Lab 2

Tues 14:15

Jack Landers Henry Fang

## Introduction

In this Lab, we learn how to use STM32cubeMX to speed up the process of development. We created a project to practice programming in C. We created a program that flashes onboard LEDs. And we practiced setup and optimizing the hardware with cubeMX. Then we practice duplicating projects in different ways. First, we duplicated the project by saving it as a new project under a different name. Then generate Keil files with the STM32cubeMX. Last we copied the original files from the old project folder to the new project folder. Another way is directly copy the STM32cubeMX file to a new directory, and rename the project. Then generate the code again. Then we got a clean project with identical pin setups.

## Procedure

Describe what you did during the lab. The way you wired up the board, what code you wrote (don’t paste your actual code here), etc…

### Step 1

Make a new project with STM32cubeMX. We created the program to flash onboard LED, with HAL functions.

### Step 2

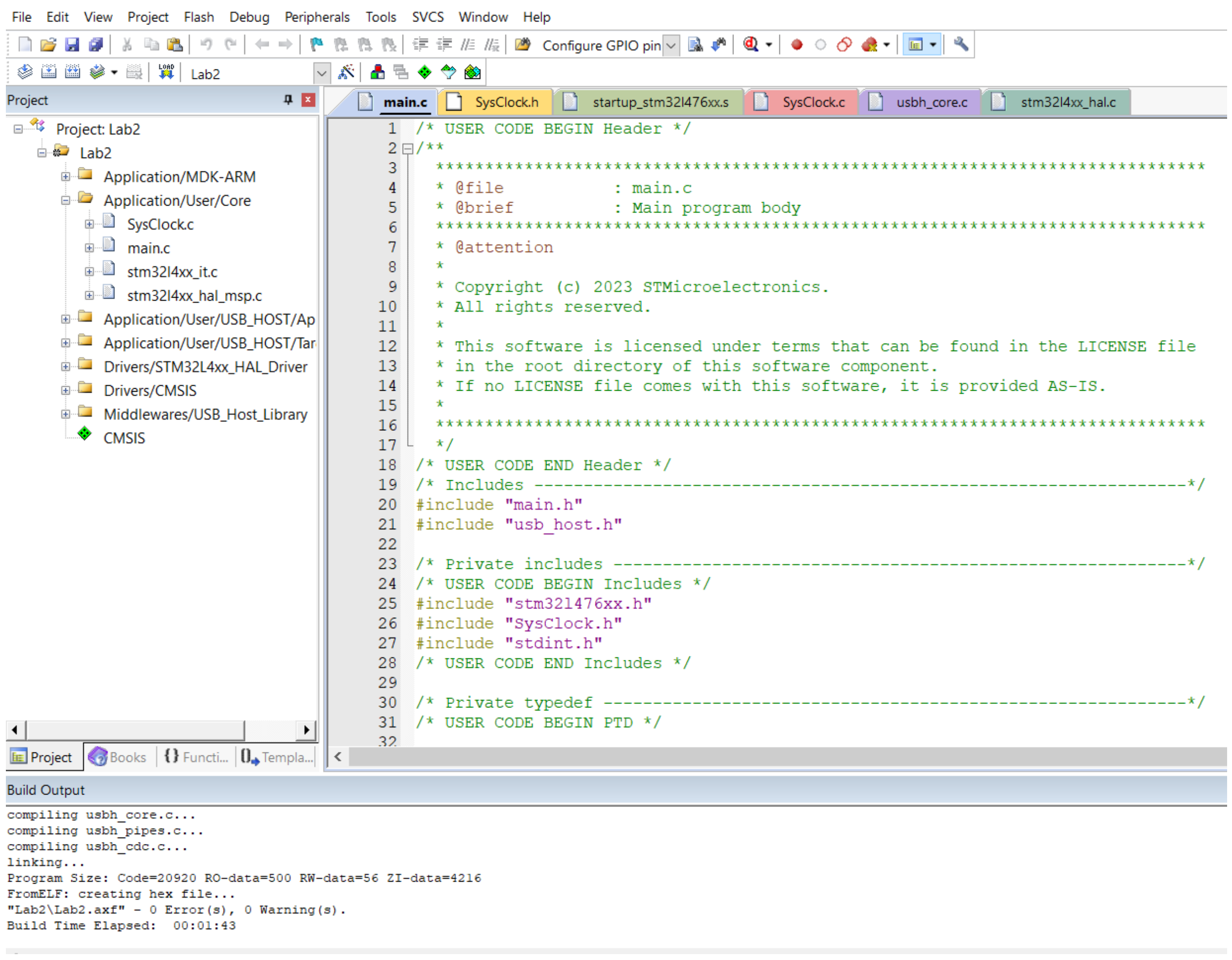
We make a copy of the first project by save as the second project. Then we optimize the hardware by turnoff unnecessary peripherals. We regenerate the code. The configuration is overwritten with the new configuration, but our user codes are kept in a place that “belongs” to users.

### Step 3

We make copies of the entire project by using the system file managers. Rename the new project. Then generate the Keil file with cubeMX. Then we get the clean project with an identical pin configuration.

