Lab 8 - squadra A15

Antonioli Gianfranco 268382
Leli Lorenzo 281728
Ribaudo Alessandro 283309

1 - Interrupt-based variable frequency square waveform generator

file `main.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"

/* 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 fmax 4500   // Hz
#define fmin 800    // Hz
#define fclk 84e6   // Hz
#define potmax 255

#define DIOR DIER
#define rapace volatile
```

```c
/* USER CODE END PD */

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

/* USER CODE END PM */

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

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
int val = 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. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_1);

  /* System interrupt 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_USART2_UART_Init();
  MX_ADC1_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
    LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | 1); // set ADON to 1
    LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | (1 << 30)); // set
SWSTART to 1

    LL_TIM_WriteReg(TIM3, CCR1, fclk/(2*fmin));  // set initial threshold
    LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~0x2);   // delete OC
flag channel 1
    LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1);  // counter
enable channel 1

    //LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | 0x1); // enable
interrupt // no, it just destroys everything
    LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | 0x2); // enable ch1
of interrupt

  while (1)
  {
      if(LL_ADC_ReadReg(ADC1, SR) & (1 << 1)){ // read EOC bit: if ADC finishes
conversion
        LL_ADC_WriteReg(ADC1, SR, LL_ADC_ReadReg(ADC1, SR) & ~(1 << 1)); // reset
EOC bit
        uint8_t pot = (uint8_t)(LL_ADC_ReadReg(ADC1, DR) & 0xFFFF); // read pot
current value
        float f = fmin + (pot/((float)potmax))*(fmax-fmin);
        val = fclk / (2*f);
        //val = 0x20D0*2;
      }
    /* USER CODE END WHILE */

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

/**
  * @brief System Clock Configuration
```

```
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_Init1msTick(84000000);
  LL_SetSystemCoreClock(84000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  LL_ADC_InitTypeDef ADC_InitStruct = {0};
```

```
  LL_ADC_REG_InitTypeDef ADC_REG_InitStruct = {0};
  LL_ADC_CommonInitTypeDef ADC_CommonInitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_ADC1);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**ADC1 GPIO Configuration
  PA0-WKUP   ------> ADC1_IN0
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_0;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ANALOG;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  ADC_InitStruct.Resolution = LL_ADC_RESOLUTION_8B;
  ADC_InitStruct.DataAlignment = LL_ADC_DATA_ALIGN_RIGHT;
  ADC_InitStruct.SequencersScanMode = LL_ADC_SEQ_SCAN_DISABLE;
  LL_ADC_Init(ADC1, &ADC_InitStruct);
  ADC_REG_InitStruct.TriggerSource = LL_ADC_REG_TRIG_SOFTWARE;
  ADC_REG_InitStruct.SequencerLength = LL_ADC_REG_SEQ_SCAN_DISABLE;
  ADC_REG_InitStruct.SequencerDiscont = LL_ADC_REG_SEQ_DISCONT_DISABLE;
  ADC_REG_InitStruct.ContinuousMode = LL_ADC_REG_CONV_CONTINUOUS;
  ADC_REG_InitStruct.DMATransfer = LL_ADC_REG_DMA_TRANSFER_NONE;
  LL_ADC_REG_Init(ADC1, &ADC_REG_InitStruct);
  LL_ADC_REG_SetFlagEndOfConversion(ADC1, LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV);
  ADC_CommonInitStruct.CommonClock = LL_ADC_CLOCK_SYNC_PCLK_DIV4;
  LL_ADC_CommonInit(__LL_ADC_COMMON_INSTANCE(ADC1), &ADC_CommonInitStruct);
  /** Configure Regular Channel
  */
  LL_ADC_REG_SetSequencerRanks(ADC1, LL_ADC_REG_RANK_1, LL_ADC_CHANNEL_0);
  LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_0,
LL_ADC_SAMPLINGTIME_3CYCLES);
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{
```

```
  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
  LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

