Design of a coherent sensor signal sampling methodology to implement a data logger to send audio sensor data from STM32 to NanoEdge AI Studio on PC

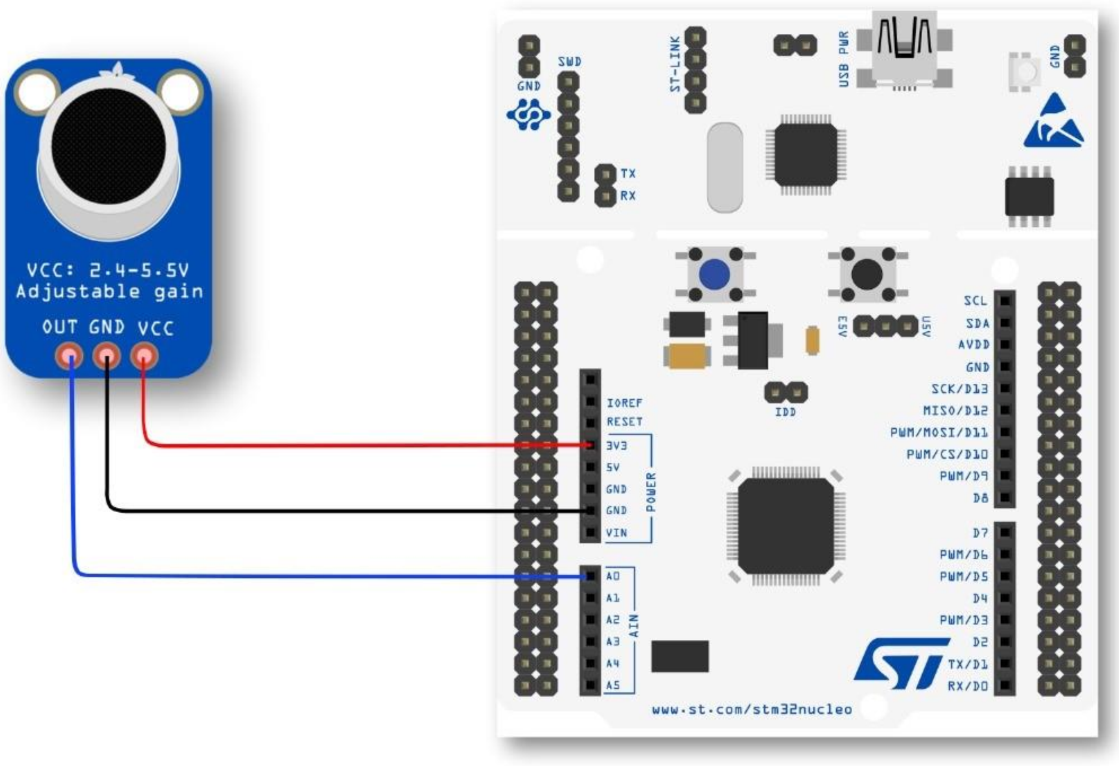
Objective:

The objective of this experiment is to interface an audio sensor to an STM32 microcontroller and create a datalogger code. This datalogger code will create a buffer where all the audio data sample will be stored, using which we will be able to create datasets of audio samples to build a machine learning model in the NanoEdge AI Studio.

Requirements:

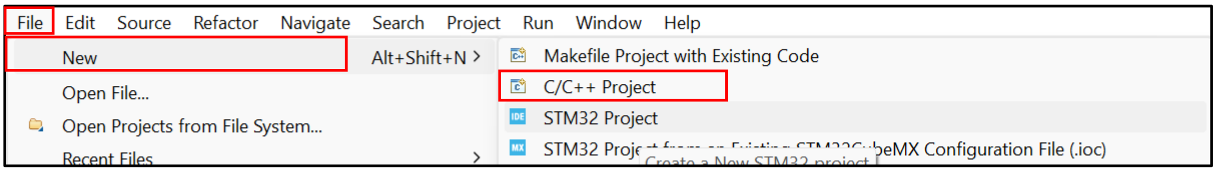
1. STM32 Cube IDE software.
2. Audio Sensor (Analog).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.
6. PC or Laptop.

