4.2. STM32F4 MEMS project preparation

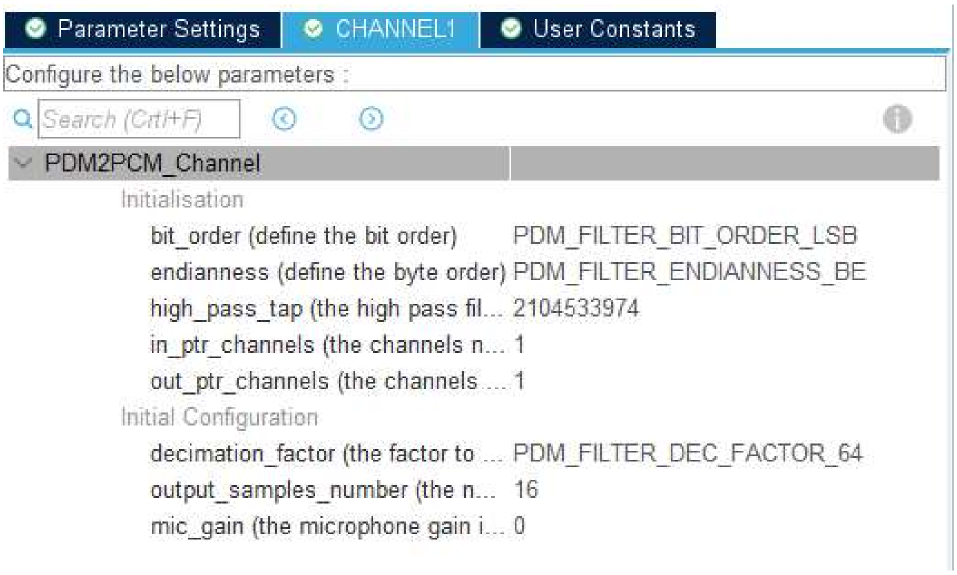
New project using STM32CubeIDE

1. Select menu: *File -> New -> STM32 Project*
2. In the tab *"Board Selector"* select STM32F407G-DISC1 board. Complete project wizard and initialize all peripherals of the board as suggested.
3. When the integrated STM32CubeMX configurator opens, make the following changes to the MEMS project:
    a. Initialize *Multimedia -> I2S2*:
        - Mode: Half-Duplex Master, set Master Clock Output
        - Transmission Mode: Mode Master Receive
        - Communication Standard: I2S Philips
        - Data and Frame Format: 16 Bits Data on 16 Bits Format
        - Selected Audio Frequency: 16 KHz
        - In DMA Settings tab add DMA stream, Peripheral To Memory, Half Word, Mode: Circular

    b. Initialize *Multimedia -> I2S3:*
        - Mode: Half-Duplex Master, set Master Clock Output
        - Transmission Mode: Mode Master Transmit
        - Communication Standard: I2S Philips
        - Data and Frame Format: 16 Bits Data on 16 Bits Format
        - Selected Audio Frequency: 96 KHz
        - In DMA Settings tab add DMA stream, Memory To Peripheral, Half Word, Mode: Normal

    c. Activate *Computing -> CRC*

    d. Enable *Middleware -> PDM2PCM:*
        - In the Tab Parameter Settings set 1 channel
        - Channel1 parameters:



Configure the below parameters :

PDM2PCM_Channel

Initialisation

bit_order (define the bit order)	PDM_FILTER_BIT_ORDER_LSB
endianness (define the byte order)	PDM_FILTER_ENDIANNESS_BE
high_pass_tap (the high pass fil...	2104533974
in_ptr_channels (the channels n...	1
out_ptr_channels (the channels	1

Initial Configuration

decimation_factor (the factor to ...	PDM_FILTER_DEC_FACTOR_64
output_samples_number (the n...	16
mic_gain (the microphone gain i...	0

4. Generate project code and add DSP libraries:
    a. Copy the following folders to the project (from the STM32Cube\Repository):
        – .\Drivers\CMSIS\DSP\Include

– .\Drivers\CMSIS\DSP\Source
– .\Drivers\CMSIS\Lib\
– .\Drivers\BSP\Components\
– .\Drivers\BSP\STM32F4-Discovery\

b. Open *Project -> Properties -> C/C++ General -> Paths and Symbols,* and add the following libraries and symbols to the project:
   - In the Tab *Libraries* add **arm_cortexM4lf_math** (just copy and paste this expression)
   - In the Tab *Library Paths* add path to library file *libarm_cortexM4lf_math.a*, e.g., **Drivers\CMSIS\Lib\GCC**
   - In the Tab *Symbols* add **ARM_MATH_CM4**
   - In the Tab *Symbols* add **__FPU_PRESENT,** Value: 1
   - In the Tab *Includes* add path: **Drivers\CMSIS\DSP\Include**
   - In the Tab *Includes* add path: **Drivers\BSP\STM32F4-Discovery**

