```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 */
/* 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);
/* 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--------------------------------------------------------*/
```

```c
/* 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();
/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
int pressed = 0;
while (1)
{
	pressed = HAL_GPIO_ReadPin(BUTTON_GPIO_Port, BUTTON_Pin);
	if (pressed) {
		break;
	}
	else {
		// HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, 1);
		// HAL_GPIO_WritePin(SOUND_GPIO_Port, SOUND_Pin, 1);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, 1);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 1);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 1);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, 1);
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, 1);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 1);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, 1);
		HAL_Delay(1000);
		// HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, 0);
		// HAL_GPIO_WritePin(SOUND_GPIO_Port, SOUND_Pin, 0);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, 0);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, 0);
		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, 0);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 0);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, 0);
		HAL_Delay(1000);
	}
} /* USER CODE END 3 */
```

```c
}
/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 /** Configure the main internal regulator output voltage
 */
 __HAL_RCC_PWR_CLK_ENABLE();
 __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
 /** 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();
 }
}
/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
 /* GPIO Ports Clock Enable */
 __HAL_RCC_GPIOH_CLK_ENABLE();
 __HAL_RCC_GPIOC_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
 __HAL_RCC_GPIOB_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|LED_Pin|SOUND_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5
                    |GPIO_PIN_6, GPIO_PIN_RESET);
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, D1_Pin|D4_Pin|D2_Pin|D3_Pin, GPIO_PIN_RESET);
  /*Configure GPIO pins : PC0 PC1 LED_Pin SOUND_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|LED_Pin|SOUND_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 pin : BUTTON_Pin */
  GPIO_InitStruct.Pin = BUTTON_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BUTTON_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pins : PA0 PA1 PA4 PA5
                         PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5
                    |GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /*Configure GPIO pins : D1_Pin D4_Pin D2_Pin D3_Pin */
  GPIO_InitStruct.Pin = D1_Pin|D4_Pin|D2_Pin|D3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* 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 */
}
```

```c
#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 */
```