Audio Scene Classification: Design of an ML based audio classification to identify & classify different sounds

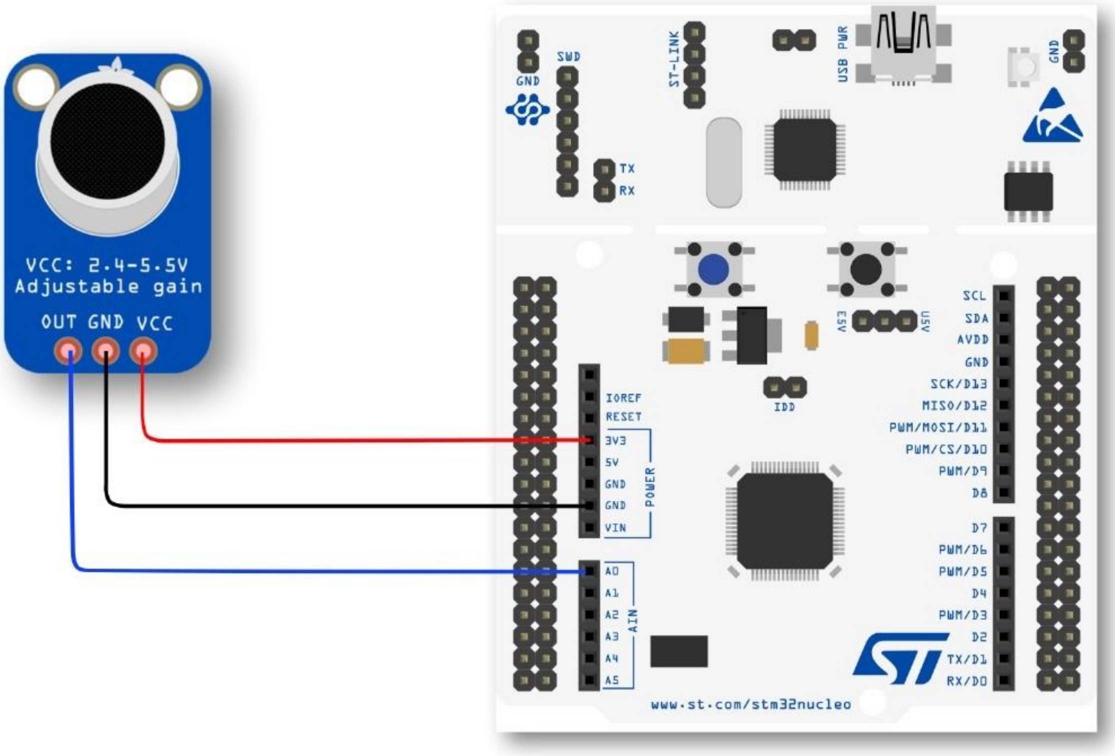
Objective:

The objective of this experiment is to interface an audio sensor to an STM32 microcontroller and deploy the Machine Learning model built using the NanoEdge AI Studio into the microcontroller. This will give the microcontroller the ability to make a decision on the device itself based on classification on real time audio sensor data.

Requirements:

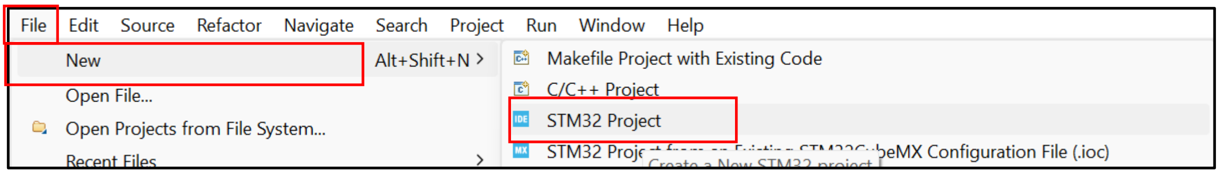
1. STM32 Cube IDE software.
2. Audio Sensor (Analog).
3. STM32 Microcontroller.
4. USB Cable for the microcontroller.
5. Jumper Wires.
6. PC or Laptop.

Connection Diagram:

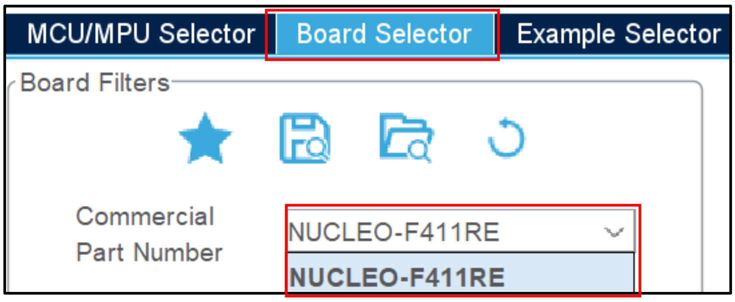


Procedure:

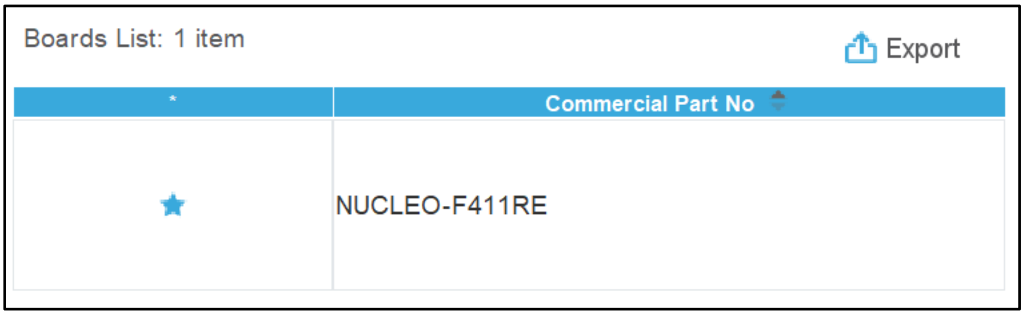
1. Click on **File→New→STM32 Project**to start your project on Cube IDE.