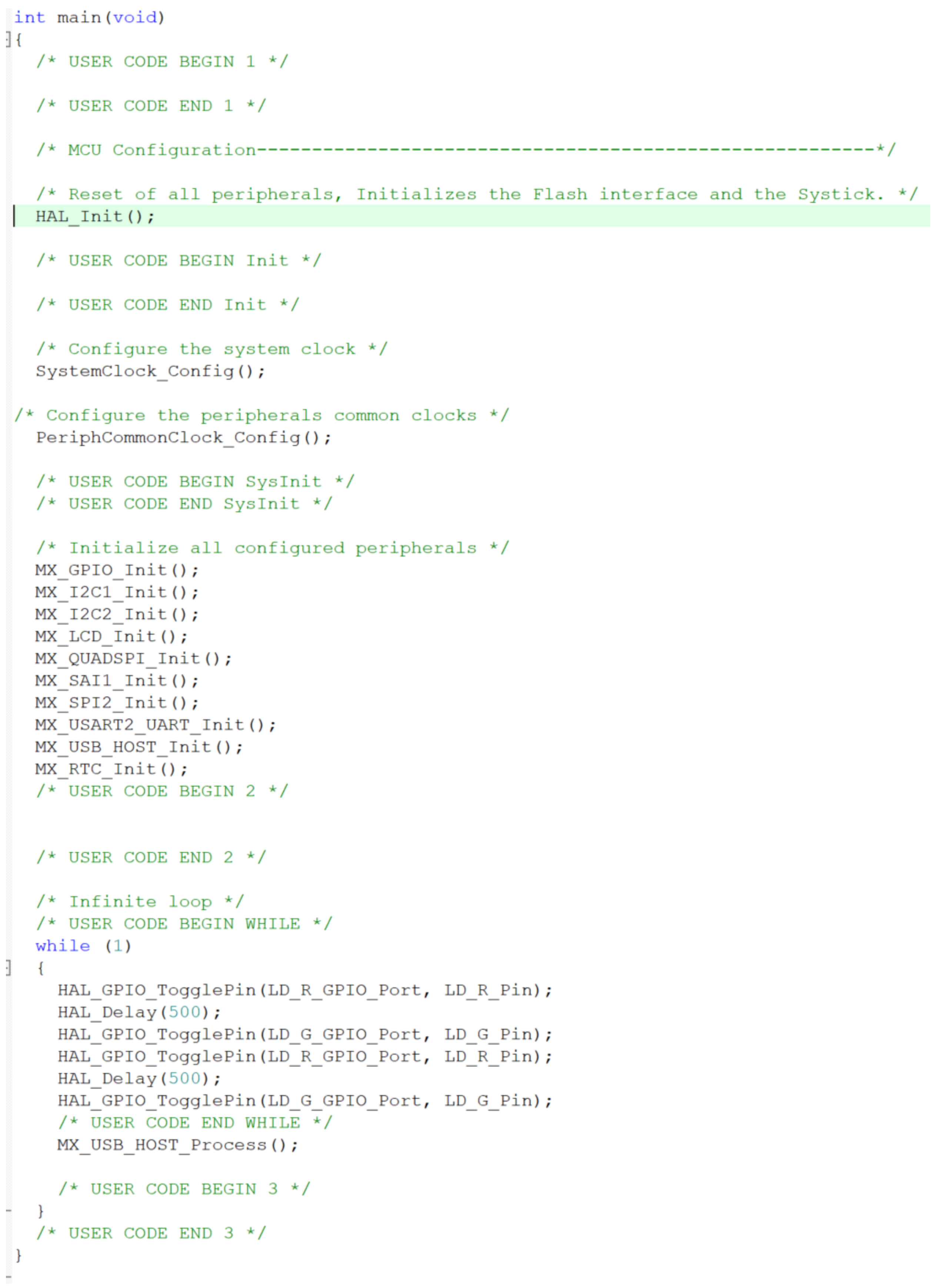
## Results

### Step 1


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Configure GPIO pin
Lab2
Project
Project: Lab2
Lab2
Application/MDK-ARM
Application/User/Core
SysClock.c
main.c
stm32l4xx_it.c
stm32l4xx_hal_msp.c
Application/User/USB_HOST/Ap
Application/User/USB_HOST/Tar
Drivers/STM32L4xx_HAL_Driver
Drivers/CMSIS
Middlewares/USB_Host_Library
CMSIS
Project Books Functi... Templa...
main.c SysClock.h startup_stm32l476xx.s SysClock.c usbh_core.c stm32l4xx_hal.c
Build Output

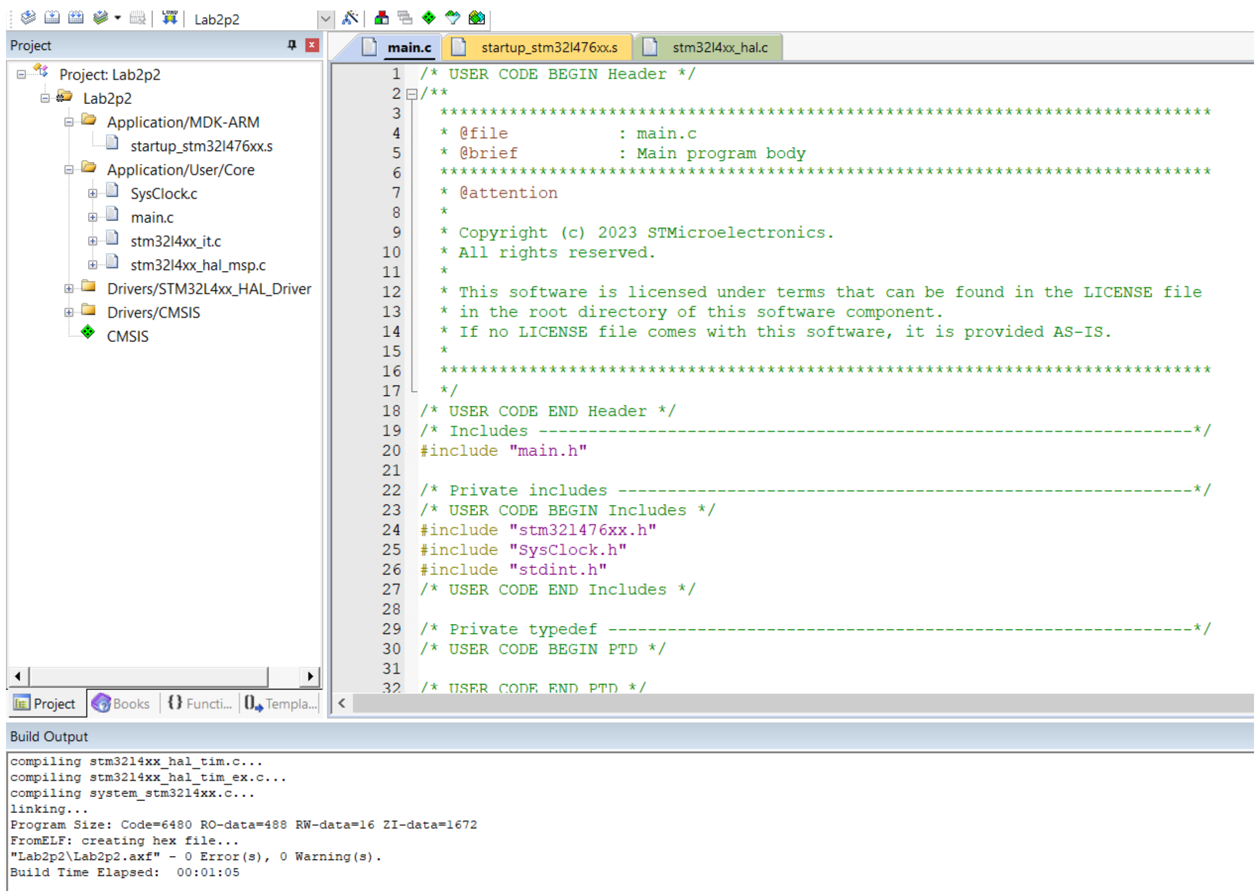
```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32l476xx.h"
#include "SysClock.h"
#include "stdint.h"
/* USER CODE END Includes */

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

