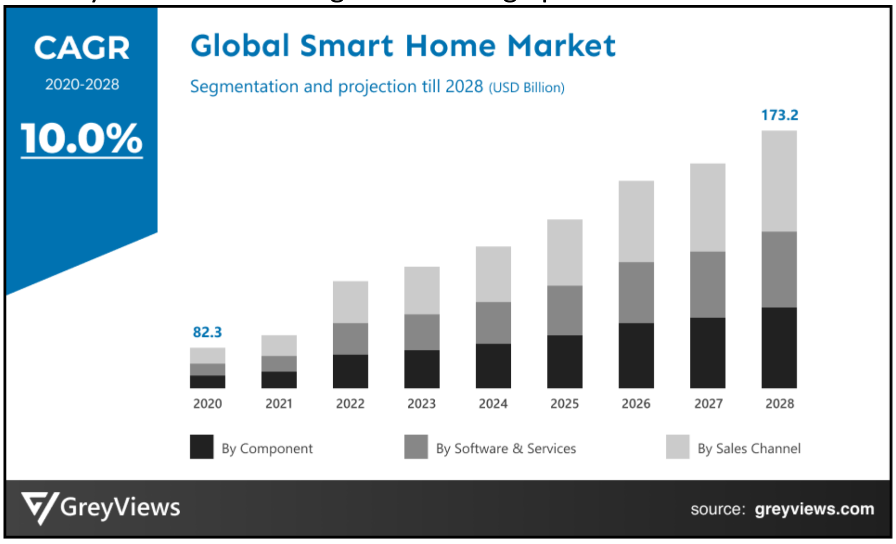


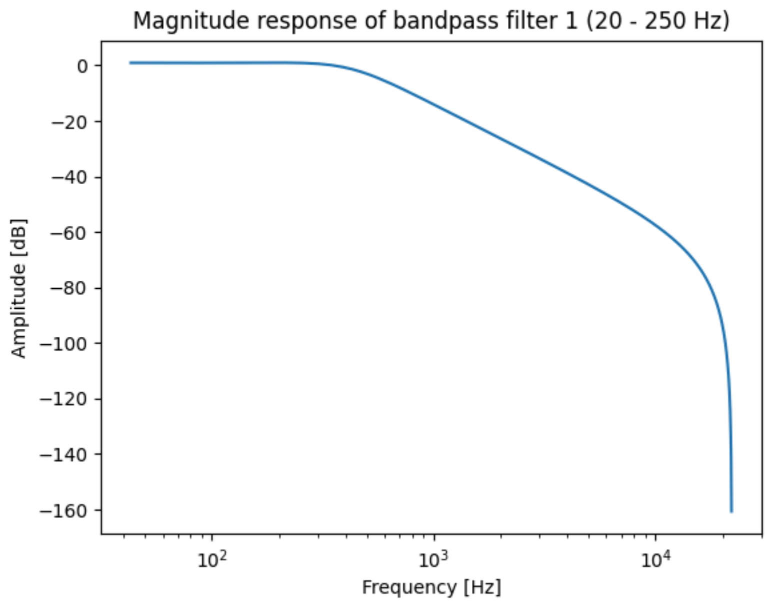
Figure 2 – Growth of the smart home market sector