Connection Diagram:

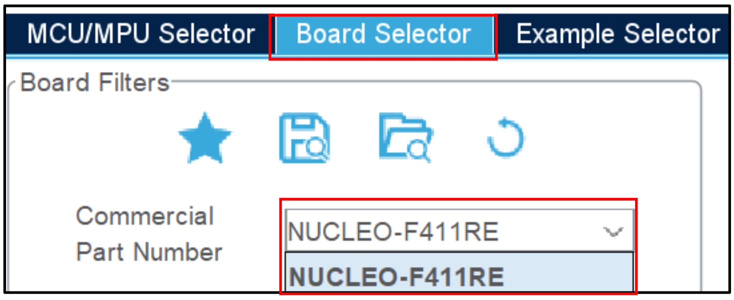


Procedure:

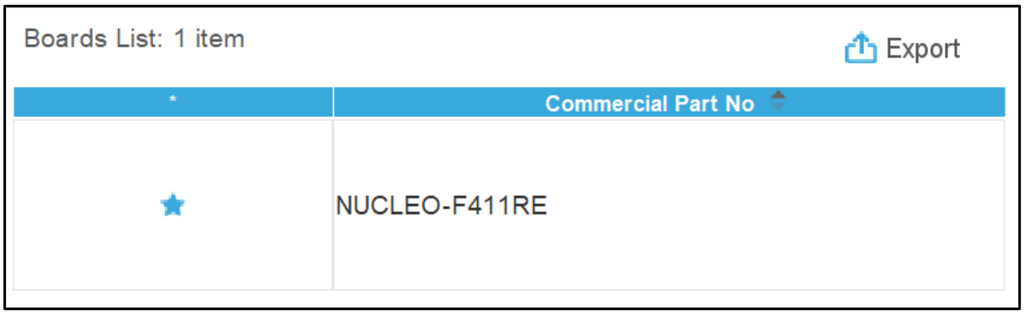
1. Click on **File→New→STM32 Project**to start your project on Cube IDE.



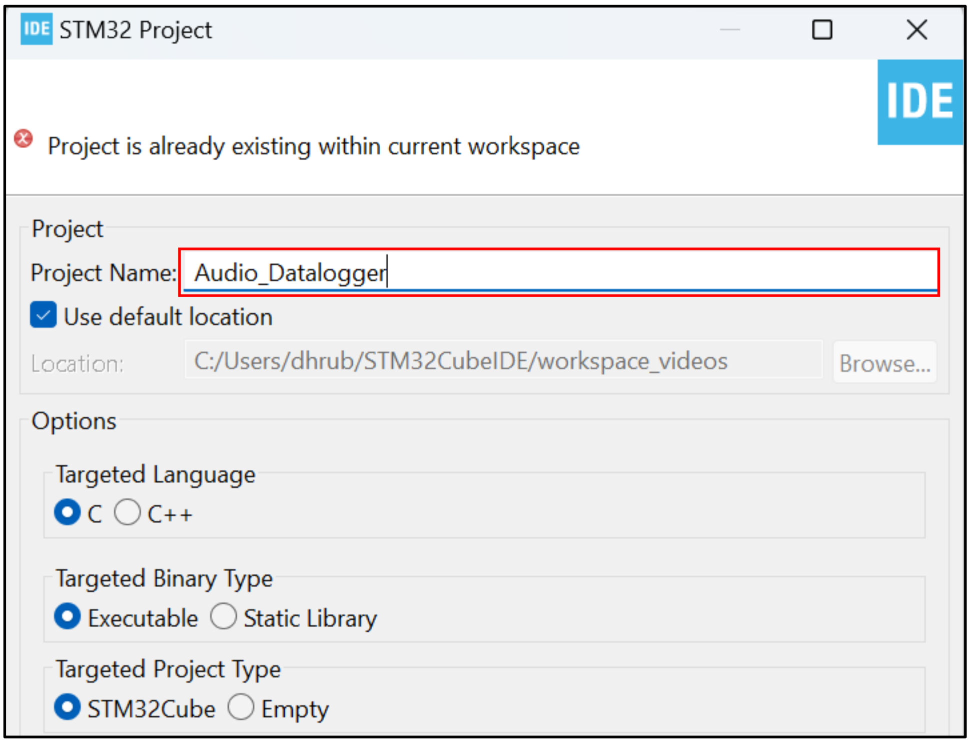
2. A **Target Selection** window will open. Click on **Board Selector**, where you need to select the microcontroller board you are working with.