```
compiling usbh_core.c...
compiling usbh_pipes.c...
compiling usbh_cdc.c...
linking...
Program Size: Code=20920 RO-data=500 RW-data=56 ZI-data=4216
FromELF: creating hex file...
"Lab2\Lab2.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:43
```



```c
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_I2C2_Init();
  MX_LCD_Init();
  MX_QUADSPI_Init();
  MX_SAI1_Init();
  MX_SPI2_Init();
  MX_USART2_UART_Init();
  MX_USB_HOST_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
    HAL_Delay(500);
    HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
    HAL_Delay(500);
    HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();

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

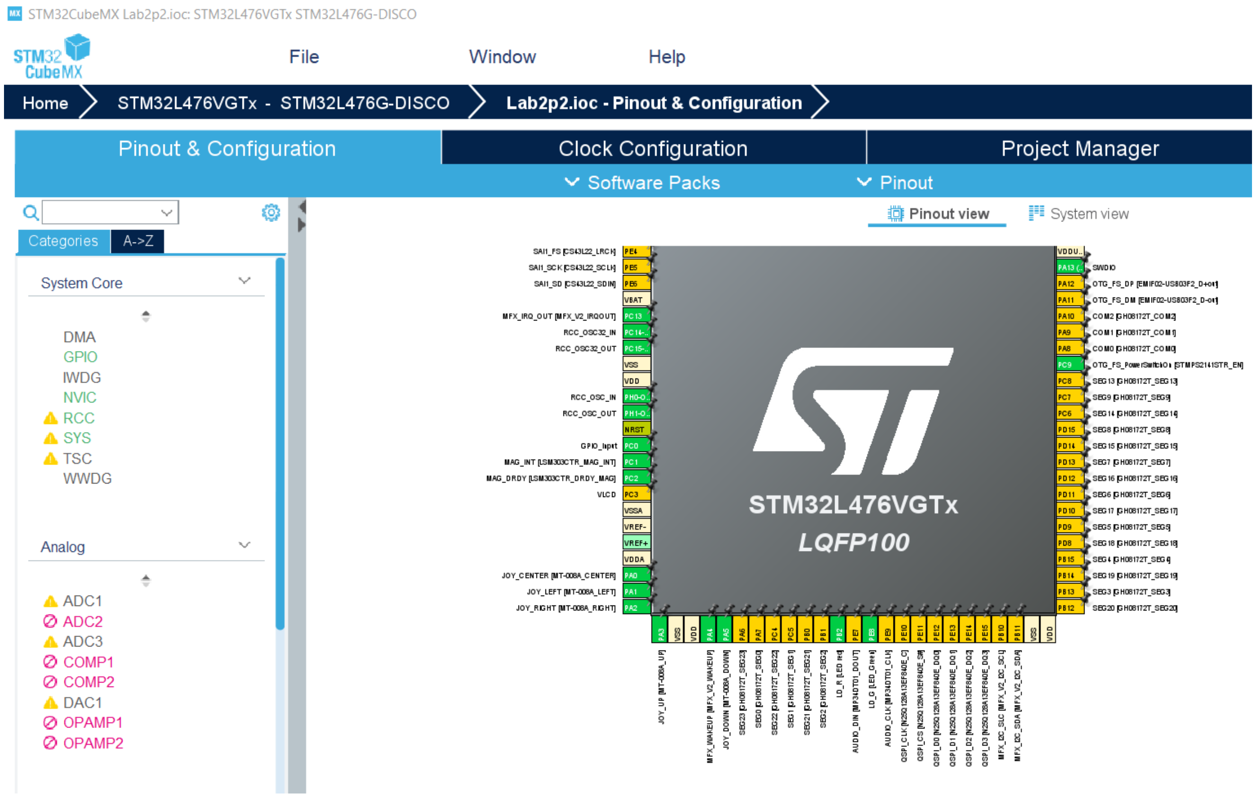
## Step 2



```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32l476xx.h"
#include "SysClock.h"
#include "stdint.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */
```

```
compiling stm32l4xx_hal_tim.c...
compiling stm32l4xx_hal_tim_ex.c...
compiling system_stm32l4xx.c...
linking...
Program Size: Code=6480 RO-data=488 RW-data=16 ZI-data=1672
FromELF: creating hex file...
"Lab2p2\Lab2p2.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:05
```