File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
New
Alt+Shift+N >
Open File...
Open Projects from File System...
Recent Files >
Makefile Project with Existing Code
C/C++ Project
STM32 Project
STM32 Project from an Existing STM32CubeMX Configuration File (.ioc)
Create a New STM32 project

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

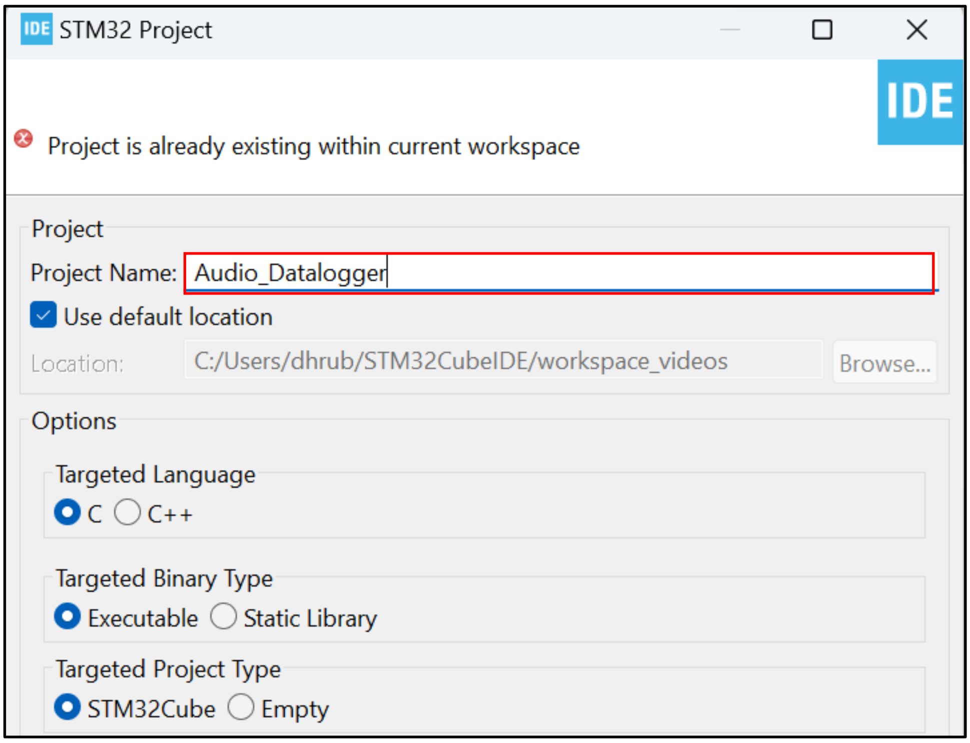


3. After this on the right-hand side of the window, under **Board List** you will see the board you have selected. Click on the board and then click on **Next.**



*	Commercial Part No
★	NUCLEO-F411RE

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 **Finish.**



STM32 Project

Project is already existing within current workspace

Project

Project Name: Audio_Datalogger

☑ Use default location

Location: C:/Users/dhrub/STM32CubeIDE/workspace_videos

Options

Targeted Language

● C ○ C++

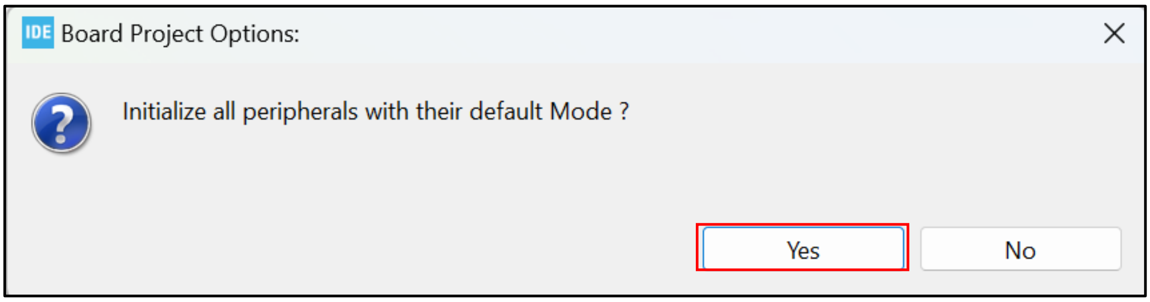
Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

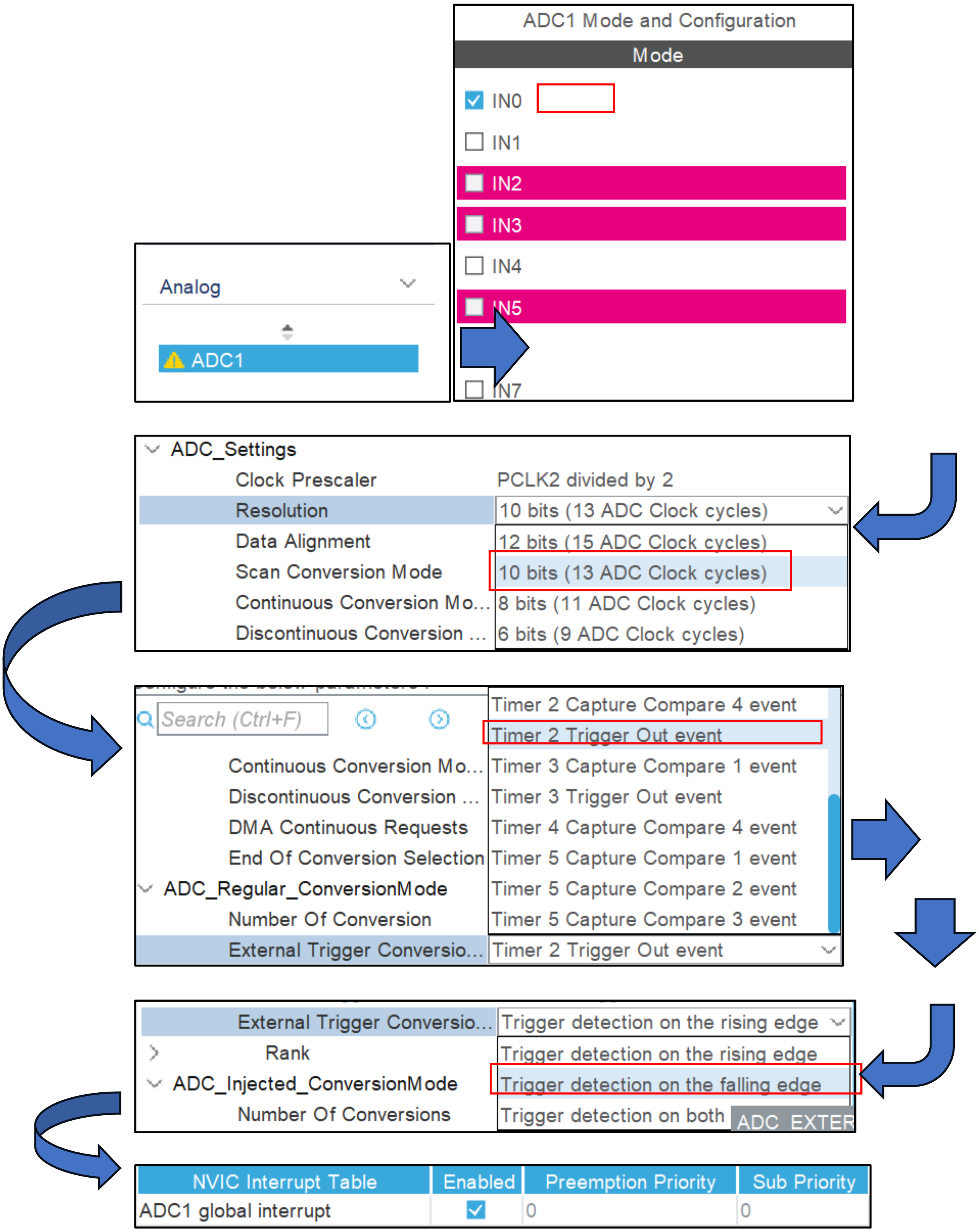
5. Cube IDE will ask if you want to initialize all peripherals with their default mode, click on **Yes.**