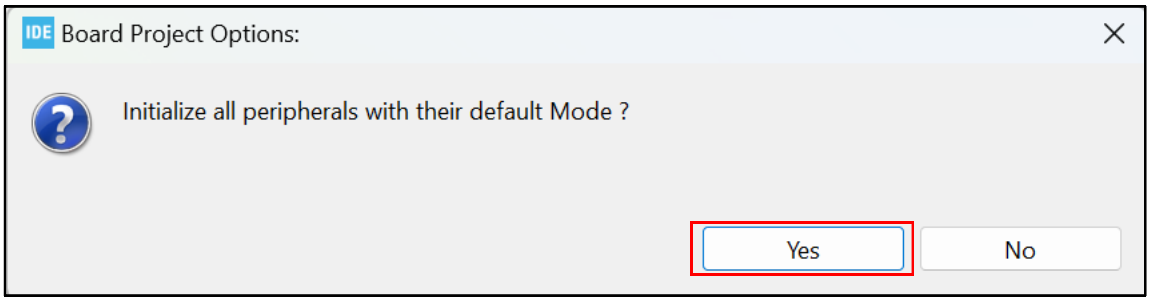
3. After this on the right-hand side of the window, under**Board List** you will see the board you have selected. Click on the board and then click on **Next.**



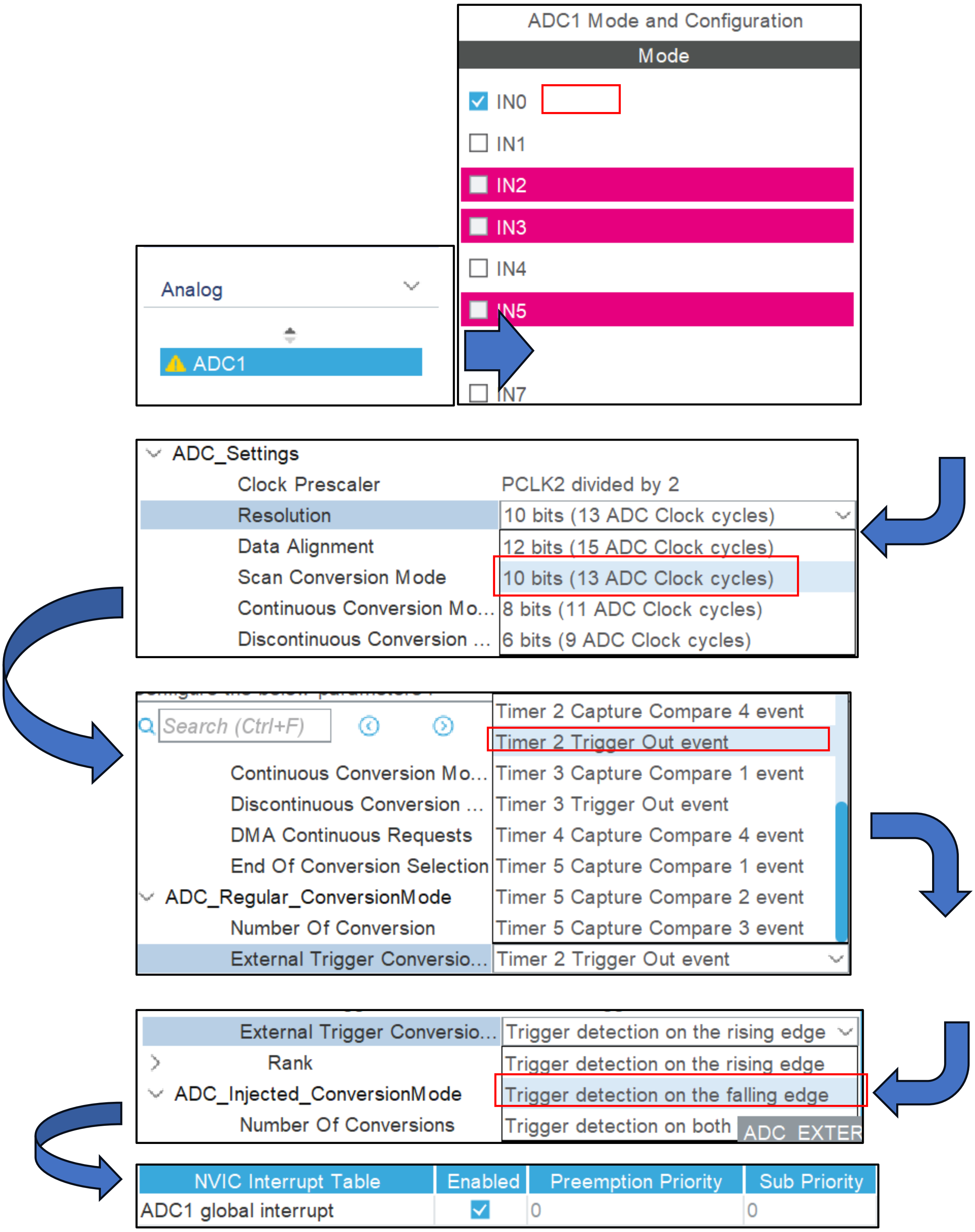
4. In the next window you need to give your project a name, rest of the things will remain by default as it is for now. Click on **Next.**