  /* TIM3 interrupt Init */
  NVIC_SetPriority(TIM3_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0,
0));
  NVIC_EnableIRQ(TIM3_IRQn);

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  TIM_InitStruct.Prescaler = 0;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 65535;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM3, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM3);
  LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
  TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_FROZEN;
  TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
  TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
  TIM_OC_InitStruct.CompareValue = 0;
  TIM_OC_InitStruct.OCPolarity = LL_TIM_OCPOLARITY_HIGH;
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
  LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
  LL_TIM_DisableMasterSlaveMode(TIM3);
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_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 */

  LL_USART_InitTypeDef USART_InitStruct = {0};
```

```
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* 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)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
```

```
  LL_GPIO_ResetOutputPin(GPIOA, LD2_Pin|LL_GPIO_PIN_10);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);

  /**/
  LL_GPIO_SetPinPull(B1_GPIO_Port, B1_Pin, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(B1_GPIO_Port, B1_Pin, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin|LL_GPIO_PIN_10;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  NVIC_SetPriority(EXTI15_10_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(EXTI15_10_IRQn);

}

/* 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 */
```

file `stm32f4xx_it.c`:

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @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 "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

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

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/

/* USER CODE BEGIN EV */
    extern int val;
/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
```

```
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
```

```
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */

  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
```

```
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles TIM3 global interrupt.
  */
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */
    uint8_t a;
    a = val;
    if(LL_TIM_IsActiveFlag_CC1(TIM3)){
        LL_TIM_ClearFlag_CC1(TIM3);

        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + val);
        LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) ^ (1 << 10));
    }


  /* USER CODE END TIM3_IRQn 0 */
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}

/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_13) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_13);
    /* USER CODE BEGIN LL_EXTI_LINE_13 */

    /* USER CODE END LL_EXTI_LINE_13 */
  }
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

2 - Multiple interrupts

2.1 - Three interrupts

file `main.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"

/* 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 fclk (84e6/1024) // prescaler set to 1023 + 1
#define f1 1000.0
#define f2 500.0
#define f3 250.0

#define DIOR DIER


/* USER CODE END PD */

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

/* USER CODE END PM */
```

```c
/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

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

int val1 = (fclk / (2*f1));
int val2 = (fclk / (2*f2));
int val3 = (fclk / (2*f3));

/* 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. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);

  /* System interrupt 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_USART2_UART_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  LL_TIM_WriteReg(TIM3, CCR1, val1);  // set initial threshold CH1
  LL_TIM_WriteReg(TIM3, CCR2, val2);  // set initial threshold CH2
  LL_TIM_WriteReg(TIM3, CCR3, val3);  // set initial threshold CH3

  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 1));   // delete OC
flag CH1
  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 2));   // delete OC
flag CH2
  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 3));   // delete OC
flag CH3

  LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1);  // counter enable

  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 1)); // enable
interrupt on  CH1
  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 2)); // enable
interrupt on  CH2
  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 3)); // enable
interrupt on  CH3


  while (1)
  {
    /* USER CODE END WHILE */

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

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
```

```
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_Init1msTick(84000000);
  LL_SetSystemCoreClock(84000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
  LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

  /* TIM3 interrupt Init */
```

```
  NVIC_SetPriority(TIM3_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0,
0));
  NVIC_EnableIRQ(TIM3_IRQn);

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  TIM_InitStruct.Prescaler = 1023;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 65535;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM3, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM3);
  LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
  TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_TOGGLE;
  TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
  TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
  TIM_OC_InitStruct.CompareValue = 0;
  TIM_OC_InitStruct.OCPolarity = LL_TIM_OCPOLARITY_HIGH;
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH3, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH3);
  LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
  LL_TIM_DisableMasterSlaveMode(TIM3);
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  /**TIM3 GPIO Configuration
  PC6   ------> TIM3_CH1
  PC7   ------> TIM3_CH2
  PC8   ------> TIM3_CH3
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7|LL_GPIO_PIN_8;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_2;
  LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/**
  * @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 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* 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)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
```

```
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_ResetOutputPin(LD2_GPIO_Port, LD2_Pin);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);

  /**/
  LL_GPIO_SetPinPull(B1_GPIO_Port, B1_Pin, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(B1_GPIO_Port, B1_Pin, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LD2_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 */
```

file `stm32f4xx_it.c`:

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @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 "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

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

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/

/* USER CODE BEGIN EV */

extern int val1;
extern int val2;
extern int val3;

/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
```

```
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
```

```
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */

  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
```

```
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles TIM3 global interrupt.
  */
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */
    if(LL_TIM_IsActiveFlag_CC1(TIM3)){
        LL_TIM_ClearFlag_CC1(TIM3);
        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + val1);
    }
    if(LL_TIM_IsActiveFlag_CC2(TIM3)){
        LL_TIM_ClearFlag_CC2(TIM3);
        LL_TIM_WriteReg(TIM3, CCR2, LL_TIM_ReadReg(TIM3, CCR2) + val2);
    }
    if(LL_TIM_IsActiveFlag_CC3(TIM3)){
        LL_TIM_ClearFlag_CC3(TIM3);
        LL_TIM_WriteReg(TIM3, CCR3, LL_TIM_ReadReg(TIM3, CCR3) + val3);
    }

  /* USER CODE END TIM3_IRQn 0 */
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

2.2 - Four interrupts

file `main.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"

/* 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 fclk (84e6/1024) // prescaler set to 1023 + 1
#define f1 1000.0
#define f2 500.0
#define f3 250.0

#define DIOR DIER

/* USER CODE END PD */

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

/* USER CODE END PM */

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

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

int val1 = (fclk / (2*f1));
int val2 = (fclk / (2*f2));
int val3 = (fclk / (2*f3));

/* 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. */

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);

  /* System interrupt 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_USART2_UART_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  LL_TIM_WriteReg(TIM3, CCR1, val1);  // set initial threshold CH1
  LL_TIM_WriteReg(TIM3, CCR2, val2);  // set initial threshold CH2
```

```
  LL_TIM_WriteReg(TIM3, CCR3, val3);  // set initial threshold CH3

  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 1));   // delete OC
flag CH1
  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 2));   // delete OC
flag CH2
  LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 3));   // delete OC
flag CH3

  LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1);  // counter enable

  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 1)); // enable
interrupt on  CH1
  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 2)); // enable
interrupt on  CH2
  LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 3)); // enable
interrupt on  CH3

  while (1)
  {
    /* USER CODE END WHILE */

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

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {
```

```
  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_Init1msTick(84000000);
  LL_SetSystemCoreClock(84000000);
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
  LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

  /* TIM3 interrupt Init */
  NVIC_SetPriority(TIM3_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0,
0));
  NVIC_EnableIRQ(TIM3_IRQn);

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  TIM_InitStruct.Prescaler = 1023;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 65535;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM3, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM3);
  LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
  TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_TOGGLE;
  TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
  TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
```

```
  TIM_OC_InitStruct.CompareValue = 0;
  TIM_OC_InitStruct.OCPolarity = LL_TIM_OCPOLARITY_HIGH;
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH3, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH3);
  LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
  LL_TIM_DisableMasterSlaveMode(TIM3);
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  /**TIM3 GPIO Configuration
  PC6   ------> TIM3_CH1
  PC7   ------> TIM3_CH2
  PC8   ------> TIM3_CH3
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7|LL_GPIO_PIN_8;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_2;
  LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/**
  * @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 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
```

```
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* 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)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_ResetOutputPin(LD2_GPIO_Port, LD2_Pin);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);
```

```
  /**/
  LL_GPIO_SetPinPull(GPIOC, LL_GPIO_PIN_13, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(GPIOC, LL_GPIO_PIN_13, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  NVIC_SetPriority(EXTI15_10_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(EXTI15_10_IRQn);

