Australian National University

Micro-controller Project:

Audio system for a gaming device using a Micro-controller board

ENGN4213/6213
Digital Systems and Microprocessors

Semester 1, 2023

The Australian National University

ANU College of Engineering, Computing & Cybernetics

Table of Contents

1. BACKGROUND

Systems that employ embedded microcontrollers are all around us: e.g., smart phones, calculators, kitchen appliances, TVs, cars, drones, Active Noise Control (ANC) headphones, and smart speakers. We believe that ENGN4213/6213 will prepare you to become a skillful engineer who will love to invent, design and build innovative systems to solve difficult problems. To provide you with some experience in building an embedded system, **in this project you will select a sensor (either a microphone or an Inertial Measurement Unit (IMU) sensor) to *(I)* obtain some real-world signals as inputs, *(II)* process that signal stream within the micro-controller, and then *(III)* output the processed signal data to external devices.** Many systems have these three elements.

Though you will use the STM32F411RE micro-controller board in this project and one of the two sensors (microphone or IMU sensor) with the toolkit provided by STM (STM32CubeIDE), our real goal is to prepare and develop your ability to build what you can imagine. Thus, the skills you gain are much more general than the details of a single processor or programming language. Chips may have a limited life span but the skills you gain will have a much longer life span.

2. PROJECT STATEMENT

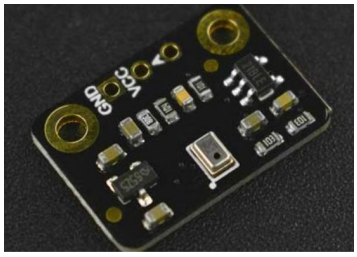
You are required to enhance the player experience of a VR MOBA (Multiplayer Online Battle Arena) game using your STM32F411RE micro-controller board. You need to allow players to have an immersive experience while ensuring their privacy. Currently this game has two features that can be designed by you - the **voice chat platform** and the **spatial audio system**. The voice chat platform allows players in a team to communicate with each other in real-time during the game to better collaborate to defeat enemies. To protect the privacy and make it fun, the player's voice can be changed to a completely different *timbre*[1]. Spatial audio systems can provide players with more accurate sounds with directional information. For example, players can determine the orientation of the enemy by the binaural sound of gunshots, and when the player turns around, the perceived orientation changes accordingly.

In this project, you will choose to develop one feature for a VR headset using your micro-controller board with only a single input sensor. Each project group should choose one of the sensors from either (I) microphone or (II) IMU. You are required to implement a set of real-time signal processing/signal analysis onboard and store/playback/display the processed data.

In your design, you must use Interrupts, Timers and suitable communication protocols (UART, I2C, SPI etc.,) to demonstrate your ability to develop on a microcontroller. You must justify what design choices you make.

[1] The character or quality of a musical sound or voice as distinct from its pitch and intensity. Timbre is what makes a particular musical instrument or human voice have a different sound from another, even when they play or sing the same note.

Sensor Option 1: Microphone



Input: You need to connect an inexpensive microphone (as in the part list) to the micro-controller board so that you can obtain a continuous stream of digital signals (audio signals) to be processed within the micro-controller. You may need to sample the analogue signal from the microphone to generate your digital signal stream (Analogue-to-Digital Converter, ADC).

Onboard Signal Processing / Signal Analysis:

To record the player's voice and provide better privacy in the voice chat platform, the system can be operated in two modes: 'normal mode' (default) and 'high privacy mode'. You should process the incoming audio signal stream onboard the microcontroller by implementing the following functions:

- **(a)** Filter out low-frequency noise (such as wind noise) by applying a high pass filter with a suitable cut-off frequency (say 200 Hz).
- **(b)** When the user enables the 'high privacy mode' through the blue push-button, the recorded speech signal should be real-time modified, so that the identity of the person speaking is obscured, but the speech can still be understood.
    - b.1. Add 'Robot Effect' to the original speech signal by applying a delay and adding up all delayed signals. After this step, the audio signal will sound like a robot. (More information is given in Appendix 3: Implement robot effect.)
    - b.2 Applying 'High pass filter' to the signal after b.1 to remove the low frequencies. This makes it harder to find out who is speaking.
- **(c)** Under 'high privacy mode', analyze the frequency spectrum of the original speech signal and modified signal. (Perform a Fourier Transform on the audio signal to identify dominant frequency contents and the change in frequency content over time. This can be done by taking a block of samples and implementing a Discrete Fourier Transform equation. Thus, you will calculate the Fourier Transform on blocks of audio samples. You can choose an appropriate block size.)
- **(d)** The system can be reset every time the user presses the black push button. During the reset, the system restarts a new recording process under 'normal mode'.

Output / Display:

You must make sure that the processed data is fully stored and clearly displayed to the user/marker. The output of **(a)** is updated continuously while **(b)** and **(c)** are updated periodically.