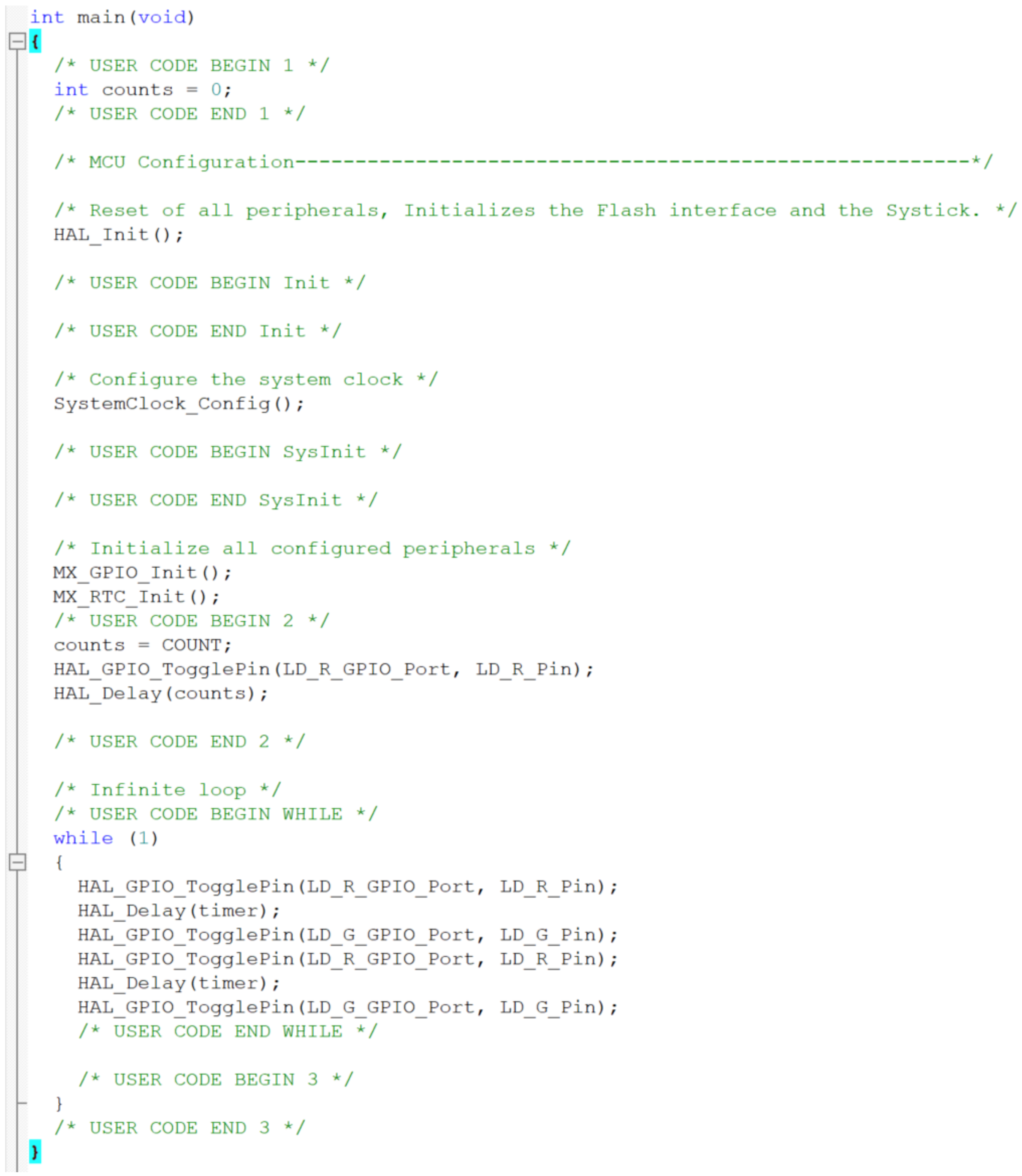
## Step 3



SAI1_FS [CS43L22_LRCK] PE4
SAI1_SCK [CS43L22_SCLK] PE5
SAI1_SD [CS43L22_SDIN] PE6
VBAT
MFX_IRQ_OUT [MFX_V2_IRQOUT] PC13
RCC_OSC32_IN PC14-..
RCC_OSC32_OUT PC15-..
VSS
VDD
RCC_OSC_IN PH0-O..
RCC_OSC_OUT PH1-O..
NRST
GPIO_Input PC0
MAG_INT [LSM303CTR_MAG_INT] PC1
MAG_DRDY [LSM303CTR_DRDY_MAG] PC2
VLCD PC3
VSSA
VREF-
VREF+
VDDA
JOY_CENTER [MT-008A_CENTER] PA0
JOY_LEFT [MT-008A_LEFT] PA1
JOY_RIGHT [MT-008A_RIGHT] PA2

PA3 JOY_UP [MT-008A_UP]
VSS
VDD
PA4 MFX_WAKEUP [MFX_V2_WAKEUP]
PA5 JOY_DOWN [MT-008A_DOWN]
PA6 SEG23 [GH08172T_SEG23]
PA7 SEG0 [GH08172T_SEG0]
PC4 SEG22 [GH08172T_SEG22]
PC5 SEG1 [GH08172T_SEG1]
PB0 SEG21 [GH08172T_SEG21]
PB1 SEG2 [GH08172T_SEG2]
PB2 LD_R [LED red]
PE7 AUDIO_DIN [MP34DT01_DOUT]
PE8 LD_G [LED_Green]
PE9 AUDIO_CLK [MP34DT01_CLK]
PE10 QSPI_CLK [N25Q128A13EF840E_C]
PE11 QSPI_CS [N25Q128A13EF840E_S#]
PE12 QSPI_D0 [N25Q128A13EF840E_DQ0]
PE13 QSPI_D1 [N25Q128A13EF840E_DQ1]
PE14 QSPI_D2 [N25Q128A13EF840E_DQ2]
PE15 QSPI_D3 [N25Q128A13EF840E_DQ3]
PB10 MFX_I2C_SLC [MFX_V2_I2C_SCL]
PB11 MFX_I2C_SDA [MFX_V2_I2C_SDA]
VSS
VDD

VDDU..
PA13 (.. SWDIO
PA12 OTG_FS_DP [EMIF02-USB03F2_D+otg]
PA11 OTG_FS_DM [EMIF02-USB03F2_D-otg]
PA10 COM2 [GH08172T_COM2]
PA9 COM1 [GH08172T_COM1]
PA8 COM0 [GH08172T_COM0]
PC9 OTG_FS_PowerSwitchOn [STMPS2141STR_EN]
PC8 SEG13 [GH08172T_SEG13]
PC7 SEG9 [GH08172T_SEG9]
PC6 SEG14 [GH08172T_SEG14]
PD15 SEG8 [GH08172T_SEG8]
PD14 SEG15 [GH08172T_SEG15]
PD13 SEG7 [GH08172T_SEG7]
PD12 SEG16 [GH08172T_SEG16]
PD11 SEG6 [GH08172T_SEG6]
PD10 SEG17 [GH08172T_SEG17]
PD9 SEG5 [GH08172T_SEG5]
PD8 SEG18 [GH08172T_SEG18]
PB15 SEG4 [GH08172T_SEG4]
PB14 SEG19 [GH08172T_SEG19]
PB13 SEG3 [GH08172T_SEG3]
PB12 SEG20 [GH08172T_SEG20]



```c
int main(void)
{
  /* USER CODE BEGIN 1 */
  int counts = 0;
  /* 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();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */
  counts = COUNT;
  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
  HAL_Delay(counts);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
    HAL_Delay(timer);
    HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
    HAL_Delay(timer);
    HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    /* USER CODE END WHILE */

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

## Conclusion

In this lab, we made 3 projects with cubeMX in a super fast manner. The STM32cubeMX indeed speeds up development. And we don't have to configure the board with manual coding. (most likely) We practiced using the HAL to code. HAL library is also a very useful tool to boost development. No need to manipulate the peripheral registers by hand anymore. :D

# Appendix

## Part 1

```
/* USER CODE BEGIN Header */
/**

**************************************************************************
****
  * @file           : main.c
  * @brief          : Main program body

**************************************************************************
****
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**************************************************************************
****
  */
/* USER CODE END Header */
/* Includes
------------------------------------------------------------------*/
#include "main.h"

/* Private includes
----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32l476xx.h"
#include "SysClock.h"
#include "stdint.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */
/* Private define
------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables
---------------------------------------------------------*/
RTC_HandleTypeDef hrtc;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_RTC_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();