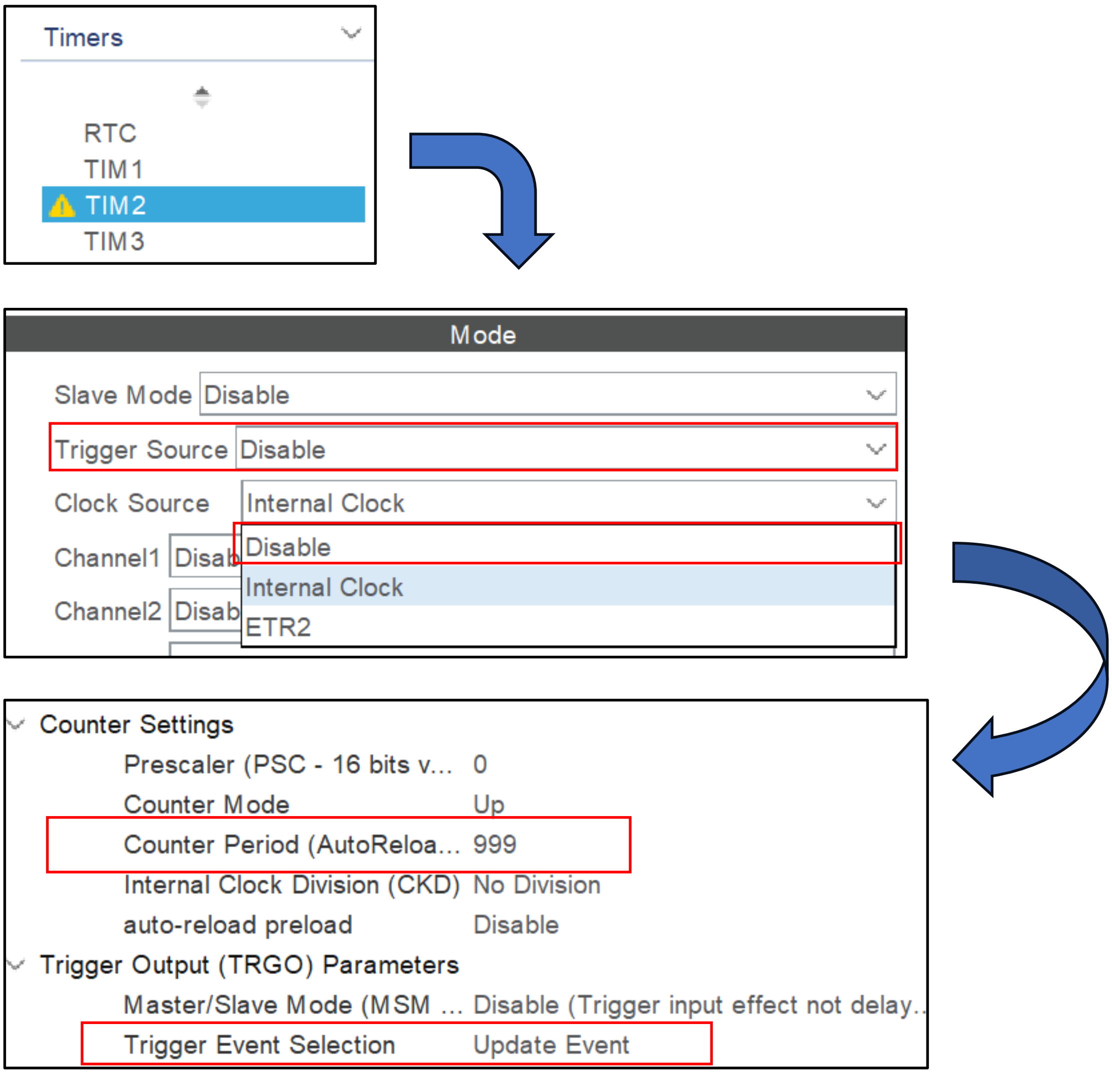
5. Cube IDE will ask if you want to initialize all peripherals with their default mode, click on **Yes.**