(i) The real-time microphone recordings in **(a)** will be displayed on the computer. One method is to transmit the processed data from the microcontroller to your computer using UART, and then use a UART terminal to see the data, or use a programming language (MATLAB, Python, etc.) to receive the UART transmission and plot the processed data on your computer/laptop screen.

(ii) The original and modified microphone recordings in **(b)** are stored on the SD card. During the reset, the system will generate new files to store the data in the new recording process. The audio streams stored on the SD card can be played back through a computer. (More information is given in Appendix 4: Store and re-play.)

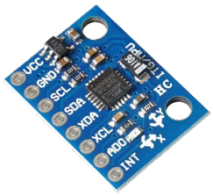
(iii) The dominant frequency contents of the captured sound and the modified sound in **(c)** will be displayed on the LCD screen.

Advanced Activities (For Higher Grades):

(i) Only display the output if there is sound activity detected. This can be implemented by monitoring the amplitude of the audio signal, and only displaying the data when the audio amplitude exceeds a specified threshold.

Sensor Option 2: IMU sensor

Input: You need to connect the provided IMU (MPU-6050) to the micro-controller board so that you can obtain a stream of directional data (azimuth angles) according to its orientation and movements. The outputs from the gyroscope and accelerometer are captured and real-time processed on the micro-controller board.



Onboard Signal Processing / Signal Analysis:
The final product should have the following functions:

(a) The signals (azimuth angles in degrees) calculated from the outputs of the gyroscope and accelerometer are real-time streamed to the user.

(b) When the IMU is rotated to face a specific direction, the direction can be set when the user presses the black push button.

(c) When the direction is set, the board should be able to generate the binaural signal of the given piece of music, so that it sounds like the music is emitted from roughly the same direction. (This can be done by taking a block of audio samples from the music file and implementing a filtering process. Thus, you will obtain the filtered binaural signals on blocks of audio samples. You can choose an appropriate block size.) *[Further details on the calculation of binaural signals and a dataset of the filter will be provided in a supporting document].*

(d) The system can be reset every time when the user presses the blue push button.

Output / Display:
You must make sure that the processed data is fully stored and clearly displayed to the user/marker.

(i) The real-time azimuth angles in **(a)** will be displayed on the computer. One method is to transmit the azimuth angles from the microcontroller to your computer using UART, and then use a UART terminal to see the data, or use a programming language (MATLAB, Python, etc.) to receive the UART transmission and plot the processed data on your computer/laptop screen.

(ii) The set direction in **(b)** will be displayed on the LCD screen and can be updated each time when the user presses the black push button.

(iii) The reproduced binaural signals in **(c)** are stored on the SD card. The audio streams stored on the SD card can be played back through a pair of headphones plugged into a computer. At the start and reset, the system will generate new files to store the newly reproduced binaural signals.

During the demonstration, in addition to the perceptual listening, you are also encouraged to demonstrate the reproduced output in **(c)** in other forms, such as plotting using MATLAB.

Advanced Activities (for Higher Grades):

(i) To improve the accuracy of the azimuth angles, the IMU is calibrated every time when the system starts and is recalibrated during the reset.

3. BORROWING SENSORS

Each group located in Canberra can borrow a sensor, a SD card reader, and a SD card from us. First choose your group partner and decide which sensor option you will implement. Collection times will be announced on Wattle. We assume that all remote groups have already purchased the relevant parts for the project.

4. RECOMMENDED WORKFLOW

The following recommendations *are not prescriptive,* but they might help you find your way through to the end of the project:

1. **Find a project partner and register** your group on Wattle.
2. **Get to work!** Try to implement a bare minimum system where you can obtain a signal stream to your microcontroller and transmit the output to a computer display using UART communication.
3. Once you have a minimum system working, you can add the onboard processing.
4. You can use your lecture notes and Labs where you learnt how to use GPIO, UART, Interrupts, I^2C, SPI, ADC. You can search the web for helpful hints.
5. There are very good videos available online (some were recommended through lectures and in Wattle). You need to provide references to others' work and provide citation detail as necessary. If you use code from the web, clearly give the source and cite the work.

These are just suggestions. What we want is for you to learn and be proficient in embedded system development.

5. DELIVERABLES AND ASSESSMENT

5.1 Assessment process

For this project **students will work in groups of 2**. The work will be assessed as group work, with the same mark awarded to both members unless one of the group members has not

contributed to the project in a fair manner. We will allow single student projects under special circumstances, but you need to obtain prior permission from the course convenor.