Board Project Options:

Initialize all peripherals with their default Mode ?

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 Conversion Source** 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.



Analog

⚠ ADC1

ADC1 Mode and Configuration

Mode

☑ IN0

□ IN1

□ IN2

□ IN3

□ IN4

□ IN5

□ IN7

ADC_Settings	
Clock Prescaler	PCLK2 divided by 2
Resolution	10 bits (13 ADC Clock cycles)
Data Alignment	12 bits (15 ADC Clock cycles)
Scan Conversion Mode	10 bits (13 ADC Clock cycles)
Continuous Conversion Mo...	8 bits (11 ADC Clock cycles)
Discontinuous Conversion ...	6 bits (9 ADC Clock cycles)

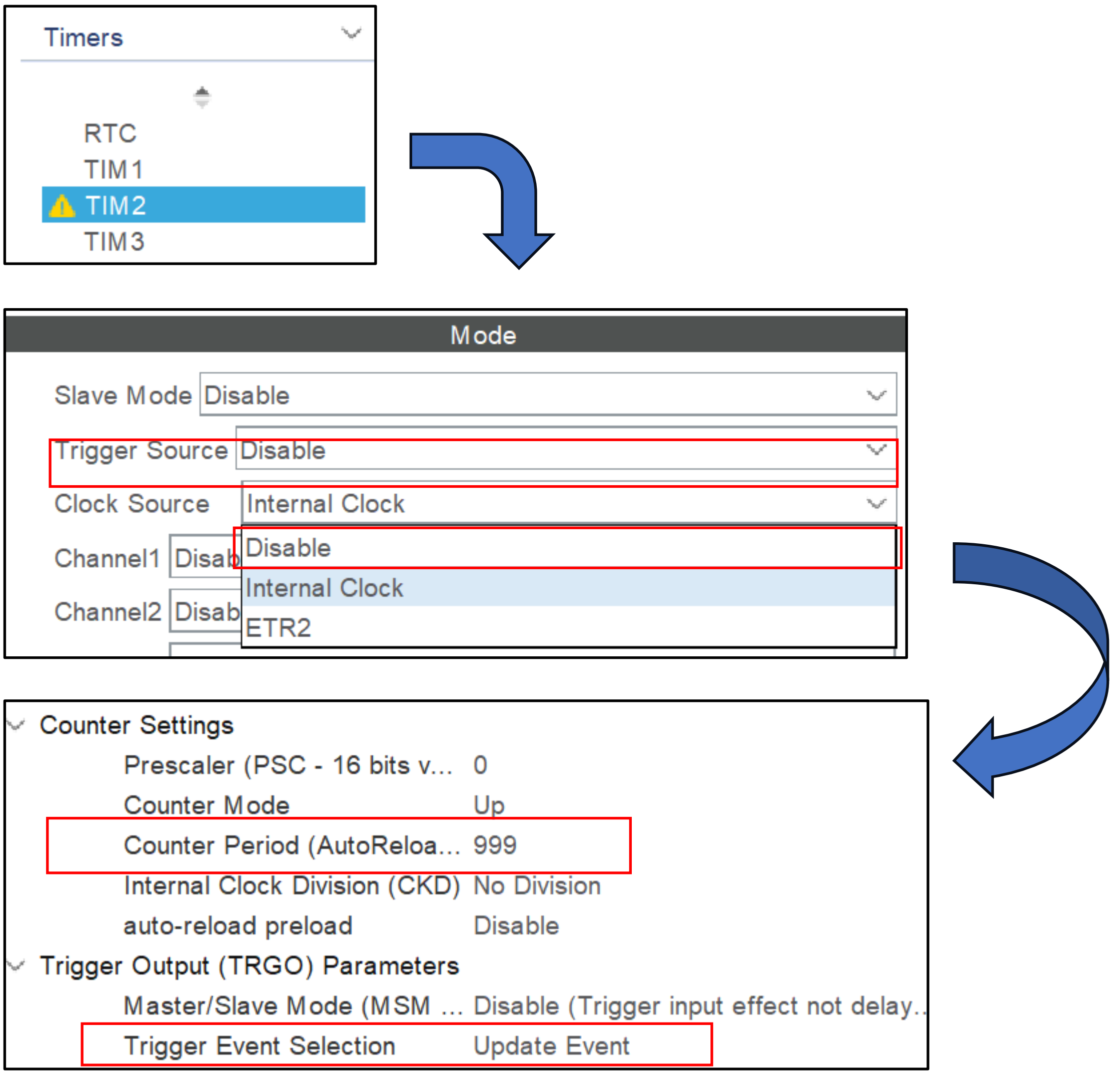
Search (Ctrl+F)

	Timer 2 Capture Compare 4 event
	Timer 2 Trigger Out event
Continuous Conversion Mo...	Timer 3 Capture Compare 1 event
Discontinuous Conversion ...	Timer 3 Trigger Out event
DMA Continuous Requests	Timer 4 Capture Compare 4 event
End Of Conversion Selection	Timer 5 Capture Compare 1 event
ADC_Regular_ConversionMode	Timer 5 Capture Compare 2 event
Number Of Conversion	Timer 5 Capture Compare 3 event
External Trigger Conversio...	Timer 2 Trigger Out event

External Trigger Conversio...	Trigger detection on the rising edge
Rank	Trigger detection on the rising edge
ADC_Injected_ConversionMode	Trigger detection on the falling edge
Number Of Conversions	Trigger detection on both