MEMS Microphone program example

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "pdm2pcm.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32f4_discovery_audio.h"
#include "arm_math.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define SAMPLING_FREQ 44100                    //BSP_audio_out sampling frequency
```

```
#define OUTPUT_BUFFER_SIZE 16000               //output buffer size

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
CRC_HandleTypeDef hcrc;
I2C_HandleTypeDef hi2c1;
I2S_HandleTypeDef hi2s2;
I2S_HandleTypeDef hi2s3;
DMA_HandleTypeDef hdma_spi2_rx;
DMA_HandleTypeDef hdma_spi3_tx;
SPI_HandleTypeDef hspi1;

/* USER CODE BEGIN PV */

            //*** BSP Audio IN ***//
            static uint16_t InternalBuffer[INTERNAL_BUFF_SIZE];
                        //PDM samples. Size defined stm32f4_discovery_audio.h
            static uint16_t RecBuf[PCM_OUT_SIZE*2];
                        //PCM stereo samples. Size defined  stm32f4_discovery_audio.h

            static int16_t OutputBuffer[OUTPUT_BUFFER_SIZE];
                        //output signal to BSP_AUDIO_OUT, left+right channels
            volatile uint32_t ITCounter = 0;
                        //Counter for the transfer to OutputBuffer

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S3_Init(void);
static void MX_SPI1_Init(void);
static void MX_DMA_Init(void);
static void MX_I2S2_Init(void);
static void MX_CRC_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

/* Configure the peripherals common clocks */
  PeriphCommonClock_Config();

  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_DMA_Init();
  MX_I2S2_Init();
  MX_CRC_Init();
  MX_PDM2PCM_Init();
  /* USER CODE BEGIN 2 */

  /*** Init wave rec. Audio freq. to be configured for the I2S peripheral. ***/
      if(BSP_AUDIO_IN_Init(DEFAULT_AUDIO_IN_FREQ, DEFAULT_AUDIO_IN_BIT_RESOLUTION,
DEFAULT_AUDIO_IN_CHANNEL_NBR) != AUDIO_OK)
      {
            Error_Handler();
      }

  /*** Init Audio codec and all related peripherals (I2S, I2C, IOExpander, IOs...) ***/
      if(BSP_AUDIO_OUT_Init(OUTPUT_DEVICE_AUTO, 70, SAMPLING_FREQ) != AUDIO_OK)
      {
            Error_Handler();
      }

  /*** Start Audio recording ***/
      BSP_AUDIO_IN_Record((uint16_t*)&InternalBuffer[0], INTERNAL_BUFF_SIZE);

  /*** Start Audio play ***/
      BSP_AUDIO_OUT_Play((uint16_t*)&OutputBuffer[0], OUTPUT_BUFFER_SIZE);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
              HAL_GPIO_TogglePin(LD4_GPIO_Port, LD4_Pin);
              HAL_Delay(200);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
```

```
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 7;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief Peripherals Common Clock Configuration
  * @retval None
  */
void PeriphCommonClock_Config(void)
{
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

  /** Initializes the peripherals clock
  */
  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2S;
  PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;
  PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```
/**
  * @brief CRC Initialization Function
  * @param None
  * @retval None
  */
static void MX_CRC_Init(void)
{

  /* USER CODE BEGIN CRC_Init 0 */
  /* USER CODE END CRC_Init 0 */
  /* USER CODE BEGIN CRC_Init 1 */
  /* USER CODE END CRC_Init 1 */
  hcrc.Instance = CRC;
  if (HAL_CRC_Init(&hcrc) != HAL_OK)
  {
    Error_Handler();
  }
  __HAL_CRC_DR_RESET(&hcrc);
  /* USER CODE BEGIN CRC_Init 2 */
  /* USER CODE END CRC_Init 2 */

}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */
  /* USER CODE END I2C1_Init 0 */
  /* USER CODE BEGIN I2C1_Init 1 */
  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */
  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief I2S2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2S2_Init(void)
```