The assessment of the work will be done by demonstrating your system and using the documentation (Report). **Demonstrations will be conducted (i) 2pm – 5:30pm on Wednesday 24 May, (ii) 9am – 1pm Thursday 25 May, (iii) 2pm – 5:30pm on Thursday 25 May, (iv) 9am – 12:30pm on Friday 26 May, and (v) 2pm – 5:30pm on Friday 26 May, locating in Ian Ross R103.** We will allocate 15 - 20 minutes for each demonstration.

If you agree, we would like to record some demonstrations to be used as examples for the benefit of the future version of this course. If you prefer to opt-in for recording, please e-mail us. This is completely optional and voluntary.

5.2. Written project report/document

Your project report should give a *clear and concise but complete* presentation of your work. You should imagine that your document is to serve as a comprehensive technical document to be read by a technically knowledgeable user who knows nothing about your design.

Do not tell a story about how you worked through your assignment rather holistically present your work. It is not expected for the final document to exceed 8-15 pages. (Please note: there is no hard page limit). It can be useful to include a listing of your code as an appendix to your report in addition to uploading your code with your submission.

If you prefer to share your project report to be used as examples for future years, please include the following sentence in the Appendix of your report: ***"The group approves this report for inclusion on the course website (Wattle)."***

5.3. Code

Your code will not be marked for programming style, but it *will* be looked at, especially if the report is not sufficiently clear or the project does not work, hence forcing the markers to dig into the program to see what partial marks can be awarded. Please use comments for your own and the markers' benefit.

You are required to submit a functioning version of your code corresponding to your final implementation as an attachment to your submission. Please ensure that you submit the correct, working version of your code.

5.4. Due dates

The project report submissions are due **on Thursday 1 June 2023. There is no cut-off date**.

This project is a major learning and assessment (25%) component of this hands-on hardware and software focused course. Further, the students will have sufficient time to write a comprehensive report.

We will be very reluctant in granting extensions because you will eat into your exam preparation time for other courses. However, we understand that there may be exceptional circumstances and if that happens, please contact us.

5.5. Submission process

The project report submissions will be done through Wattle.

It is strongly recommended that **you re-download your own submission after uploading it to wattle** in order to check for completeness and version accuracy before you finally send it for marking. A failure to upload the correct files will not be a ground for appeal of a poor mark. Also, remember to press 'submit' otherwise your submission is in the system as a 'draft'. We have chosen this option to allow the project partner to double-check the files before submission.

5.6. Marking Criteria

You will be graded on several aspects of the project: Level of hardware/software complexity; Appropriate use of external hardware, C Programming; Does the project work according to specification (which you will write); Demonstration of the system.

For each component of the embedded project, the following make up the partial mark:

ASSESSMENT CRITERION		WEIGHT
Clarity of written documentation	Have the students provided documents that clearly show their design approach, the design structure and the features of their system? Does the document allow the markers to gain a clear, comprehensive understanding of the design without having to refer to the fine details of the C code? Does the documentation resemble a 'professional Engineering report' as you would find in the industry?	30%
Quality of design	Does the system use 'Interrupts', GPIOs, I^2C, SPI and UART? Have the processed data been fully stored in the SD card? Does the report provide proper justification for the design including the choice of the filters, thresholds, and other choices? Does the group implement all signal analysis/processing tasks under the chosen option?	40%
Correct operation	Does the system operate as intended? (Assessed through the demonstrations)	30%

Late delivery: We will allow late submission only under special circumstances, but you need to obtain prior permission from the course convenor.

Referencing: You should cite any code and publications used from external sources.

Plagiarism will not be tolerated. Make sure you are the author of your own work as ANU has very strict policies against academic misconduct. *You must acknowledge **any** external sources* (if any) you may have referred to in creating your design and specify the extent of the impact of said external resources on your work. This includes providing in-code referencing and citations in the report. Please be aware that our prevention strategies include a check of your work against a broad list of internet resources.

Appendix 1: The recommended workflow for the Microcontroller project.

Step 1: Find and check the datasheet of the sensor, stm32f411re, and Nucleo board pinout (power voltage, pin connection, etc). Then connect the sensor to the stm32 board following pin connection and power requirement of the sensor and stm32f411re.

Step 2: Build a simple system, which contains only input and output, and test whether the sensor is working well through some initial readings.
One possible method: Set up the sensor, use the polling method to read the sensor output (internal buffer in the board), and transmit the data to the PC through UART. For PC serial terminals, it is better to choose a terminal that can real-time display the data. You can also use MATLAB or another PC application to plot your readings from the file (offline, optional).
Once you confirm the readings from the plots, you are ready to start the next step.

Step 3: Build a real-time system, that contains only 'continuously sample the input', 'save the data to buffer/buffers' and 'continuously display the output' without on-board processing.
Make sure that (1) data transmission from the sensor to on-board memory, 3) data transmission from on-board memory to PC (UART) are all working well. Hint: we need to double-check that the sampling rate, buffer size and UART transmission rate are compatible.
Once you confirm that you can read data from the sensor under desired sampling rate and transmit data to the PC simultaneously without losing data, you are ready to start the next step.