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

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
	  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
	  HAL_Delay(500);
	  HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
	  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
	  HAL_Delay(500);
	  HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    /* 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
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure LSE Drive Capability
  */
  HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_LSE
                              |RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

```c
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Enable MSI Auto calibration
  */
  HAL_RCCEx_EnableMSIPLLMode();
}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */

  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutRemap = RTC_OUTPUT_REMAP_NONE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
```

```
  * @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(GPIOE, AUDIO_RST_Pin|LD_G_Pin|XL_CS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD_R_Pin|M3V3_REG_ON_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(OTG_FS_VBUS_GPIO_Port, OTG_FS_VBUS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GYRO_CS_GPIO_Port, GYRO_CS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : SAI1_MCK_Pin SAI1_FS_Pin SAI1_SCK_Pin
SAI1_SD_Pin
                           AUDIO_DIN_Pin */
  GPIO_InitStruct.Pin = SAI1_MCK_Pin|SAI1_FS_Pin|SAI1_SCK_Pin|SAI1_SD_Pin
                          |AUDIO_DIN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF13_SAI1;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pin : AUDIO_RST_Pin */
  GPIO_InitStruct.Pin = AUDIO_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(AUDIO_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : MFX_IRQ_OUT_Pin OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = MFX_IRQ_OUT_Pin|OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PC0 MAG_INT_Pin MAG_DRDY_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_0|MAG_INT_Pin|MAG_DRDY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : VLCD_Pin SEG22_Pin SEG1_Pin SEG14_Pin
                           SEG9_Pin SEG13_Pin */
  GPIO_InitStruct.Pin = VLCD_Pin|SEG22_Pin|SEG1_Pin|SEG14_Pin
                          |SEG9_Pin|SEG13_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : JOY_CENTER_Pin JOY_LEFT_Pin JOY_RIGHT_Pin
JOY_UP_Pin
                           JOY_DOWN_Pin */
  GPIO_InitStruct.Pin =
JOY_CENTER_Pin|JOY_LEFT_Pin|JOY_RIGHT_Pin|JOY_UP_Pin
                          |JOY_DOWN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

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

  /*Configure GPIO pins : SEG23_Pin SEG0_Pin COM0_Pin COM1_Pin
                           COM2_Pin SEG10_Pin */
  GPIO_InitStruct.Pin = SEG23_Pin|SEG0_Pin|COM0_Pin|COM1_Pin
                          |COM2_Pin|SEG10_Pin;
```

```
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : SEG21_Pin SEG2_Pin SEG20_Pin SEG3_Pin
                         SEG19_Pin SEG4_Pin SEG11_Pin SEG12_Pin
                         COM3_Pin */
GPIO_InitStruct.Pin = SEG21_Pin|SEG2_Pin|SEG20_Pin|SEG3_Pin
                        |SEG19_Pin|SEG4_Pin|SEG11_Pin|SEG12_Pin
                        |COM3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : LD_R_Pin */
GPIO_InitStruct.Pin = LD_R_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(LD_R_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : LD_G_Pin */
GPIO_InitStruct.Pin = LD_G_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(LD_G_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : AUDIO_CLK_Pin */
GPIO_InitStruct.Pin = AUDIO_CLK_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF13_SAI1;
HAL_GPIO_Init(AUDIO_CLK_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : QSPI_CLK_Pin QSPI_CS_Pin QSPI_D0_Pin QSPI_D1_Pin
                         QSPI_D2_Pin QSPI_D3_Pin */
GPIO_InitStruct.Pin = QSPI_CLK_Pin|QSPI_CS_Pin|QSPI_D0_Pin|QSPI_D1_Pin
                        |QSPI_D2_Pin|QSPI_D3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF10_QUADSPI;
HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

/*Configure GPIO pins : MFX_I2C_SLC_Pin MFX_I2C_SDA_Pin */
GPIO_InitStruct.Pin = MFX_I2C_SLC_Pin|MFX_I2C_SDA_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF4_I2C2;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : SEG18_Pin SEG5_Pin SEG17_Pin SEG6_Pin
                         SEG16_Pin SEG7_Pin SEG15_Pin SEG8_Pin */
GPIO_InitStruct.Pin = SEG18_Pin|SEG5_Pin|SEG17_Pin|SEG6_Pin
                        |SEG16_Pin|SEG7_Pin|SEG15_Pin|SEG8_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : OTG_FS_PowerSwitchOn_Pin OTG_FS_VBUS_Pin */
GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin|OTG_FS_VBUS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : OTG_FS_DM_Pin OTG_FS_DP_Pin */
GPIO_InitStruct.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_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : EXT_RST_Pin GYRO_INT1_Pin */
GPIO_InitStruct.Pin = EXT_RST_Pin|GYRO_INT1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : MEMS_SCK_Pin MEMS_MISO_Pin MEMS_MOSI_Pin */
GPIO_InitStruct.Pin = MEMS_SCK_Pin|MEMS_MISO_Pin|MEMS_MOSI_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : USART_TX_Pin USART_RX_Pin */
GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pin : GYRO_CS_Pin */
GPIO_InitStruct.Pin = GYRO_CS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(GYRO_CS_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : M3V3_REG_ON_Pin */
GPIO_InitStruct.Pin = M3V3_REG_ON_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(M3V3_REG_ON_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : I2C1_SCL_Pin I2C1_SDA_Pin */
GPIO_InitStruct.Pin = I2C1_SCL_Pin|I2C1_SDA_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : GYRO_INT2_Pin */
GPIO_InitStruct.Pin = GYRO_INT2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GYRO_INT2_GPIO_Port, &GPIO_InitStruct);

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

/*Configure GPIO pin : XL_INT_Pin */
```

```
  GPIO_InitStruct.Pin = XL_INT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(XL_INT_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* 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 */
  __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 */
```

