# COMP3601 Project Plan
# Team8

Aron (z5371654), Sagar (z5364817), Priyansh (z5501423),
Nishant (z5270429)

## *Project Overview*

We aim to create a custom microphone system using the Xilinx Kria KV260 FPGA-based Multiprocessor System-on-Chip (MPSoC). The Kria KV260 offers a unique combination of FPGA fabric and a quad-core ARM processor, allowing us to harness the benefits of both custom hardware and traditional processing. Our primary goal is to establish a functional system that can capture audio data from I2S MEMS microphones and output it either via the Terminal or save it as a .wav audio file. The Kria KV260 Vision AI Starter Kit is our development platform, featuring an ARM Cortex A53 processor and FPGA. We will run a customized Linux image on the ARM processor, enabling us to develop device drivers and applications for audio processing. The I2S MEMS microphones provided are digital audio devices capable of capturing a broad range of sound frequencies. They utilize the I2S serial bus interface protocol for transmitting audio data to the FPGA-based audio processing pipeline. Our team will leverage existing and newly written code to realize the Basic Requirements and then plan to work on the Application Extension after M2.

## *Current Progress Towards Basic Requirements*

Ensure that the Xilinx Kria KV260 boards and I2S MEMs microphones are set up correctly. This includes connecting the microphone to the FPGA board using the I2S interface and providing power to both devices. Configure the FPGA fabric to communicate with the I2S microphone. Implement the necessary I2S protocol for audio data transmission. Work on developing software to handle the audio data. This involves buffering, processing, and storing the audio samples. Create code to output audio data and save it as .wav files. Create a testbench for the implementation of the FSM and debug any issues that may arise during testing.

Our team has made significant strides toward achieving the Basic Requirements of our custom microphone system project. We successfully connected the I2S MEMS microphone to the Xilinx Kria KV260 board, enabling the collection of audio data. This involved configuring the I2S interface on the FPGA to interface with the microphone,

ensuring a seamless data transfer process. On the software side, we've made substantial progress with the PetaLinux system. We have started developing device drivers to communicate with the microphone and manage audio data. These drivers are essential for configuring the microphone and ensuring that audio data is correctly sampled and transmitted to the FPGA fabric. Some challenges we faced were Vitis not being available on the non-intel Mac. Even when we had a logically correct i2s_master, even after debugging for a while, the output was incorrect which hindered our testbench building ability.

Our current progress demonstrates that we have successfully tackled the fundamental hardware and software integration necessary to capture audio data from the I2S MEMS microphone. Our next steps will focus on refining and optimizing these processes to ensure the seamless collection and output of high-quality audio data.

## *Plan To Complete Basic Requirements*

In this project we are working on a microphone, and implementing it on the FPGA board. We are using the Kria KV260 board for implementing this. We collected the audio samples inside the microphone and then created something meaningful out of it.

Implementing the I2S protocol for audio data transmission falls under this category. The FPGA setup will be optimized to handle the audio data effectively, ensuring that it is sampled accurately and transmitted to the ARM CPU.Aron is working on the finite state machine and implementation of the code and the FSM. Sagar is also working on the code implementation and debugging as they are the two with the board and have the software installed.

Priyansh and Nishant are working on the report and helped in following the quick start follow up guide. Furthermore, we have established a foundation for data handling and output. We are working on code to enable real-time audio data output and to save audio samples as .wav files for later analysis.

The I2S MEMs microphone and the Xilinx Kria KV260 board will be used to build a dependable custom microphone system that can record audio data, making it suitable for future additions and applications. This fundamental accomplishment will enable us to investigate more sophisticated functions and distinctive use cases. We also held regular meetings to discuss progress, challenges, and solutions to ensure that all team members are aware of each other's work and can provide support when needed.

## *Current Vision For Application Extension*

Our group plans for further application extension is ***'wake word'*** as seen in Google Assistant and Siri. We will try to implement it by using AI models. When the word is said, the wake word audio is sent in cloud from speech to text. We accomplish this by putting in place a pipeline for real-time audio data processing. The pipeline should have stages for wake word detection and audio feature extraction.

***Files needed:***

*PreparingData.py* where we collect the data of saying the wake word around 100 times and also the sound not containing the wake word.
*PreProcessing.py* where we process our audio files.
*Training.py* where we train our model using TensorFlow.
*Prediction.py* we evaluate our trained model.

*(Note - We can either code in C, C++ or Python depending on the most preferred coding language for all group members)*

| Audio | Label |
|---|---|
| Contains wake word | 1 |
| Contains wake word | 1 |
| ... | |
| ... | |
| Does not contain wake word | 0 |
| Does not contain wake word | 0 |

*(label '1' indicates the audio file containing a wake word and label '0' indicates the audio file not containing a wake word)*

We are also planning to implement ***'soft authentication'***. This is where the identity of the user is verified using non invasive and non biometric means. We plan to use a system that can use voice recognition technology to verify a user's identity based on their unique vocal characteristics. We plan to collect voice samples from authorized users and create a secure voice database and develop or integrate voice recognition software for analyzing and comparing voice samples. Users record their voice samples,

which are linked to their identity in the voice database by speaking a predefined passphrase. The system then analyzes the incoming voice sample and compares it to existing samples to grant access if there's a match. We will have to regularly update and retrain the voice recognition system to enhance accuracy.