

# Міністерство освіти і науки України Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського” Факультет інформатики та обчислювальної техніки

# Кафедра інформаційних систем та технологій

Лабораторна робота №2

**Технології програмування на C/Embedded**

*«Lab2 - LEDs, Buttons, Interrupts»*

# Виконав:

студент групи ІА-33

Мартинюк Ю.Р.

# Київ 2025

**Тема:** LEDs, Buttons, Interrupts

**Мета:** вивчити керування кнопками, світлодіодами, розібратися з роботою портів та перериваннями

**Результат виконання роботи:**

1) Посилання на додаток:

<https://github.com/masonabor/C-Embedded/tree/main/lab2/IA-33-MYR-lab2/IA-33-MYR-lab2>  
2) Посилання на відео роботи:

<https://github.com/masonabor/C-Embedded/blob/main/lab2/lab2.mp4>

3) Трейс з логічного аналізатора:

<https://github.com/masonabor/C-Embedded/blob/main/lab2/lab2.sal>

**Хід роботи:**

/\* USER CODE BEGIN Header \*/

/\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @file           : main.c

  \* @brief          : Main program body

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @attention

  \*

  \* Copyright (c) 2025 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 LED\_SWITCH\_DELAY (5000)

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

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

TIM\_HandleTypeDef htim6;

/\* USER CODE BEGIN PV \*/

unsigned int ledN = 0;

/\* USER CODE END PV \*/

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

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM6\_Init(void);

/\* USER CODE BEGIN PFP \*/

void startFlash(uint16\_t GPIO\_Pin, uint16\_t delay);

void flashAllOnce(uint32\_t delay);

void flashParallel(uint16\_t parallel);

void flashNeighbour(uint16\_t neighbour);

/\* 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\_TIM6\_Init();

  HAL\_TIM\_Base\_Start\_IT(&htim6);

  /\* USER CODE BEGIN 2 \*/

  /\* USER CODE END 2 \*/

  /\* Infinite loop \*/

  /\* USER CODE BEGIN WHILE \*/

  while (1)

  {

    /\* USER CODE END WHILE \*/

    switch(ledN) {

          case 0:

            uint16\_t delay = 150;

            startFlash(GPIO\_PIN\_12, delay);

            startFlash(GPIO\_PIN\_13, delay);

            startFlash(GPIO\_PIN\_14, delay);

            startFlash(GPIO\_PIN\_15, delay);

            break;

          case 1:

            flashAllOnce(200);

            break;

          case 2:

            flashParallel(0);

            flashParallel(1);

            break;

          case 3:

            flashNeighbour(0);

            flashNeighbour(1);

        }

    /\* USER CODE BEGIN 3 \*/

  }

  /\* USER CODE END 3 \*/

}

/\*\*

  \* @brief System Clock Configuration

  \* @retval None

  \*/

void SystemClock\_Config(void)

{

  RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

  RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

  /\*\* Configure the main internal regulator output voltage

  \*/

  \_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

  \_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

  /\*\* 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 TIM6 Initialization Function

  \* @param None

  \* @retval None

  \*/

static void MX\_TIM6\_Init(void)

{

  /\* USER CODE BEGIN TIM6\_Init 0 \*/

  /\* USER CODE END TIM6\_Init 0 \*/

  TIM\_MasterConfigTypeDef sMasterConfig = {0};

  /\* USER CODE BEGIN TIM6\_Init 1 \*/

  /\* USER CODE END TIM6\_Init 1 \*/

  htim6.Instance = TIM6;

  htim6.Init.Prescaler = 7999;

  htim6.Init.CounterMode = TIM\_COUNTERMODE\_UP;

  htim6.Init.Period = 2999;

  htim6.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

  if (HAL\_TIM\_Base\_Init(&htim6) != HAL\_OK)

  {

    Error\_Handler();

  }

  sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_UPDATE;

  sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

  if (HAL\_TIMEx\_MasterConfigSynchronization(&htim6, &sMasterConfig) != HAL\_OK)

  {

    Error\_Handler();

  }

  /\* USER CODE BEGIN TIM6\_Init 2 \*/

  /\* USER CODE END TIM6\_Init 2 \*/

}

/\*\*

  \* @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\_GPIOD\_CLK\_ENABLE();