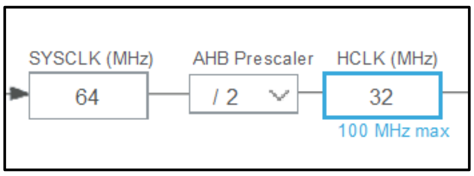
6. Next on the left-hand side**Categories →Analog**select **ADC1**then select **IN0**under **Mode.** Under **Configuration**select **Parameter Settings,** change **Resolution** to 10-bits, then change the **External Trigger ConversionSource** to **Timer 2 Trigger Out Event**and **External Trigger Conversion Mode** to **Trigger detection on the falling edge**. Under**NVIC**enable the **ADC1 global interrupt.**Otherwise, you can also go to **System Core →NVIC** and enable the same.



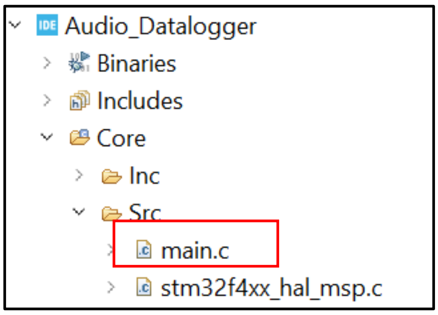
7. Next go to **Timer**, select **TIM2**and select**Clock Source** as **Internal Clock**. Update the **Counter Period** value as **999** and change **Trigger Event Selection** to **Update Event.**



8. Go to **Clock Configuration**, enter 32 as a value in **HCLK** and press enter to configure the peripheral and timer clocks. In this experiment we are trying to sample the audio signals at 32 KHz.



9. Press C**trl+S**to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example I have named my project as (Audio_Datalogger)**Audio_Datalogger→Core →Src →main.c (**double click to load the code).



10. Cube IDE automatically generates a code format based on the configurations you have done. Cube IDE uses HAL libraries. Below is the code snippets, please put your code in the appropriate places in the **main.c** file.

```c
/* Private includes ------------
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
/* USER CODE END Includes */
```

```c
/* USER CODE BEGIN PD */
#define DATA_INPUT_USER 256
#define AXIS_NUMBER 1
/* USER CODE END PD */
```

```c
/* USER CODE BEGIN PV */
float mic_x = 0.0;
float mic_buffer[DATA_INPUT_USER * AXIS_NUMBER] = {0};
volatile uint32_t buffer_index = 0;
/* USER CODE END PV */
```

```c
/* USER CODE BEGIN PFP */
void Log(void);
void fill_mic_buffer(void);
int __io_putchar(int);
/* USER CODE END PFP */
```