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 global interrupt	☑	0	0

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.**



Timers

RTC
TIM1
TIM2
TIM3

Mode

Slave Mode Disable

Trigger Source Disable

Clock Source Internal Clock

Channel1 Disab

Channel2 Disab

Disable
Internal Clock
ETR2

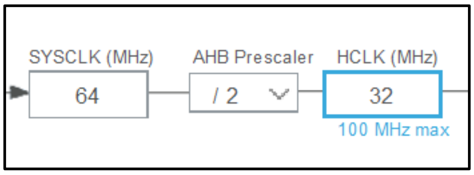
Counter Settings

Prescaler (PSC - 16 bits v…	0
Counter Mode	Up
Counter Period (AutoReloa…	999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

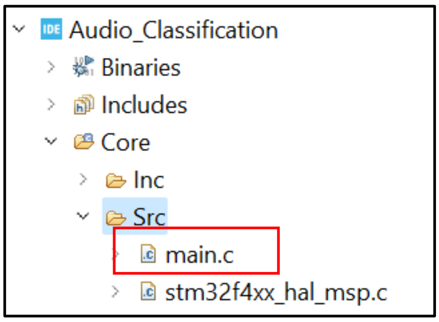
Master/Slave Mode (MSM …	Disable (Trigger input effect not delay…
Trigger Event Selection	Update Event

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.

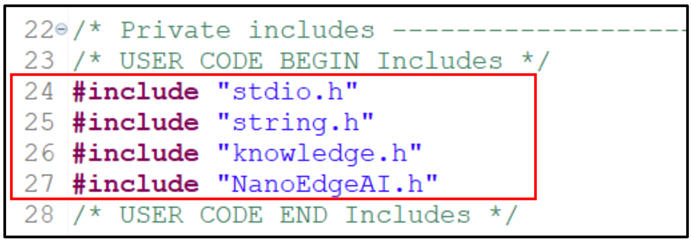


SYSCLK (MHz)	AHB Prescaler	HCLK (MHz)
64	/ 2	32

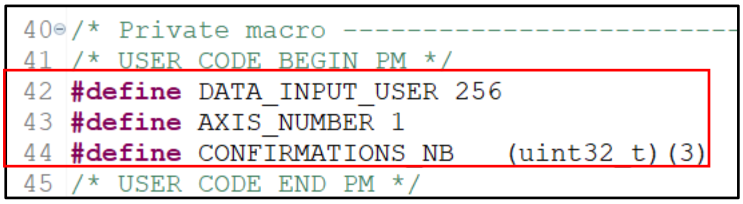
9. Press C**trl+S**to generate your code. On the left-hand side of the Cube IDE, under **Project Explorer** go to the project you have created (For example I have named my project as (Audio_Classification)**Audio_Classification→Core →Src→main.c (**double click to load the code).



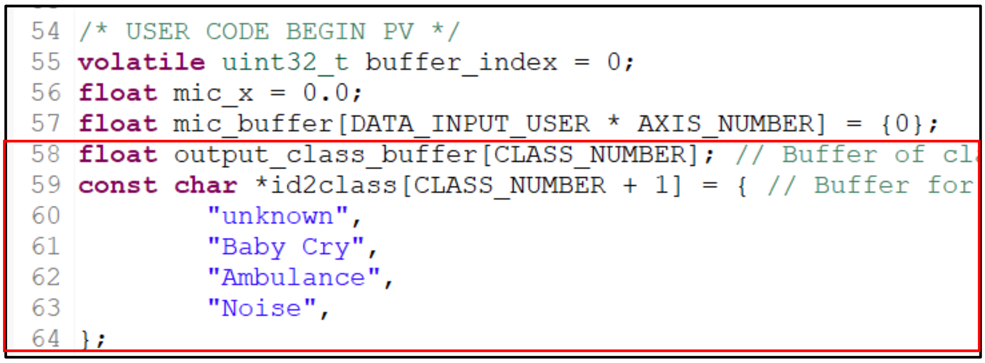
10. Cube IDE automatically generates the initialization codebased on the configurations you have done. Cube IDE uses HAL libraries. Below are the code snippets, please put your code in the appropriate places in the **main.c** file.The highlighted part where the class names are written should be the same as given in the **NanoEdgeAI.h** file downloaded from NanoEdge AI Studio in the **Deployment** stage.



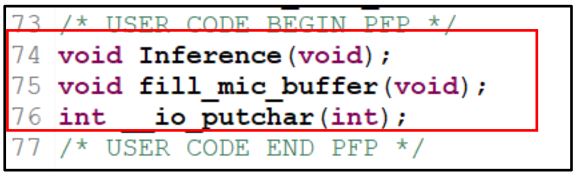
```
/* Private includes --------------------
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
#include "knowledge.h"
#include "NanoEdgeAI.h"
/* USER CODE END Includes */
```



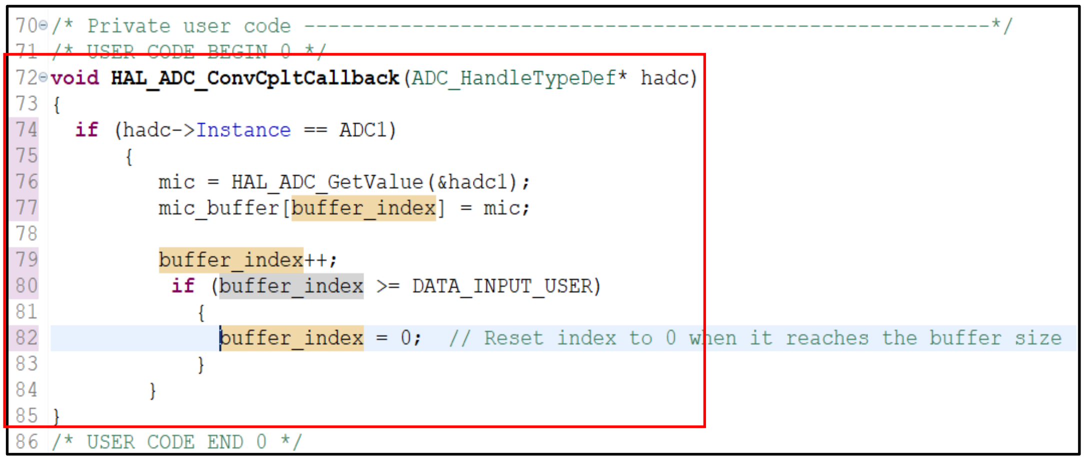
```
/* Private macro -------------------------
/* USER CODE BEGIN PM */
#define DATA_INPUT_USER 256
#define AXIS_NUMBER 1
#define CONFIRMATIONS_NB    (uint32_t)(3)
/* USER CODE END PM */
```