```
{

  /* USER CODE BEGIN I2S2_Init 0 */
  /* USER CODE END I2S2_Init 0 */
  /* USER CODE BEGIN I2S2_Init 1 */
  /* USER CODE END I2S2_Init 1 */
  hi2s2.Instance = SPI2;
  hi2s2.Init.Mode = I2S_MODE_MASTER_RX;
  hi2s2.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s2.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
  hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_16K;
  hi2s2.Init.CPOL = I2S_CPOL_LOW;
  hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
  if (HAL_I2S_Init(&hi2s2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2S2_Init 2 */
  /* USER CODE END I2S2_Init 2 */

}

/**
  * @brief I2S3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2S3_Init(void)
{

  /* USER CODE BEGIN I2S3_Init 0 */
  /* USER CODE END I2S3_Init 0 */
  /* USER CODE BEGIN I2S3_Init 1 */
  /* USER CODE END I2S3_Init 1 */
  hi2s3.Instance = SPI3;
  hi2s3.Init.Mode = I2S_MODE_MASTER_TX;
  hi2s3.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s3.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s3.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
  hi2s3.Init.AudioFreq = I2S_AUDIOFREQ_96K;
  hi2s3.Init.CPOL = I2S_CPOL_LOW;
  hi2s3.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s3.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
  if (HAL_I2S_Init(&hi2s3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2S3_Init 2 */
  /* USER CODE END I2S3_Init 2 */

}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
```

```
{

  /* USER CODE BEGIN SPI1_Init 0 */
  /* USER CODE END SPI1_Init 0 */
  /* USER CODE BEGIN SPI1_Init 1 */
  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */
  /* USER CODE END SPI1_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream3_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream3_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream3_IRQn);
  /* DMA1_Stream5_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream5_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin,
GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                          |Audio_RST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : CS_I2C_SPI_Pin */
  GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : BOOT1_Pin */
  GPIO_InitStruct.Pin = BOOT1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BOOT1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin
                           Audio_RST_Pin */
  GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                          |Audio_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : VBUS_FS_Pin */
  GPIO_InitStruct.Pin = VBUS_FS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(VBUS_FS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : OTG_FS_ID_Pin OTG_FS_DM_Pin OTG_FS_DP_Pin */
  GPIO_InitStruct.Pin = OTG_FS_ID_Pin|OTG_FS_DM_Pin|OTG_FS_DP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

    /*** MEMS data processing ***/
    void BSP_AUDIO_IN_HalfTransfer_CallBack(void)
    {
            /* PDM to PCM data convert */
            BSP_AUDIO_IN_PDMToPCM((uint16_t*)&InternalBuffer[0], (uint16_t*)&RecBuf[0]);

            /* Copy PCM data in internal buffer */
            memcpy((uint16_t*)&OutputBuffer[ITCounter * (PCM_OUT_SIZE*2)], RecBuf, 
PCM_OUT_SIZE*4);

            if(ITCounter == (OUTPUT_BUFFER_SIZE/(PCM_OUT_SIZE*2))-1)
                        ITCounter = 0;
            else
                        ITCounter++;
    }

    void BSP_AUDIO_IN_TransferComplete_CallBack(void)
    {
            /* PDM to PCM data convert */
            BSP_AUDIO_IN_PDMToPCM((uint16_t*)&InternalBuffer[INTERNAL_BUFF_SIZE/2], 
(uint16_t*)&RecBuf[0]);

            /* Copy PCM data in internal buffer */
            memcpy((uint16_t*)&OutputBuffer[ITCounter * (PCM_OUT_SIZE*2)], RecBuf, 
PCM_OUT_SIZE*4);

            if(ITCounter == (OUTPUT_BUFFER_SIZE/(PCM_OUT_SIZE*2))-1)
                        ITCounter = 0;
            else
                        ITCounter++;

            HAL_GPIO_TogglePin(LD6_GPIO_Port, LD6_Pin);
    }

    /*** Audio data output ***/
    void BSP_AUDIO_OUT_HalfTransfer_CallBack(void)
    {

    }
    /*** Back to Buffer beginning ***/
    void BSP_AUDIO_OUT_TransferComplete_CallBack(void)
    {
            BSP_AUDIO_OUT_ChangeBuffer((uint16_t*)&OutputBuffer[0], OUTPUT_BUFFER_SIZE);
            HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);
    }

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
    HAL_GPIO_WritePin(LD5_GPIO_Port, LD5_Pin, GPIO_PIN_SET);

  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Lab work tasks

1. Analyze the presented program. Create an algorithm for program operation.
2. Apply one of the known filters or audio effects to the transmitted sound.