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if (hadc->Instance == ADC1)
      {
        mic = HAL_ADC_GetValue(&hadc1);
        mic_buffer[buffer_index] = mic;

        buffer_index++;
         if (buffer_index >= DATA_INPUT_USER)
            {
              buffer_index = 0;  // Reset index to 0 when it reaches the buffer size
            }
      }
}
/* USER CODE END 0 */
```

```
  /* USER CODE BEGIN 2 */
HAL_ADC_Start_IT(&hadc1);
HAL_TIM_Base_Start(&htim2);
  /* USER CODE END 2 */
```

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        Log();
    /* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 4 */
int __io_putchar(int ch){
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
void fill_mic_buffer(){
    for(int i=0; i<DATA_INPUT_USER; i++){
        mic_buffer[AXIS_NUMBER * i] = mic_x;
        HAL_Delay(3);
    }
}
void Log(){
    fill_mic_buffer();
    for(int i=0; i<DATA_INPUT_USER; i++){
        printf("%.2f", mic_buffer[AXIS_NUMBER * i]);
        printf(" ");
    }
    printf("\r\n");
}
```

11. Right click on the Audio_Classification project and select **Properties**. Go to **C/C++ Build →Settings.** Next select **MCU Settings** and enable the option **Use float with printf from newlib-nano (-u_printf_float).**

MCU Toolchain
MCU Settings
MCU Post build outputs

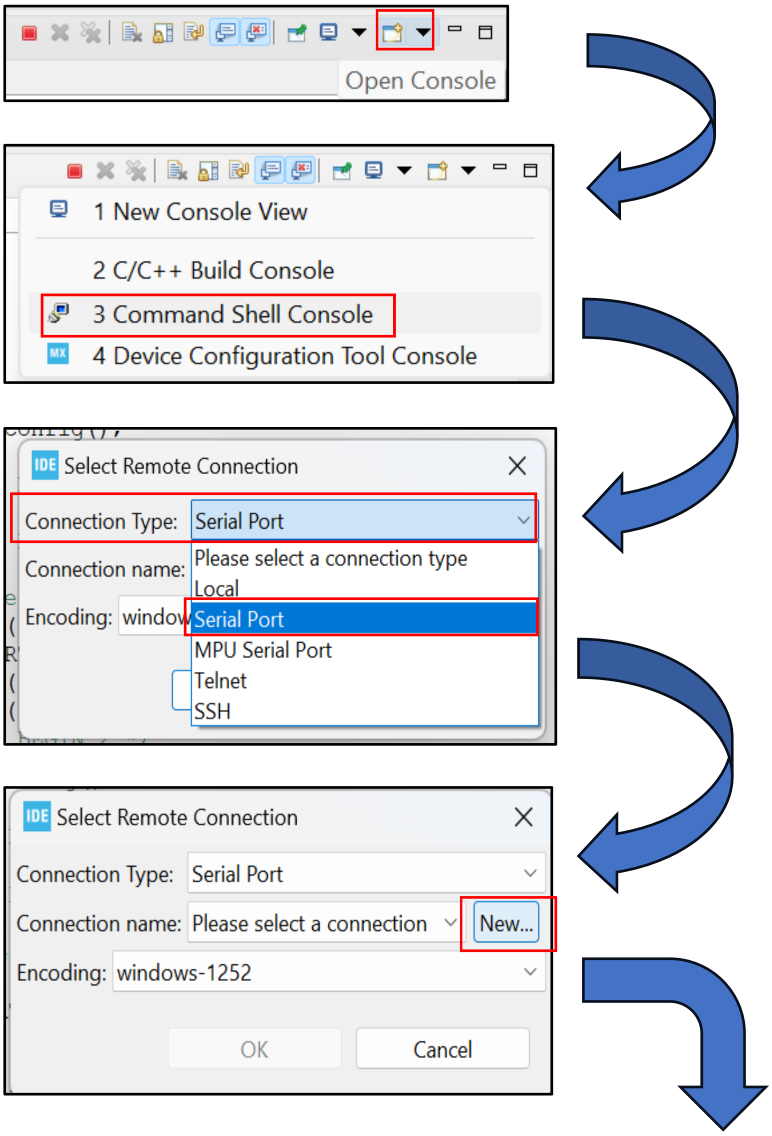
☑ Use float with printf from newlib-nano (-u _printf_float)

