```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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 "i2c.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define LSM6DS0_ADDRESS 0x6B
#define LSM6DS0_WHO_AM_I 0x0F //Default 0x68
#define LSM6DS0_CTRL_REG1_G 0x10 //Angular rate sensor control register 1
#define LSM6DS0_OUT_TEMP_L 0x15 //Temperature data output low
#define LSM6DS0_OUT_TEMP_H 0x16 //Temperature data output high
#define LSM6DS0_CTRL_REG6_XL 0x20 //Linear acceleration sensor control register 6
#define LSM6DS0_CTRL_REG8 0x22 //Control register 8
#define LSM6DS0_STATUS_REG 0x27 //Status register
#define LSM6DS0_OUT_X_XL 0x28 //OUT_X_XL (28h-29h) linear acceleration sensor X-
axis output register
#define LSM6DS0_OUT_Y_XL 0x2A //OUT_Y_XL (2Ah-2Bh) linear acceleration sensor Y-
axis output register
#define LSM6DS0_OUT_Z_XL 0x2C //OUT_Z_XL (2Ch-2Dh) linear acceleration sensor Z-
axis output register
#define LSM6DS0_FIFO_CTRL 0x2E //FIFO control register
/* USER CODE END PD */

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

/* USER CODE END PM */

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

```c
/* USER CODE BEGIN PV */
uint8_t userButtonFlag = 0;
uint8_t i2cMemRead[4] = {0, 0, 0, 0};
uint8_t i2cMemWrite[4] = {0, 0, 0, 0};
uint8_t lsm6ds0outTemp[2] = {0, 0};
uint8_t lsm6ds0outXxl[2] = {0, 0};
uint8_t lsm6ds0outYxl[2] = {0, 0};
uint8_t lsm6ds0outZxl[2] = {0, 0};
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
int __io_putchar(int ch){
      HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
      return ch;
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin == USER_BUTTON_Pin) userButtonFlag = 1;
}
/* 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_I2C1_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  //Software reset
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG8, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CTRL REG8: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  i2cMemWrite[0] = i2cMemRead[0] | 0x01;
  printf("WRITE TO CTRL REG8: %hx\r\n", i2cMemWrite[0]);
  HAL_I2C_Mem_Write(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG8,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemWrite, 1, HAL_MAX_DELAY);
  do{
        HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_STATUS_REG,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
        printf("STATUS REG: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  }while(i2cMemRead[0] & 0x08);

  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_WHO_AM_I, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("WHO AM I: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG6_XL,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CTRL REG6 XL: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG1_G,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CTRL REG1 G: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_FIFO_CTRL,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("FIFO CTRL: %hx\r\n", (unsigned short int)i2cMemRead[0]);

  //Zapisanie w ODR_XL[2:0] wartosci 0x01, aby wlaczyc akcelerometr i pomiar byl z
czestoscia 10Hz
  i2cMemWrite[0] = 0x20;
  printf("WRITE TO CTRL REG6 XL: %hx\r\n", i2cMemWrite[0]);
  HAL_I2C_Mem_Write(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG6_XL,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemWrite, 1, HAL_MAX_DELAY);
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_CTRL_REG6_XL,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CTRL REG6 XL: %hx\r\n", (unsigned short int)i2cMemRead[0]);

  //Odczyt temperatury
  HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_OUT_TEMP_L|0x80,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)lsm6ds0outTemp, 2, HAL_MAX_DELAY);
  printf("OUT TEMP L: %hx\r\nOUT TEMP H: %hx\r\n", (unsigned short
int)lsm6ds0outTemp[0], (unsigned short int)lsm6ds0outTemp[1]);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      if(userButtonFlag == 1){
            userButtonFlag = 0;
            HAL_GPIO_TogglePin(USER_LED_GPIO_Port, USER_LED_Pin);
            //odczyt przyspieszenia w osi X
            HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_STATUS_REG,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
          printf("STATUS REG: %hx\r\n", (unsigned short int)i2cMemRead[0]);
          if(i2cMemRead[0] & 0x01){
```

```
            HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_OUT_X_XL|0x80,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)lsm6ds0outXxl, 2, HAL_MAX_DELAY);
            printf("OUT X XL L: %hx   OUT X XL H: %hx\r\n", (unsigned short
int)lsm6ds0outXxl[0], (unsigned short int)lsm6ds0outXxl[1]);
            HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_OUT_Y_XL|0x80,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)lsm6ds0outYxl, 2, HAL_MAX_DELAY);
            printf("OUT Y XL L: %hx   OUT Y XL H: %hx\r\n", (unsigned short
int)lsm6ds0outYxl[0], (unsigned short int)lsm6ds0outYxl[1]);
            HAL_I2C_Mem_Read(&hi2c1, LSM6DS0_ADDRESS<<1, LSM6DS0_OUT_Z_XL|0x80,
I2C_MEMADD_SIZE_8BIT, (uint8_t*)lsm6ds0outZxl, 2, HAL_MAX_DELAY);
            printf("OUT Z XL L: %hx   OUT Z XL H: %hx\r\n", (unsigned short
int)lsm6ds0outZxl[0], (unsigned short int)lsm6ds0outZxl[1]);
          }
        }
    /* 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};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  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_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

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

/* USER CODE BEGIN 4 */
```

```c
/* 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 */
```