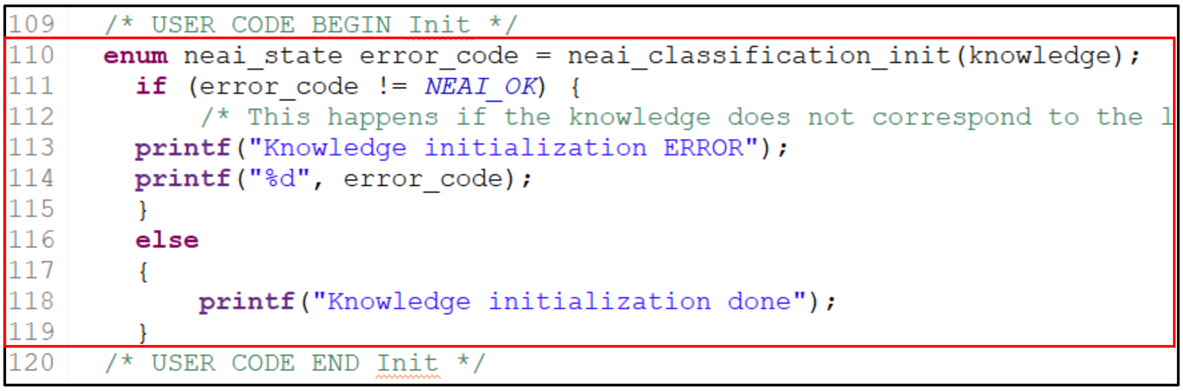
```
/* USER CODE BEGIN PV */
volatile uint32_t buffer_index = 0;
float mic_x = 0.0;
float mic_buffer[DATA_INPUT_USER * AXIS_NUMBER] = {0};
float output_class_buffer[CLASS_NUMBER]; // Buffer of cl
const char *id2class[CLASS_NUMBER + 1] = { // Buffer for
        "unknown",
        "Baby Cry",
        "Ambulance",
        "Noise",
};
```



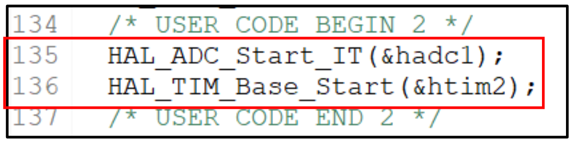
```
/* USER CODE BEGIN PFP */
void Inference(void);
void fill_mic_buffer(void);
int __io_putchar(int);
/* USER CODE END PFP */
```



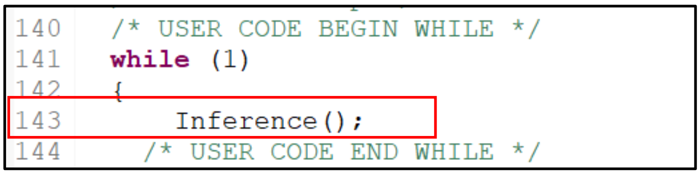
```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if (hadc->Instance == ADC1)
      {
         mic = HAL_ADC_GetValue(&hadc1);
         mic_buffer[buffer_index] = mic;

         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 */
```



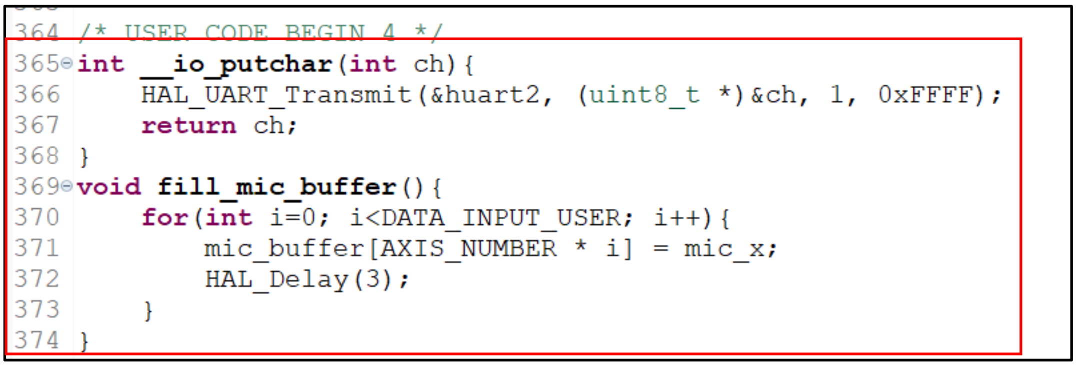
```
  /* USER CODE BEGIN Init */
  enum neai_state error_code = neai_classification_init(knowledge);
    if (error_code != NEAI_OK) {
        /* This happens if the knowledge does not correspond to the l
    printf("Knowledge initialization ERROR");
    printf("%d", error_code);
    }
    else
    {
        printf("Knowledge initialization done");
    }
  /* USER CODE END Init */
```