}

/* 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 */
```

file `stm32f4xx_it.c`:

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @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 "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

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

/* USER CODE END TD */

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

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */
```

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

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/

/* USER CODE BEGIN EV */

extern int val1;
extern int val2;
extern int val3;

/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
```

```
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
```

```
  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */

  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles TIM3 global interrupt.
  */
```

```
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */

    if(LL_TIM_IsActiveFlag_CC1(TIM3)){
        LL_TIM_ClearFlag_CC1(TIM3);
        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + val1);
    }
    if(LL_TIM_IsActiveFlag_CC2(TIM3)){
        LL_TIM_ClearFlag_CC2(TIM3);
        LL_TIM_WriteReg(TIM3, CCR2, LL_TIM_ReadReg(TIM3, CCR2) + val2);
    }
    if(LL_TIM_IsActiveFlag_CC3(TIM3)){
        LL_TIM_ClearFlag_CC3(TIM3);
        LL_TIM_WriteReg(TIM3, CCR3, LL_TIM_ReadReg(TIM3, CCR3) + val3);
    }

  /* USER CODE END TIM3_IRQn 0 */
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}

/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_13) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_13);
    /* USER CODE BEGIN LL_EXTI_LINE_13 */
        LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) ^ (1 << 5));
    /* USER CODE END LL_EXTI_LINE_13 */
  }
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

2.3 - Five interrupts

file `main.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"

/* 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 fclk (84e6/(2048+1)) // prescaler set to 2048 TODO why if I put 2047 it
does not work as expected?

#define f1max (10.0e3)
#define f1min (1.0e3)
#define f2max (5.0e3)
#define f2min (0.5e3)
#define f3max (2.5e3)
#define f3min (0.25e3)

#define f4 2.0

#define potmax 255

#define DIOR DIER

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PM */

#define CALCULATE_FREQUENCY(fmin, fmax, potx) (fmin + (pot/((float)potmax))*(fmax-
fmin))

/* USER CODE END PM */

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

TIM_HandleTypeDef htim4;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void MX_ADC1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

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

int val = (fclk / (2*f1min));
int oldval;

int val4 = (fclk / f4); // with no 2* because this is not a toggle

/* 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_USART2_UART_Init();
  MX_TIM3_Init();
  MX_TIM4_Init();
  MX_ADC1_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
    oldval = val;

    LL_TIM_WriteReg(TIM3, CCR1, val);  // set initial threshold CH1
    LL_TIM_WriteReg(TIM3, CCR2, val*2);  // set initial threshold CH2
    LL_TIM_WriteReg(TIM3, CCR3, val*4);  // set initial threshold CH3

    LL_TIM_WriteReg(TIM4, CCR2, val4);  // set initial threshold TIM4 CH2

    LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 1));   // delete
OC flag CH1
    LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 2));   // delete
OC flag CH2
    LL_TIM_WriteReg(TIM3, SR, LL_TIM_ReadReg(TIM3, SR) & ~(1 << 3));   // delete
OC flag CH3

    LL_TIM_WriteReg(TIM4, SR, LL_TIM_ReadReg(TIM4, SR) & ~(1 << 2));   // delete
OC flag TIM4 CH2

    LL_TIM_WriteReg(TIM3, CR1, LL_TIM_ReadReg(TIM3, CR1) | 0x1);  // counter
enable
    LL_TIM_WriteReg(TIM4, CR1, LL_TIM_ReadReg(TIM4, CR1) | 0x1);  // counter
enable

    LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 1)); // enable
interrupt on  CH1
    LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 2)); // enable
interrupt on  CH2
    LL_TIM_WriteReg(TIM3, DIOR, LL_TIM_ReadReg(TIM3, DIOR) | (1 << 3)); // enable
interrupt on  CH3

    LL_TIM_WriteReg(TIM4, DIOR, LL_TIM_ReadReg(TIM4, DIOR) | (1 << 2)); // enable
interrupt on  CH2

    LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | 1); // set ADON to 1
```

```
  while (1)
  {
      if(LL_ADC_ReadReg(ADC1, SR) & (1 << 1)){ // read EOC bit: if ADC finishes
conversion
        LL_ADC_WriteReg(ADC1, SR, LL_ADC_ReadReg(ADC1, SR) & ~(1 << 1)); // reset
EOC bit
        uint8_t pot = (uint8_t)(LL_ADC_ReadReg(ADC1, DR) & 0xFFFF); // read pot
current value
        float f = CALCULATE_FREQUENCY(f1min, f1max, pot);
        //float f1 = f;
        val = fclk / (2*f);
        /*
        f = CALCULATE_FREQUENCY(f2min, f2max, pot);
        float f2 = f;
        //val2 = val1*2; //fclk / (2*f);
        f = CALCULATE_FREQUENCY(f3min, f3max, pot);
        val3 = val1*4; //fclk / (2*f);
        //float f3 = f;
        int i = 1;
        */
      }
    /* USER CODE END WHILE */

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

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSI_SetCalibTrimming(16);
  LL_RCC_HSI_Enable();

   /* Wait till HSI is ready */
  while(LL_RCC_HSI_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLM_DIV_16, 336,
LL_RCC_PLLP_DIV_4);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
```

```
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_SetSystemCoreClock(84000000);

   /* Update the time base */
  if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
  {
    Error_Handler();
  }
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  LL_ADC_InitTypeDef ADC_InitStruct = {0};
  LL_ADC_REG_InitTypeDef ADC_REG_InitStruct = {0};
  LL_ADC_CommonInitTypeDef ADC_CommonInitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_ADC1);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**ADC1 GPIO Configuration
  PA0-WKUP   ------> ADC1_IN0
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_0;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ANALOG;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN ADC1_Init 1 */
```

```
  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  ADC_InitStruct.Resolution = LL_ADC_RESOLUTION_8B;
  ADC_InitStruct.DataAlignment = LL_ADC_DATA_ALIGN_RIGHT;
  ADC_InitStruct.SequencersScanMode = LL_ADC_SEQ_SCAN_DISABLE;
  LL_ADC_Init(ADC1, &ADC_InitStruct);
  ADC_REG_InitStruct.TriggerSource = LL_ADC_REG_TRIG_SOFTWARE;
  ADC_REG_InitStruct.SequencerLength = LL_ADC_REG_SEQ_SCAN_DISABLE;
  ADC_REG_InitStruct.SequencerDiscont = LL_ADC_REG_SEQ_DISCONT_DISABLE;
  ADC_REG_InitStruct.ContinuousMode = LL_ADC_REG_CONV_SINGLE;
  ADC_REG_InitStruct.DMATransfer = LL_ADC_REG_DMA_TRANSFER_NONE;
  LL_ADC_REG_Init(ADC1, &ADC_REG_InitStruct);
  LL_ADC_REG_SetFlagEndOfConversion(ADC1, LL_ADC_REG_FLAG_EOC_UNITARY_CONV);
  ADC_CommonInitStruct.CommonClock = LL_ADC_CLOCK_SYNC_PCLK_DIV4;
  LL_ADC_CommonInit(__LL_ADC_COMMON_INSTANCE(ADC1), &ADC_CommonInitStruct);
  /** Configure Regular Channel
  */
  LL_ADC_REG_SetSequencerRanks(ADC1, LL_ADC_REG_RANK_1, LL_ADC_CHANNEL_0);
  LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_0,
LL_ADC_SAMPLINGTIME_3CYCLES);
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
  LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);

  /* TIM3 interrupt Init */
  NVIC_SetPriority(TIM3_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0,
0));
  NVIC_EnableIRQ(TIM3_IRQn);