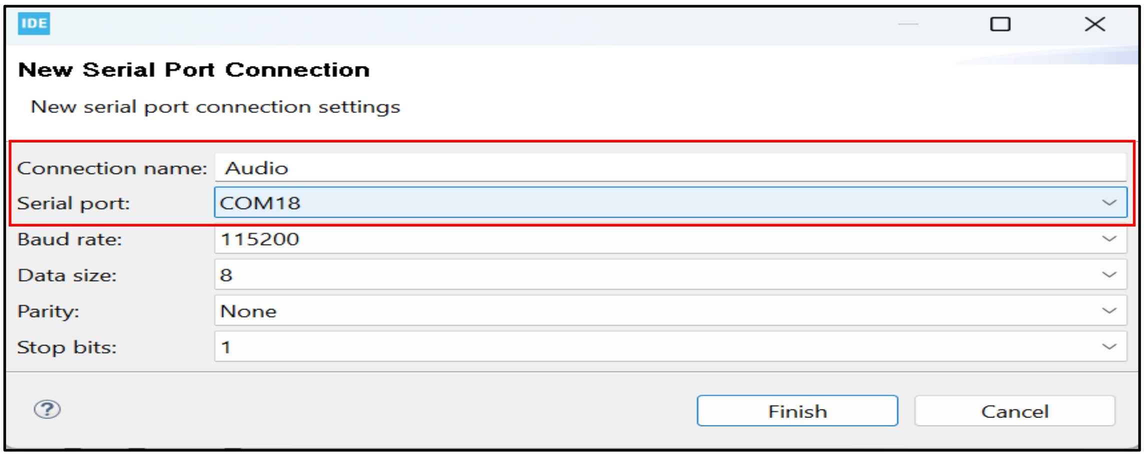
12. Now click on the build symbol on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.

```
CDT Build Console [Audio_Datalogger]

Finished building: Audio_Datalogger.list

14:32:13 Build Finished. 0 errors, 0 warnings. (took 5s.694ms)
```

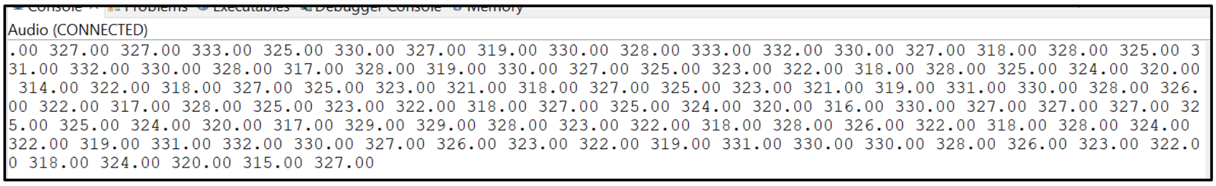
13. Next connect your STM32 board with your audio sensor connect to it to your PC and click on the **Debug** icon to start the Debugging process. An**Edit Configuration** window will open, click on **OK,** without making any changes.

14. In the debug mode, go to the bottom right hand side corner, click on open console. Selectthe **Connection Type** as **Serial Port**, then click on **New.** In the new window, in **Connection name** give some name to your new connection, and select the **Serial port** correctly. Then click on **Finish** and then **Ok.** A console with the given name will be opened at the bottom of your screen.





15. Click on the **Resume** icon to run your code. You should be able to see the value ofaudio sensor in the form of buffer containing 256 samples.



16. Before moving out of the debugging mode, click on **Disconnect** and close the console then click on the **Terminate** icon . You will be moved out of the debugging mode.

Note: All important steps and parts are highlighted with a red colour box for the proper understanding of the user. This document is for the use of education purpose only.