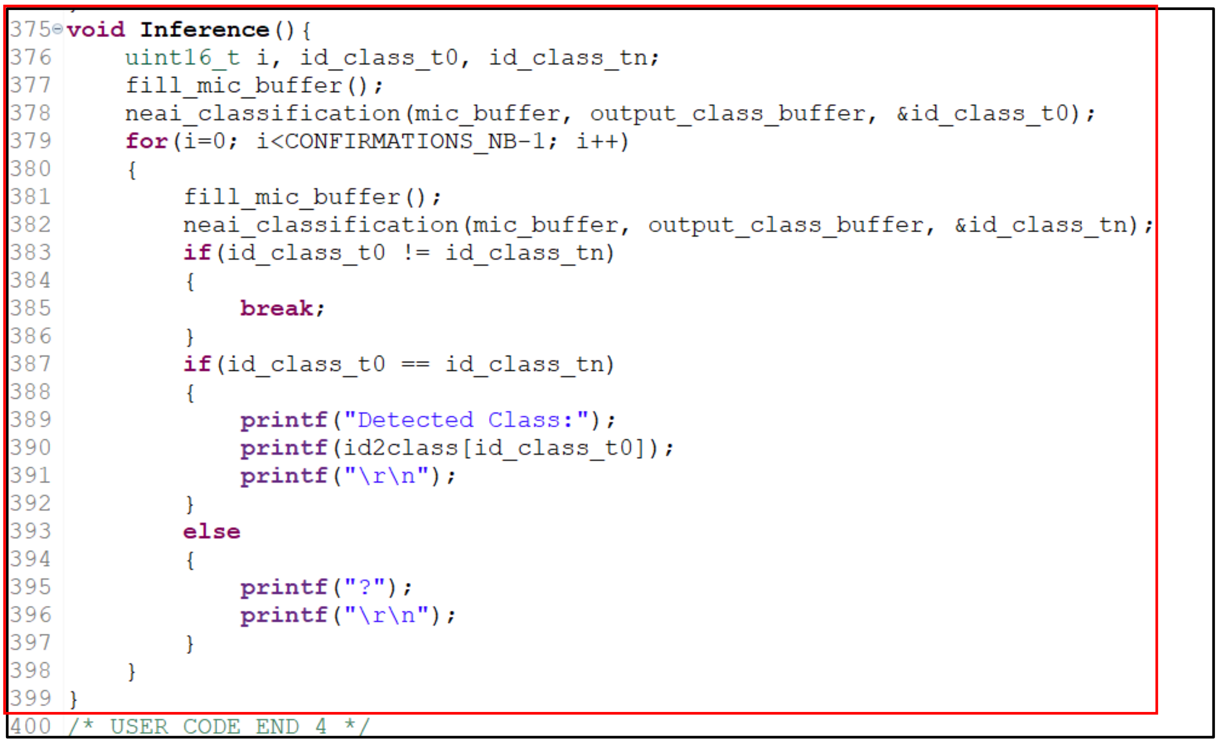
```
  /* USER CODE BEGIN 2 */
  HAL_ADC_Start_IT(&hadc1);
  HAL_TIM_Base_Start(&htim2);
  /* USER CODE END 2 */
```



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

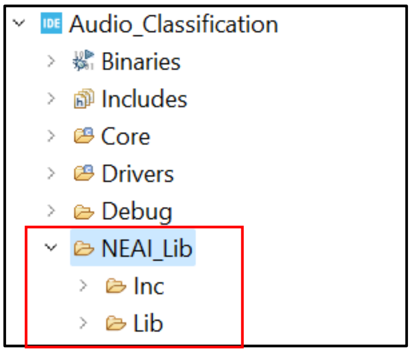


```
/* USER CODE BEGIN 4 */
int __io_putchar(int ch){
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
void fill_mic_buffer(){
    for(int i=0; i<DATA_INPUT_USER; i++){
        mic_buffer[AXIS_NUMBER * i] = mic_x;
        HAL_Delay(3);
    }
}
```



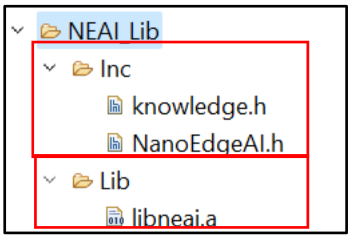
```
void Inference(){
    uint16_t i, id_class_t0, id_class_tn;
    fill_mic_buffer();
    neai_classification(mic_buffer, output_class_buffer, &id_class_t0);
    for(i=0; i<CONFIRMATIONS_NB-1; i++)
    {
        fill_mic_buffer();
        neai_classification(mic_buffer, output_class_buffer, &id_class_tn);
        if(id_class_t0 != id_class_tn)
        {
            break;
        }
        if(id_class_t0 == id_class_tn)
        {
            printf("Detected Class:");
            printf(id2class[id_class_t0]);
            printf("\r\n");
        }
        else
        {
            printf("?");
            printf("\r\n");
        }
    }
}
/* USER CODE END 4 */
```

11. Right click on the Audio_Classification project and select **New→Folder**. Name the new folder as NEAI and click on finish.You will be able to see a folder named NEAI inside of your Audio_Classification project. Now right click on this newly created NEAI folder and create two separate new folders and name them Inc and Lib.



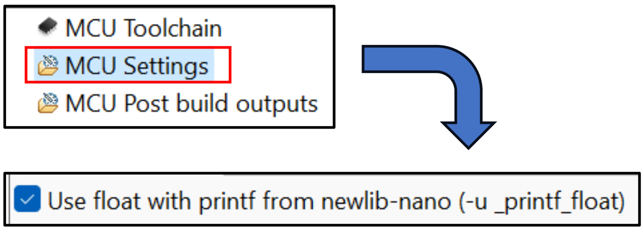
```
Audio_Classification
  Binaries
  Includes
  Core
  Drivers
  Debug
  NEAI_Lib
    Inc
    Lib
```

12. Now inside the Inc folder copy and paste the NanoEdgeAI.h and knowledge.h files. Inside the Lib folder copy the libneai.a file. These files are in the folder you downloaded from NanoEdge AI Studio during the deployment phase of Machine Learning model building.