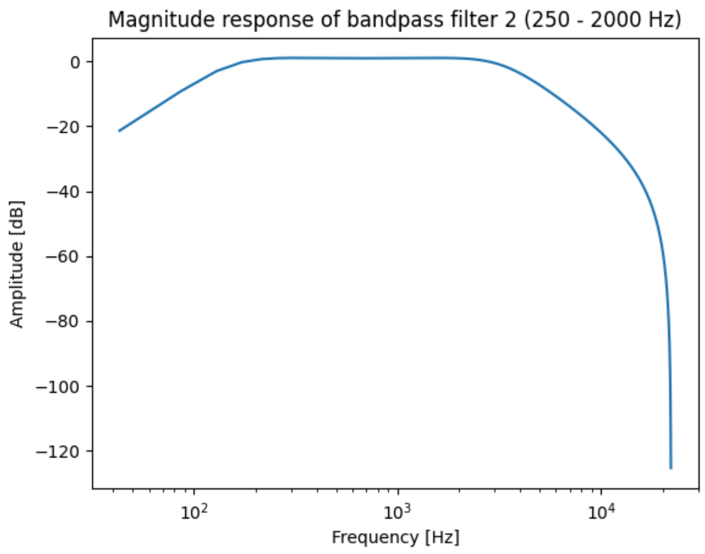
What our group actually achieved

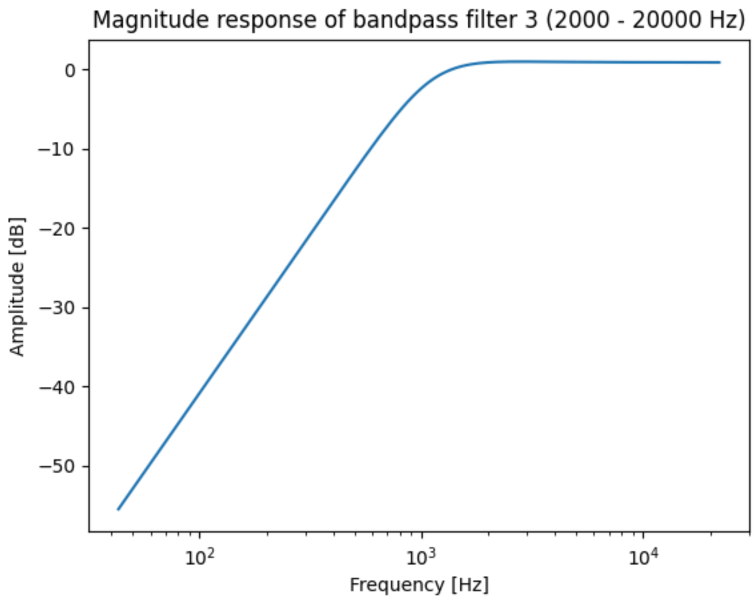
Feature	Planned implementation	Actual implementation
Speech recognition	A system that is able to listen to constantly listen to speech input from user (which is saved as a wav file) and respond correctly to input detected	• A wav file is generated every 2 seconds to capture speech input from user • A simple speech recognition algorithm that looks for key words/phrases in the captured audio in order to respond appropriately
Audio equaliser	Produce a 2 channel wav file music. Then using speech recognition feature adjust different aspects of the music playing in real time	• A two channel audio is successfully generated • Filters for each of the 3 channels (bass, mid-range and treble) implemented

As shown from the table above showcasing a summary of what was planned as part of the application extension for this project and what was actually implemented, it is clear that due to the extremely short timeframe available for implementation, only a portion of the planned feature was actually implemented. This includes the ability of the system to detect speech input from the i2s microphone of the FPGA board and a complete speech recognition algorithm implemented using Python to perform certain actions (e.g. determining current time and date). The algorithm has also been optimised by looking for certain phrases/words in the sentence captured, making it able to respond correctly to a wider range of inputs from users rather than forcing them to provide input in a specific form, making the system more user friendly.

In addition to the speech recognition feature, our group also made an attempt at implementing the multiple channel audio equaliser feature. At this stage, the microphone system consists of 2 i2s microphones which allows it to capture input audio and save the audio as a 2 channel wav file. While some progress has been made from the software perspective in setting up the three channel audio equaliser (as shown from the magnitude response of the 3 bandpass filters below), the complexity remains on our limited knowledge of audio processing and in particular how to deal with multiple channel audios and implementing digital filters in Python given our limited knowledge in digital signal processing. As time was very limited for implementation, our group only got to implement all the features outlined above and was unable to stream the audio using the audio equaliser that we have designed in real time.







Frequency response of the digital filters

How the above features would have fit into the big picture

If we were given more time (perhaps around 2 months), we would be able to complete the audio equaliser algorithm which was also implemented in Python which allows us to stream the captured audio from the FPGA in real time and adjust specific features of the music while it is streaming. This would then allow us to incorporate the audio equaliser with the speech recognition algorithm which allows user to adjust the tone of the music to suit their needs using voice over control by exploring the different key words/phrases for several specific actions. By completing these tasks, we would have achieved our goal of implementing an buttonless audio equaliser with voice over control.