Part 2

```
/* USER CODE BEGIN Header */
/**

*************************************************************************
****
  * @file           : main.c
  * @brief          : Main program body

*************************************************************************
****
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

*************************************************************************
****
  */
/* USER CODE END Header */
/* Includes
------------------------------------------------------------------*/
#include "main.h"

/* Private includes
----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define
------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define COUNT 10000;
/* USER CODE END PD */
/* Private macro
-------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables
---------------------------------------------------------*/
RTC_HandleTypeDef hrtc;

/* USER CODE BEGIN PV */
int timer = 300;
/* USER CODE END PV */

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_RTC_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 */
      int counts = 0;
  /* 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();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */
      counts = COUNT;
      HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
      HAL_Delay(counts);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
            HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
            HAL_Delay(timer);
            HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
            HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
            HAL_Delay(timer);
            HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    /* 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
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure LSE Drive Capability
  */
  HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_LSE
                              |RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Enable MSI Auto calibration
  */
  HAL_RCCEx_EnableMSIPLLMode();
}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */

  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutRemap = RTC_OUTPUT_REMAP_NONE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
  * @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(GPIOE, AUDIO_RST_Pin|LD_G_Pin|XL_CS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD_R_Pin|M3V3_REG_ON_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(OTG_FS_VBUS_GPIO_Port, OTG_FS_VBUS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GYRO_CS_GPIO_Port, GYRO_CS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : SAI1_MCK_Pin SAI1_FS_Pin SAI1_SCK_Pin
SAI1_SD_Pin
                           AUDIO_DIN_Pin */
  GPIO_InitStruct.Pin = SAI1_MCK_Pin|SAI1_FS_Pin|SAI1_SCK_Pin|SAI1_SD_Pin
                          |AUDIO_DIN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF13_SAI1;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pin : AUDIO_RST_Pin */
  GPIO_InitStruct.Pin = AUDIO_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(AUDIO_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : MFX_IRQ_OUT_Pin OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = MFX_IRQ_OUT_Pin|OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PC0 MAG_INT_Pin MAG_DRDY_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_0|MAG_INT_Pin|MAG_DRDY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : VLCD_Pin SEG22_Pin SEG1_Pin SEG14_Pin
                           SEG9_Pin SEG13_Pin */
  GPIO_InitStruct.Pin = VLCD_Pin|SEG22_Pin|SEG1_Pin|SEG14_Pin
                          |SEG9_Pin|SEG13_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : JOY_CENTER_Pin JOY_LEFT_Pin JOY_RIGHT_Pin
JOY_UP_Pin
                           JOY_DOWN_Pin */
  GPIO_InitStruct.Pin =
JOY_CENTER_Pin|JOY_LEFT_Pin|JOY_RIGHT_Pin|JOY_UP_Pin
                          |JOY_DOWN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

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

  /*Configure GPIO pins : SEG23_Pin SEG0_Pin COM0_Pin COM1_Pin
                           COM2_Pin SEG10_Pin */
  GPIO_InitStruct.Pin = SEG23_Pin|SEG0_Pin|COM0_Pin|COM1_Pin
                          |COM2_Pin|SEG10_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : SEG21_Pin SEG2_Pin SEG20_Pin SEG3_Pin
                           SEG19_Pin SEG4_Pin SEG11_Pin SEG12_Pin
                           COM3_Pin */
  GPIO_InitStruct.Pin = SEG21_Pin|SEG2_Pin|SEG20_Pin|SEG3_Pin
                          |SEG19_Pin|SEG4_Pin|SEG11_Pin|SEG12_Pin
                          |COM3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : LD_R_Pin */
  GPIO_InitStruct.Pin = LD_R_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LD_R_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD_G_Pin */
  GPIO_InitStruct.Pin = LD_G_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LD_G_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : AUDIO_CLK_Pin */
  GPIO_InitStruct.Pin = AUDIO_CLK_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF13_SAI1;
  HAL_GPIO_Init(AUDIO_CLK_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : QSPI_CLK_Pin QSPI_CS_Pin QSPI_D0_Pin QSPI_D1_Pin
                           QSPI_D2_Pin QSPI_D3_Pin */
  GPIO_InitStruct.Pin = QSPI_CLK_Pin|QSPI_CS_Pin|QSPI_D0_Pin|QSPI_D1_Pin
                          |QSPI_D2_Pin|QSPI_D3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF10_QUADSPI;
```

```
HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

/*Configure GPIO pins : MFX_I2C_SLC_Pin MFX_I2C_SDA_Pin */
GPIO_InitStruct.Pin = MFX_I2C_SLC_Pin|MFX_I2C_SDA_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF4_I2C2;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : SEG18_Pin SEG5_Pin SEG17_Pin SEG6_Pin
                         SEG16_Pin SEG7_Pin SEG15_Pin SEG8_Pin */
GPIO_InitStruct.Pin = SEG18_Pin|SEG5_Pin|SEG17_Pin|SEG6_Pin
                        |SEG16_Pin|SEG7_Pin|SEG15_Pin|SEG8_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF11_LCD;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : OTG_FS_PowerSwitchOn_Pin OTG_FS_VBUS_Pin */
GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin|OTG_FS_VBUS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : OTG_FS_DM_Pin OTG_FS_DP_Pin */
GPIO_InitStruct.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_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : EXT_RST_Pin GYRO_INT1_Pin */
GPIO_InitStruct.Pin = EXT_RST_Pin|GYRO_INT1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : MEMS_SCK_Pin MEMS_MISO_Pin MEMS_MOSI_Pin */
GPIO_InitStruct.Pin = MEMS_SCK_Pin|MEMS_MISO_Pin|MEMS_MOSI_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
```

```
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : USART_TX_Pin USART_RX_Pin */
GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pin : GYRO_CS_Pin */
GPIO_InitStruct.Pin = GYRO_CS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
HAL_GPIO_Init(GYRO_CS_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : M3V3_REG_ON_Pin */
GPIO_InitStruct.Pin = M3V3_REG_ON_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(M3V3_REG_ON_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : I2C1_SCL_Pin I2C1_SDA_Pin */
GPIO_InitStruct.Pin = I2C1_SCL_Pin|I2C1_SDA_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : GYRO_INT2_Pin */
GPIO_InitStruct.Pin = GYRO_INT2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GYRO_INT2_GPIO_Port, &GPIO_InitStruct);

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

/*Configure GPIO pin : XL_INT_Pin */
GPIO_InitStruct.Pin = XL_INT_Pin;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(XL_INT_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* 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 */
  __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 */
```

