Accelerometer – Source code detail.

1. **Including files and Initialize all components.**

```
#include "main.h"
#include "stm32f4xx_hal.h"

/* Private variables ---------------------------------------------------------*/

I2C_HandleTypeDef hi2c1;
I2S_HandleTypeDef hi2s3;
TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart2;
```

2. **Variable for playing sound.**

```
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
uint16_t demoMode = 1;
uint8_t QUANTUM = 100;
uint8_t initV[2];
uint16_t Istr[100];
uint8_t note[] = {
0x02, 0x12, 0x22, 0x32, 0x42, 0x52, 0x62, 0x72, 0x82, 0x92, 0xA2, 0xB2, 0xC2,
0xD2, 0xE2, 0xF2
};
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/

void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S3_Init(void);
static void MX_TIM1_Init(void);
static void MX_USART2_UART_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

3. **Play sound.**

```
int main(void)
{

  HAL_Init();
```

```
  SystemClock_Config();

  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_TIM1_Init();
  MX_USART2_UART_Init();

  /* USER CODE END 2 */
```

3.1 Initialize all sound component.

```
  InitSound();
  /* Infinite loop */
  int i = 0;
  int j[]={-1,-1,-1,3,-1,-1,2,1,-1,2,-1,3,-1,3,-1,3,-1,-1,-1,2,-1,2,-1,2,-1,-
1,-1,3,-1,5,-1,5,-1,-1,-1,3,-1,-1,2,1,-1,2,-1,3,-1,3,-1,3,-1,-1,-1,2,-1,2,-
1,3,-1,2,-1,1};
  while (1)
  {
```

3.2 Playing sound → Mode 1 : play sound from keyboard input
Mode 2 : play nu ma lee song
Mode 3 : play all note

```
        if(demoMode==1){ // play note from keyboard.
                    DemoMode();
        }else if(demoMode==2){ // play nu ma lee song.
                    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
                    PlaySound(j[i]);
                    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
                    if(i+1==60) HAL_Delay(5000);
                    i = (i+1)%60;
        }
        else{ // play all note.
                    i = (i + 1)%16;
                    PlaySound(i);
                    HAL_Delay(QUANTUM);
        }
  }
  /* USER CODE END 3 */

}
```

3.3 Mode 1 function : check note from UART and play that note.

```
void DemoMode(){
      // Receiving note
      uint8_t receive;
      if (HAL_UART_Receive(&huart2, &receive, 1, 1000) == HAL_OK){
            HAL_UART_Transmit(&huart2, &receive, 1, 1000);
            int8_t notee = -1;
              switch (receive) {
                        case 'z' : notee = 0;break;
                        case 'x' : notee = 1;break;
                        case 'c' : notee = 2;break;
                        case 'v' : notee = 3;break;
                        case 'b' : notee = 4;break;
                        case 'n' : notee = 5;break;
                        case 'm' : notee = 6;break;
                        case 'a' : notee = 7;break;
```

```
                        case 's' : notee = 8;break;
                        case 'd' : notee = 9;break;
                        case 'f' : notee = 10;break;
                        case 'g' : notee = 11;break;
                        case 'h' : notee = 12;break;
                        case 'j' : notee = 13;break;
                        case 'k' : notee = 14;break;
                        case 'l' : notee = 15;break;
                        default : notee = -1; break;
                }
                PlaySound(notee);
        }
}
```

3.4 Play sound function: (0-15 : range of note)

```
void PlaySound(int r){

        if(r < 0 || r > 16) {
                HAL_Delay(25);
                return;
        }
```

3.4.1 Turn off the beep generator.

```
        // Off
        initV[0] = 0x1E;
        initV[1] = 0x20;
        HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
```

3.4.2 Change the beep frequency.

```
        // Change note
        initV[0] = 0x1C;
        initV[1] = note[r];
        HAL_I2C_Master_Transmit (&hi2c1, 0x94, initV, 2, 50);
```

3.4.3 Turn on the beep generator.

```
        // On
        initV[0] = 0x1E;
        initV[1] = 0xE0;
        HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
```

3.4.4 Generate that beep sound.

```
        int i;
        for(i=0;i<100;i++) HAL_I2S_Transmit (&hi2s3, Istr , 100, 10);
}
```

3.5 Initialize sound function (following the guide in data sheet)

```
void InitSound(){

          Istr[0] = 0;
          //not sure if this is needed but i just put it here
          if( HAL_I2S_Transmit (&hi2s3, Istr , 0x10, 10 ) != HAL_OK){
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
          }
          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 0);  //Reset is set down
          //confirmation LED
          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
          HAL_Delay(500);
```

```
	HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 0);
	//Initialization sequence for CS43L22:
	HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 1);
	//Reset is set Up (Power CS43L22)
	HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
	HAL_Delay(500);
	HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 0);
	initV[0] = 0x00;
	initV[1] = 0x99;
	//check if transfer failed. If so: turn on Red LED
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
	}
	initV[0] = 0x47;
	initV[1] = 0x80;
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
	}
	initV[0] = 0x32;
	initV[1] = 0x80; // 0xBB or 0x80
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 1);
	}
	initV[0] = 0x32;
	initV[1] = 0x00; // 0x3B or 0x00
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
	}
	initV[0] = 0x00;
	initV[1] = 0x00;
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
	}
	initV[0] = 0x1E;
	initV[1] = 0xC0;
	if(HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50) != HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 1);
	}
	initV[0] = 0x02;
	initV[1] = 0x9E;
	if( HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50)!= HAL_OK){
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
	}
}
```

4. Other configuration

```
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 50;
  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 busses 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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2S;
  PeriphClkInitStruct.PLLI2S.PLLI2SN = 50;
  PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
```

```
    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* I2C1 init function */
static void MX_I2C1_Init(void)
{

  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();
  }

}

/* I2S3 init function */
static void MX_I2S3_Init(void)
{

  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_16K;
  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();
  }

}

/* TIM1 init function */
static void MX_TIM1_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim1.Instance = TIM1;
```

```
  htim1.Init.Prescaler = 1680;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 99;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }

}

/* USART2 init function */
static void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
     PC3   ------> I2S2_SD
     PA5   ------> SPI1_SCK
     PA6   ------> SPI1_MISO
     PA7   ------> SPI1_MOSI
     PB10   ------> I2S2_CK
     PA9   ------> USB_OTG_FS_VBUS
```

```
     PA10   ------> USB_OTG_FS_ID
     PA11   ------> USB_OTG_FS_DM
     PA12   ------> USB_OTG_FS_DP
*/
static void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* 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 : PDM_OUT_Pin */
  GPIO_InitStruct.Pin = PDM_OUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PA5 PA6 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
  HAL_GPIO_Init(GPIOA, &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 pin : CLK_IN_Pin */
  GPIO_InitStruct.Pin = CLK_IN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(CLK_IN_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_INT1_Pin */
  GPIO_InitStruct.Pin = MEMS_INT1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(MEMS_INT1_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);

}
```

```c
/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler */
  /* User can add his own implementation to report the HAL error return state
*/
  while(1)
  {
  }
  /* USER CODE END Error_Handler */
}

#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

/**
  * @}
  */

/**
  * @}
*/

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