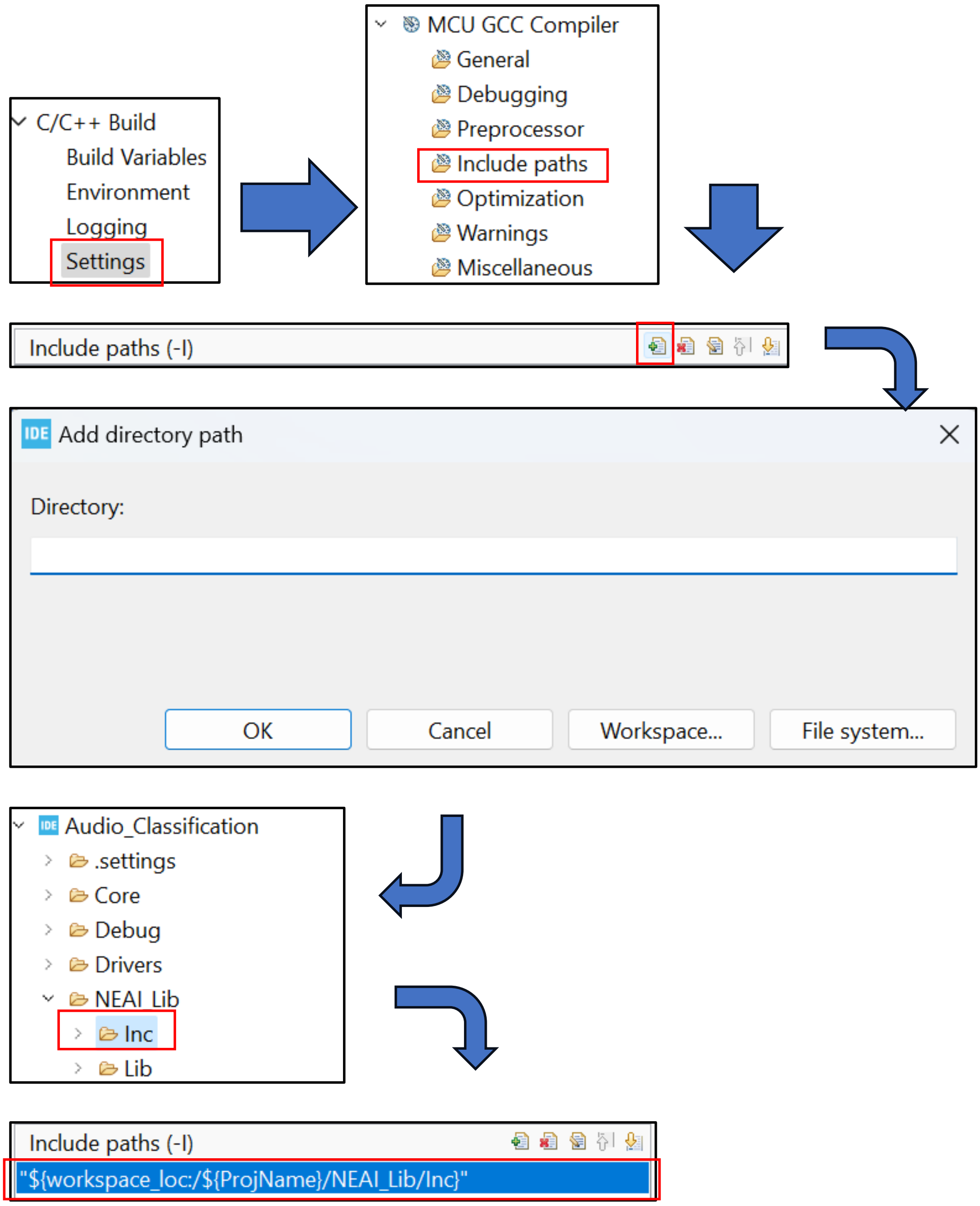
```
NEAI_Lib
  Inc
    knowledge.h
    NanoEdgeAI.h
  Lib
    libneai.a
```

13. 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).**



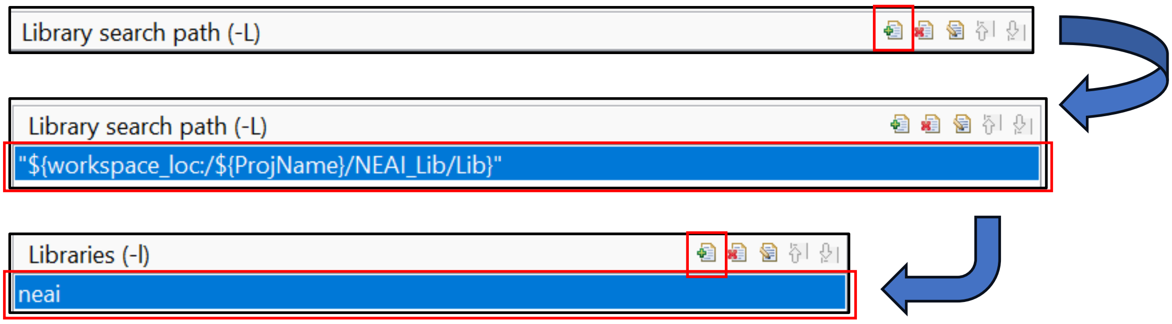
☑ Use float with printf from newlib-nano (-u _printf_float)

14. Next select **MCU GCC Compiler →Include paths** click on **Add** and in the new window select **Workspace** and select the **Inc** folder and click on **Ok.** The path of the **Inc** folder will be now added.



"${workspace_loc:/${ProjName}/NEAI_Lib/Inc}"

15. Next go to **MCU GCC Linker →Libraries**. In **Library search path (-L)** section, add the path of **Lib** folder similarly to the previous step of adding the **Inc** folder path. Next in the **Libraries(-l)** section click on **Add** and type **neai** then click on **Ok.**

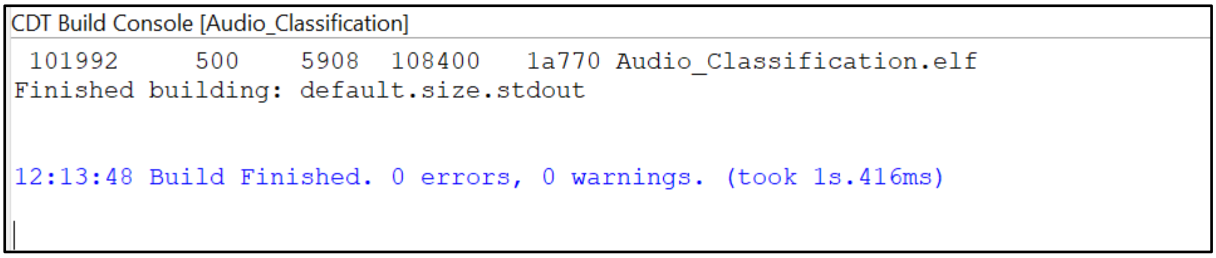


Library search path (-L)

Library search path (-L)
"${workspace_loc:/${ProjName}/NEAI_Lib/Lib}"

Libraries (-l)
neai

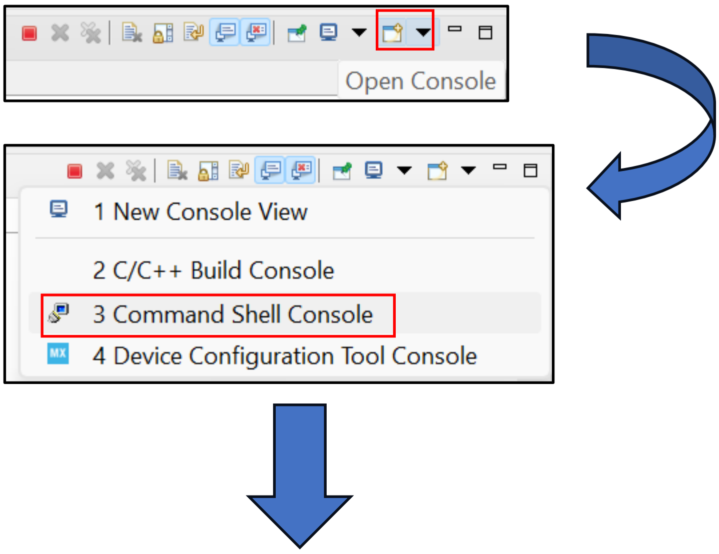
16. Now click on the build symbol on the top left corner on your Cube IDE. If you have done everything correctly your code should be built without any errors.