Part 3

```
/* USER CODE BEGIN Header */
```

```
/**

************************************************************************
****
  * @file           : main.c
  * @brief          : Main program body

************************************************************************
****
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE
file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

************************************************************************
****
  */
/* USER CODE END Header */
/* Includes
------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"

/* Private includes
----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stm32l476xx.h"
#include "SysClock.h"
#include "stdint.h"
/* USER CODE END Includes */

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

/* USER CODE END PTD */

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

/* USER CODE END PM */

/* Private variables
---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
I2C_HandleTypeDef hi2c2;

LCD_HandleTypeDef hlcd;

QSPI_HandleTypeDef hqspi;

RTC_HandleTypeDef hrtc;

SAI_HandleTypeDef hsai_BlockA1;
SAI_HandleTypeDef hsai_BlockB1;

SPI_HandleTypeDef hspi2;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* 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_I2C2_Init(void);
static void MX_LCD_Init(void);
static void MX_QUADSPI_Init(void);
static void MX_SAI1_Init(void);
static void MX_SPI2_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_RTC_Init(void);
void MX_USB_HOST_Process(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_I2C2_Init();
 // MX_LCD_Init();
  MX_QUADSPI_Init();
  MX_SAI1_Init();
  MX_SPI2_Init();
  MX_USART2_UART_Init();
  MX_USB_HOST_Init();
```

```c
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
	  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
	  HAL_Delay(500);
	  HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
	  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
	  HAL_Delay(500);
	  HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();

    /* 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
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure LSE Drive Capability
  */
  HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_LSE
                              |RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Enable MSI Auto calibration
  */
  HAL_RCCEx_EnableMSIPLLMode();
}

/**
  * @brief Peripherals Common Clock Configuration
  * @retval None
  */
```

```
void PeriphCommonClock_Config(void)
{
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the peripherals clock
  */
  PeriphClkInit.PeriphClockSelection =
RCC_PERIPHCLK_SAI1|RCC_PERIPHCLK_USB;
  PeriphClkInit.Sai1ClockSelection = RCC_SAI1CLKSOURCE_PLLSAI1;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLLSAI1;
  PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC_PLLSOURCE_MSI;
  PeriphClkInit.PLLSAI1.PLLSAI1M = 1;
  PeriphClkInit.PLLSAI1.PLLSAI1N = 24;
  PeriphClkInit.PLLSAI1.PLLSAI1P = RCC_PLLP_DIV7;
  PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC_PLLQ_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1R = RCC_PLLR_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1ClockOut =
RCC_PLLSAI1_SAI1CLK|RCC_PLLSAI1_48M2CLK;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @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.Timing = 0x00404C74;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Analogue filter
  */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Digital filter
  */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief I2C2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C2_Init(void)
{

  /* USER CODE BEGIN I2C2_Init 0 */

  /* USER CODE END I2C2_Init 0 */

  /* USER CODE BEGIN I2C2_Init 1 */

  /* USER CODE END I2C2_Init 1 */
  hi2c2.Instance = I2C2;
  hi2c2.Init.Timing = 0x00404C74;
  hi2c2.Init.OwnAddress1 = 0;
  hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c2.Init.OwnAddress2 = 0;
  hi2c2.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
  hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c2) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Analogue filter
  */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c2, I2C_ANALOGFILTER_ENABLE) !=
HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Digital filter
  */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c2, 0) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C2_Init 2 */

  /* USER CODE END I2C2_Init 2 */

}

/**
  * @brief LCD Initialization Function
  * @param None
  * @retval None
  */
static void MX_LCD_Init(void)
{

  /* USER CODE BEGIN LCD_Init 0 */

  /* USER CODE END LCD_Init 0 */

  /* USER CODE BEGIN LCD_Init 1 */

  /* USER CODE END LCD_Init 1 */
  hlcd.Instance = LCD;
  hlcd.Init.Prescaler = LCD_PRESCALER_1;
  hlcd.Init.Divider = LCD_DIVIDER_16;
  hlcd.Init.Duty = LCD_DUTY_1_4;
  hlcd.Init.Bias = LCD_BIAS_1_4;
  hlcd.Init.VoltageSource = LCD_VOLTAGESOURCE_INTERNAL;
  hlcd.Init.Contrast = LCD_CONTRASTLEVEL_0;
  hlcd.Init.DeadTime = LCD_DEADTIME_0;
  hlcd.Init.PulseOnDuration = LCD_PULSEONDURATION_0;
  hlcd.Init.MuxSegment = LCD_MUXSEGMENT_DISABLE;
  hlcd.Init.BlinkMode = LCD_BLINKMODE_OFF;
  hlcd.Init.BlinkFrequency = LCD_BLINKFREQUENCY_DIV8;
  hlcd.Init.HighDrive = LCD_HIGHDRIVE_DISABLE;
  if (HAL_LCD_Init(&hlcd) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN LCD_Init 2 */

  /* USER CODE END LCD_Init 2 */

}

/**
  * @brief QUADSPI Initialization Function
  * @param None
  * @retval None
  */
static void MX_QUADSPI_Init(void)
{

  /* USER CODE BEGIN QUADSPI_Init 0 */

  /* USER CODE END QUADSPI_Init 0 */

  /* USER CODE BEGIN QUADSPI_Init 1 */

  /* USER CODE END QUADSPI_Init 1 */
  /* QUADSPI parameter configuration*/
  hqspi.Instance = QUADSPI;
  hqspi.Init.ClockPrescaler = 1;
  hqspi.Init.FifoThreshold = 4;
  hqspi.Init.SampleShifting = QSPI_SAMPLE_SHIFTING_HALFCYCLE;
  hqspi.Init.FlashSize = 24;
  hqspi.Init.ChipSelectHighTime = QSPI_CS_HIGH_TIME_1_CYCLE;
  hqspi.Init.ClockMode = QSPI_CLOCK_MODE_0;
  if (HAL_QSPI_Init(&hqspi) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN QUADSPI_Init 2 */
  /* USER CODE END QUADSPI_Init 2 */

}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */

  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutRemap = RTC_OUTPUT_REMAP_NONE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