  /* USER CODE BEGIN TIM3_Init 1 */
```

```
  /* USER CODE END TIM3_Init 1 */
  TIM_InitStruct.Prescaler = 2048;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 65535;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM3, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM3);
  LL_TIM_SetClockSource(TIM3, LL_TIM_CLOCKSOURCE_INTERNAL);
  TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_TOGGLE;
  TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_ENABLE;
  TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
  TIM_OC_InitStruct.CompareValue = 0;
  TIM_OC_InitStruct.OCPolarity = LL_TIM_OCPOLARITY_HIGH;
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH2, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH2);
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH3, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH3);
  LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
  LL_TIM_DisableMasterSlaveMode(TIM3);
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  /**TIM3 GPIO Configuration
  PC6   ------> TIM3_CH1
  PC7   ------> TIM3_CH2
  PC8   ------> TIM3_CH3
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7|LL_GPIO_PIN_8;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_2;
  LL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
```

```
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 2048;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 65535;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_OC_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_TIMING;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_OC_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_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 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 115200;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* 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)
{
  LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
```

```
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_ResetOutputPin(LD2_GPIO_Port, LD2_Pin);

  /**/
  LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTC, LL_SYSCFG_EXTI_LINE13);

  /**/
  EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_13;
  EXTI_InitStruct.LineCommand = ENABLE;
  EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
  EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_FALLING;
  LL_EXTI_Init(&EXTI_InitStruct);

  /**/
  LL_GPIO_SetPinPull(GPIOC, LL_GPIO_PIN_13, LL_GPIO_PULL_NO);

  /**/
  LL_GPIO_SetPinMode(GPIOC, LL_GPIO_PIN_13, LL_GPIO_MODE_INPUT);

  /**/
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  NVIC_SetPriority(EXTI15_10_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(EXTI15_10_IRQn);

}

/* 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 */
```

file `stm32f4xx_it.c`:

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @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 "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

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

```
/* USER CODE END TD */

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

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/
extern TIM_HandleTypeDef htim4;
/* USER CODE BEGIN EV */

extern int val;
extern int oldval;

extern int val4;


/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
```

```
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
```

```
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
```

```
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles TIM3 global interrupt.
  */
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */
    if(LL_TIM_IsActiveFlag_CC3(TIM3)){
        LL_TIM_ClearFlag_CC3(TIM3);
        oldval = val;
        LL_TIM_WriteReg(TIM3, CCR3, LL_TIM_ReadReg(TIM3, CCR3) + oldval*4);
    }
    if(LL_TIM_IsActiveFlag_CC2(TIM3)){
        LL_TIM_ClearFlag_CC2(TIM3);
        LL_TIM_WriteReg(TIM3, CCR2, LL_TIM_ReadReg(TIM3, CCR2) + oldval*2);
    }
    if(LL_TIM_IsActiveFlag_CC1(TIM3)){
        LL_TIM_ClearFlag_CC1(TIM3);
        LL_TIM_WriteReg(TIM3, CCR1, LL_TIM_ReadReg(TIM3, CCR1) + oldval);
    }



  /* USER CODE END TIM3_IRQn 0 */
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}

/**
  * @brief This function handles TIM4 global interrupt.
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */
    if(LL_TIM_IsActiveFlag_CC2(TIM4)){
        LL_TIM_ClearFlag_CC2(TIM4);
        LL_TIM_WriteReg(TIM4, CCR2, LL_TIM_ReadReg(TIM4, CCR2) + val4);
        // start the ADC conversion
        LL_ADC_WriteReg(ADC1, CR2, LL_ADC_ReadReg(ADC1, CR2) | (1 << 30)); // set
```

```
SWSTART to 1
    }

  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_13) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_13);
    /* USER CODE BEGIN LL_EXTI_LINE_13 */
        LL_GPIO_WriteReg(GPIOA, ODR, LL_GPIO_ReadReg(GPIOA, ODR) ^ (1 << 5));
    /* USER CODE END LL_EXTI_LINE_13 */
  }
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```