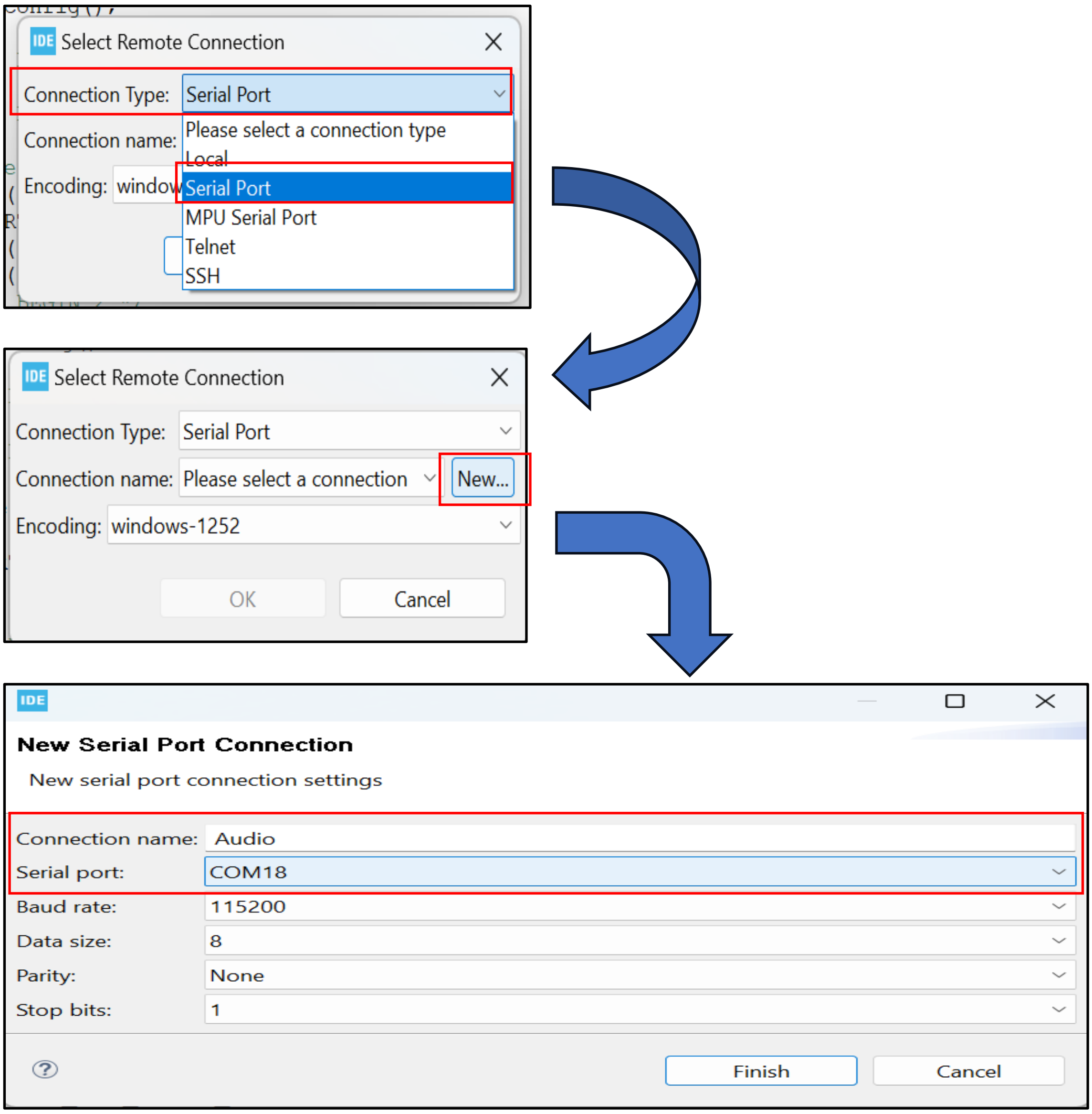


CDT Build Console [Audio_Classification]

```
 101992     500    5908   108400    1a770 Audio_Classification.elf
Finished building: default.size.stdout


12:13:48 Build Finished. 0 errors, 0 warnings. (took 1s.416ms)
```

17. Next connect your STM32 board with your audio sensor connect 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.
18. 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.





Select Remote Connection

Connection Type: Serial Port

Connection name:

Encoding: window

Select Remote Connection

Connection Type: Serial Port

Connection name: Please select a connection

Encoding: windows-1252

New Serial Port Connection

New serial port connection settings

Connection name: Audio

Serial port: COM18

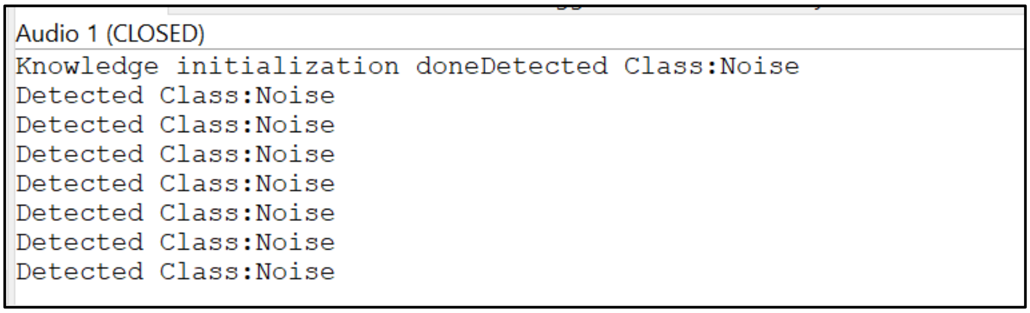
Baud rate: 115200

Data size: 8

Parity: None

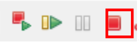
Stop bits: 1

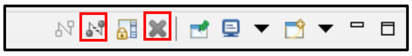
19. Click on the **Resume** icon to run your code. You should be able to see the output as different classes based on your project. Based on the audio sensor data, your model will identify which class the data belongs to.



Audio 1 (CLOSED)

```
Knowledge initialization doneDetected Class:Noise
Detected Class:Noise
Detected Class:Noise
Detected Class:Noise
Detected Class:Noise
Detected Class:Noise
Detected Class:Noise
Detected Class:Noise
```

20. 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.