  * @brief SAI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SAI1_Init(void)
```

```
{

  /* USER CODE BEGIN SAI1_Init 0 */

  /* USER CODE END SAI1_Init 0 */

  /* USER CODE BEGIN SAI1_Init 1 */

  /* USER CODE END SAI1_Init 1 */
  hsai_BlockA1.Instance = SAI1_Block_A;
  hsai_BlockA1.Init.Protocol = SAI_FREE_PROTOCOL;
  hsai_BlockA1.Init.AudioMode = SAI_MODEMASTER_TX;
  hsai_BlockA1.Init.DataSize = SAI_DATASIZE_8;
  hsai_BlockA1.Init.FirstBit = SAI_FIRSTBIT_MSB;
  hsai_BlockA1.Init.ClockStrobing = SAI_CLOCKSTROBING_FALLINGEDGE;
  hsai_BlockA1.Init.Synchro = SAI_ASYNCHRONOUS;
  hsai_BlockA1.Init.OutputDrive = SAI_OUTPUTDRIVE_DISABLE;
  hsai_BlockA1.Init.NoDivider = SAI_MASTERDIVIDER_ENABLE;
  hsai_BlockA1.Init.FIFOThreshold = SAI_FIFOTHRESHOLD_EMPTY;
  hsai_BlockA1.Init.AudioFrequency = SAI_AUDIO_FREQUENCY_192K;
  hsai_BlockA1.Init.SynchroExt = SAI_SYNCEXT_DISABLE;
  hsai_BlockA1.Init.MonoStereoMode = SAI_STEREOMODE;
  hsai_BlockA1.Init.CompandingMode = SAI_NOCOMPANDING;
  hsai_BlockA1.Init.TriState = SAI_OUTPUT_NOTRELEASED;
  hsai_BlockA1.FrameInit.FrameLength = 8;
  hsai_BlockA1.FrameInit.ActiveFrameLength = 1;
  hsai_BlockA1.FrameInit.FSDefinition = SAI_FS_STARTFRAME;
  hsai_BlockA1.FrameInit.FSPolarity = SAI_FS_ACTIVE_LOW;
  hsai_BlockA1.FrameInit.FSOffset = SAI_FS_FIRSTBIT;
  hsai_BlockA1.SlotInit.FirstBitOffset = 0;
  hsai_BlockA1.SlotInit.SlotSize = SAI_SLOTSIZE_DATASIZE;
  hsai_BlockA1.SlotInit.SlotNumber = 1;
  hsai_BlockA1.SlotInit.SlotActive = 0x00000000;
  if (HAL_SAI_Init(&hsai_BlockA1) != HAL_OK)
  {
    Error_Handler();
  }
  hsai_BlockB1.Instance = SAI1_Block_B;
  hsai_BlockB1.Init.Protocol = SAI_FREE_PROTOCOL;
  hsai_BlockB1.Init.AudioMode = SAI_MODESLAVE_RX;
  hsai_BlockB1.Init.DataSize = SAI_DATASIZE_8;
  hsai_BlockB1.Init.FirstBit = SAI_FIRSTBIT_MSB;
  hsai_BlockB1.Init.ClockStrobing = SAI_CLOCKSTROBING_FALLINGEDGE;
  hsai_BlockB1.Init.Synchro = SAI_SYNCHRONOUS;
  hsai_BlockB1.Init.OutputDrive = SAI_OUTPUTDRIVE_DISABLE;
  hsai_BlockB1.Init.FIFOThreshold = SAI_FIFOTHRESHOLD_EMPTY;
  hsai_BlockB1.Init.SynchroExt = SAI_SYNCEXT_DISABLE;
  hsai_BlockB1.Init.MonoStereoMode = SAI_STEREOMODE;
  hsai_BlockB1.Init.CompandingMode = SAI_NOCOMPANDING;
  hsai_BlockB1.Init.TriState = SAI_OUTPUT_NOTRELEASED;
  hsai_BlockB1.FrameInit.FrameLength = 8;
  hsai_BlockB1.FrameInit.ActiveFrameLength = 1;
  hsai_BlockB1.FrameInit.FSDefinition = SAI_FS_STARTFRAME;
  hsai_BlockB1.FrameInit.FSPolarity = SAI_FS_ACTIVE_LOW;
  hsai_BlockB1.FrameInit.FSOffset = SAI_FS_FIRSTBIT;
  hsai_BlockB1.SlotInit.FirstBitOffset = 0;
  hsai_BlockB1.SlotInit.SlotSize = SAI_SLOTSIZE_DATASIZE;
  hsai_BlockB1.SlotInit.SlotNumber = 1;
  hsai_BlockB1.SlotInit.SlotActive = 0x00000000;
  if (HAL_SAI_Init(&hsai_BlockB1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SAI1_Init 2 */

  /* USER CODE END SAI1_Init 2 */

}

/**
  * @brief SPI2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI2_Init(void)
{

  /* USER CODE BEGIN SPI2_Init 0 */

  /* USER CODE END SPI2_Init 0 */

  /* USER CODE BEGIN SPI2_Init 1 */

  /* USER CODE END SPI2_Init 1 */
  /* SPI2 parameter configuration*/
  hspi2.Instance = SPI2;
  hspi2.Init.Mode = SPI_MODE_MASTER;
  hspi2.Init.Direction = SPI_DIRECTION_2LINES;
  hspi2.Init.DataSize = SPI_DATASIZE_4BIT;
  hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi2.Init.NSS = SPI_NSS_SOFT;
  hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
  hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi2.Init.CRCPolynomial = 7;
  hspi2.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi2.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI2_Init 2 */

  /* USER CODE END SPI2_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  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;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */
```

```
}

/**
  * @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(GPIOE, AUDIO_RST_Pin|LD_G_Pin|XL_CS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD_R_Pin|M3V3_REG_ON_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(OTG_FS_VBUS_GPIO_Port, OTG_FS_VBUS_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GYRO_CS_GPIO_Port, GYRO_CS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : AUDIO_RST_Pin */
  GPIO_InitStruct.Pin = AUDIO_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(AUDIO_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : MFX_IRQ_OUT_Pin OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = MFX_IRQ_OUT_Pin|OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PC0 MAG_INT_Pin MAG_DRDY_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_0|MAG_INT_Pin|MAG_DRDY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : JOY_CENTER_Pin JOY_LEFT_Pin JOY_RIGHT_Pin
JOY_UP_Pin
                           JOY_DOWN_Pin */
  GPIO_InitStruct.Pin =
JOY_CENTER_Pin|JOY_LEFT_Pin|JOY_RIGHT_Pin|JOY_UP_Pin
                          |JOY_DOWN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

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

  /*Configure GPIO pin : LD_R_Pin */
  GPIO_InitStruct.Pin = LD_R_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LD_R_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD_G_Pin */
  GPIO_InitStruct.Pin = LD_G_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(LD_G_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : OTG_FS_PowerSwitchOn_Pin OTG_FS_VBUS_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin|OTG_FS_VBUS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  /*Configure GPIO pins : EXT_RST_Pin GYRO_INT1_Pin */
  GPIO_InitStruct.Pin = EXT_RST_Pin|GYRO_INT1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : GYRO_CS_Pin */
  GPIO_InitStruct.Pin = GYRO_CS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GYRO_CS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : M3V3_REG_ON_Pin */
  GPIO_InitStruct.Pin = M3V3_REG_ON_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(M3V3_REG_ON_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : GYRO_INT2_Pin */
  GPIO_InitStruct.Pin = GYRO_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GYRO_INT2_GPIO_Port, &GPIO_InitStruct);

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

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

}

/* USER CODE BEGIN 4 */

/* 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 */
  __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 */
```