  /\*Configure GPIO pin Output Level \*/

  HAL\_GPIO\_WritePin(GPIOD, LED\_GREEN\_Pin|LED\_ORANGE\_Pin|LED\_RED\_Pin|LED\_BLUE\_Pin, GPIO\_PIN\_RESET);

  /\*Configure GPIO pins : LED\_GREEN\_Pin LED\_ORANGE\_Pin LED\_RED\_Pin LED\_BLUE\_Pin \*/

  GPIO\_InitStruct.Pin = LED\_GREEN\_Pin|LED\_ORANGE\_Pin|LED\_RED\_Pin|LED\_BLUE\_Pin;

  GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

  GPIO\_InitStruct.Pull = GPIO\_NOPULL;

  GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

  HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

  /\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

  /\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

    if (htim->Instance == TIM6)

    {

        ledN++;

        if (ledN > 3)

            ledN = 0;

    }

}

void startFlash(uint16\_t GPIO\_Pin, uint16\_t delay) {

  HAL\_GPIO\_WritePin(GPIOD, GPIO\_Pin, GPIO\_PIN\_SET);

  HAL\_Delay(delay);

  HAL\_GPIO\_WritePin(GPIOD, GPIO\_Pin, GPIO\_PIN\_RESET);

  HAL\_Delay(delay);

}

void flashAllOnce(uint32\_t delay) {

  HAL\_GPIO\_WritePin(GPIOD, LED\_GREEN\_Pin, GPIO\_PIN\_SET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_ORANGE\_Pin, GPIO\_PIN\_SET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_RED\_Pin, GPIO\_PIN\_SET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_BLUE\_Pin, GPIO\_PIN\_SET);

  HAL\_Delay(delay);

  HAL\_GPIO\_WritePin(GPIOD, LED\_GREEN\_Pin, GPIO\_PIN\_RESET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_ORANGE\_Pin, GPIO\_PIN\_RESET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_RED\_Pin, GPIO\_PIN\_RESET);

  HAL\_GPIO\_WritePin(GPIOD, LED\_BLUE\_Pin, GPIO\_PIN\_RESET);

  HAL\_Delay(delay);

}

void flashParallel(uint16\_t parallel) {

  uint16\_t led1;

  uint16\_t led2;

  if (parallel == 0) {

    led1 = LED\_GREEN\_Pin;

    led2 = LED\_RED\_Pin;

  } else if (parallel == 1) {

    led1 = LED\_ORANGE\_Pin;

    led2 = LED\_BLUE\_Pin;

  }

  HAL\_GPIO\_WritePin(GPIOD, led1, GPIO\_PIN\_SET);

  HAL\_GPIO\_WritePin(GPIOD, led2, GPIO\_PIN\_SET);

  HAL\_Delay(200);

  HAL\_GPIO\_WritePin(GPIOD, led1, GPIO\_PIN\_RESET);

  HAL\_GPIO\_WritePin(GPIOD, led2, GPIO\_PIN\_RESET);

  HAL\_Delay(200);

}

void flashNeighbour(uint16\_t neighbour) {

  uint16\_t led1;

  uint16\_t led2;

  if (neighbour == 0) {

    led1 = LED\_GREEN\_Pin;

    led2 = LED\_ORANGE\_Pin;

  } else if (neighbour == 1) {

    led1 = LED\_RED\_Pin;

    led2 = LED\_BLUE\_Pin;

  }

  HAL\_GPIO\_WritePin(GPIOD, led1, GPIO\_PIN\_SET);

  HAL\_GPIO\_WritePin(GPIOD, led2, GPIO\_PIN\_SET);

  HAL\_Delay(200);

  HAL\_GPIO\_WritePin(GPIOD, led1, GPIO\_PIN\_RESET);

  HAL\_GPIO\_WritePin(GPIOD, led2, GPIO\_PIN\_RESET);

  HAL\_Delay(200);

}

/\* 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 \*/

**Висновок:** Під час виконання даної лабораторної роботи я ознайомився з принципом роботи в CubeIDE та створив програму для керування чотирьома світлодіодами з чотирма алгоритмами, які перемикаються кожні 3 секунди, що було реалізовано за допомогою конфігурації таймера та переривань.