Step 4: Add the on-board signal processing to the system, continuously display the processing results on the PC and store the data on the SD card (10%). Note that for the filtering process and Fast Fourier Transform process, there are libraries available online (e.g., CMSIS DSP library).

For Microphone:

a) Filter out low-frequency noise (10%)
b) Add 'Robot Effect' and high pass filter (20%)
c) Fast Fourier Transform (10%)

For IMU sensor:

a) Calculate the azimuth angles and real-time display (10%)
b) When the IMU is rotated to face a specific direction, the direction can be set when the user presses the black push button and is displayed on the LCD screen (10%)
c) Read the given piece of music from the SD card and generate the binaural signal (20%)

Step 5: Attempt the Advanced Activities

For Microphone:

a) Only display the output if there is sound activity detected (10%)

For IMU sensor:

a) IMU calibration (10%)

Marking criteria:

a) If you complete step 1,2,3, you will get ~40%
b) If you complete step 1,2,3,4, you will get ~90%
c) If you complete step 1,2,3,4,5, you will get 100%

Notice that the completeness/quality of each step will be reflected through both 'report' and 'demo'.

Appendix 2: Important concepts, libraries, and useful links.

- **Real-time sampling, processing, and saving to the SD card:** It is recommended to adjust the sampling rate, buffer size, etc., to make sure the speed for ADC reading, processing, and SD card writing can be compatible with each other. One way to check is by using a timer to check the runtime for each function block and making sure all the function blocks can be finished before the next ADC buffer is ready for processing.

- **SD Card Management:** The SD card only operates during the privacy mode. The coding for SD card operations is greatly simplified using libraries, specifically, the **FatFs** and the **fatfs_sd** libraries.

  - **FatFs:** FatFs is a library included with STM32CubeIDE, obtained by checking a box within the "Pinout and Configuration" interface. It provides the high-level functions for operation within a File Allocation Table (FAT) file system. The functions used within this library were:
    1. **f_mount() -** This begins communication (governed by the low-level fatfs_sd library) and initializes the file system on the SD card. Additionally, some memory on the development board is allocated to a "work area" which enables the use of subsequent functions.
    2. **f_stat() -** This checks the existence of a certain file within the file system. This is used inside a fileNameGen() function to create a file name which does not overwrite those already present on the SD card. Such a function uses a simple for() loop, adding a number to the end of a desired file name. For each name, f stat() is called. If the file already exists, the file name number is incremented and tested again, until an unused filename is found.

3. **f_open() -** This creates and/or opens a file in the file system. Its "file name" parameter is generated with *fileNameGen()* such that new files are generated with a unique name upon each activation of privacy mode.
4. **f_write() -** This writes byte of data to the opened file.

- **fatfs_sd:** FatFs is a high-level library and does not include any code to handle the actual low-level communication with the SD card. The fatfs sd library (provided with the assignment on Wattle) provides the needed low-level code to communicate with SD card. It uses the hardware abstraction layer (HAL) SPI drivers to communicate with the SD card. Corresponding to each FatFs function, it sends the required command bytes (corresponding to CMD0, CMD1, etc.) which perform the desired function over SPI.

- **To design the filter coefficients for low-pass filter or high-pass filter:**
  - Link 1: http://t-filter.engineerjs.com/
  - Link 2: https://www.arc.id.au/FilterDesign.html

- **Overlap-save and overlap-add in the filtering process:**
  Link: https://blog.robertelder.org/overlap-add-overlap-save/

- **To install and use the CMSIS DSP library:**
  Link: https://www.youtube.com/watch?v=vCcALaGNlyw

Appendix 3: Implement robot effect.

- **Robot effect:** Robot effect is an audio processing technology, which has been used in several areas, such as games, music, and movies. This technology aims to modify sound signals (e.g., speech signals), making them similar to robots' voices.

- **Method:** A method to implement robot effect is to create several delayed copies of the original speech.
  - Firstly, we can record a speech signal as the original speech signal.
  - Secondly, we can create some copies of the original speech, and add different delays to different copies. Therefore, we can obtain several delayed copies.
  - Finally, we can summarize these delayed copies. The result will be similar to a robot's voice.

Appendix 4: Store and re-play.

- **Store format:** The audio streams can be stored as either audio files (e.g., *.wav) or text files (e.g., *.txt).

- **Re-play method:**
  - If the audio streams are stored as audio files, these files can be played through a computer directly in general.
  - If the audio streams are stored as text files, the stored audio streams can be played with the help of related code (e.g., MATLAB coding) through a computer.
    - A demo about reading a text file and play the stored audio stream via MATLAB: Link: https://www.mathworks.com/matlabcentral/answers/482805-read-text-file-and-extract-audio-value