Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh

Khoa Cơ khí Chế tạo Máy

Bộ môn Cơ Điện tử

BÁO CÁO DỰ ÁN

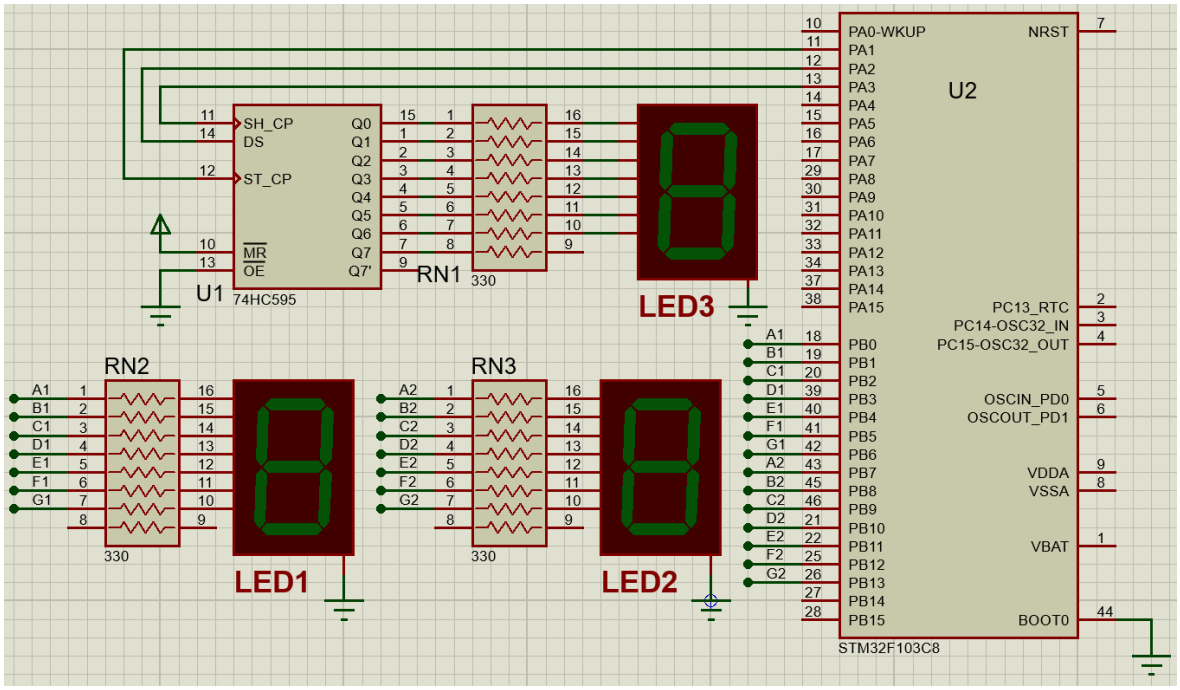
1. Thông tin người thực hiện:

STT	Người thực hiện	MSSV	Ngày
01	Nguyễn Hữu Chí	20146479	25/03/2023

2. Yêu cầu dự án

- Sử dụng kỹ thuật tra bảng mã để điều khiển hiển thị LED 7 đoạn **LED1** sáng lần lượt từ 0 đến 9 (mỗi lần chuyển số trì hoãn 0,5 giây)

- Sử dụng kỹ thuật tra bảng mã và các lệnh vòng lặp để điều khiển hiển thị 2 LED 7 đoạn **LED1** và **LED2** sáng lần lượt từ 00 đến 99 và lặp lại (mỗi lần chuyển số trì hoãn 0,5 giây)



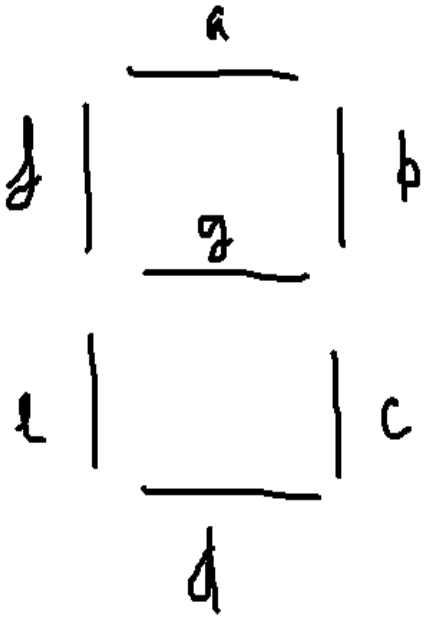
3. Nội dung:

3.1. Phân tích dự án:

- Để sáng được LED thì tín hiệu từ vi điều khiển STM32 phải cấp mức nào? Mức 1

Vì sao? Vì đây là led 7 đoạn có cực âm của các led con nối chung với nhau và nối xuống đất

- Để LED 7 đoạn sáng được số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 thì các chân tín hiệu trên vi điều khiển STM32 phải như thế nào?



LED 1: A1->a ; B1->b ; C1->c ; D1->d ; E1->e ; F1->f ; G1->g

Ví dụ muốn LED1 hiện số 2. Thì trên hình bên các led con a, b, g, e, d sẽ sáng tương ứng chân A1, B1, G1, E1, D1 được kích lên mức 1

Tương tự:

Số 0: A1, B1, C1, D1, E1, F1 được kích lên mức 1

Số 1: B1, C1 được kích lên mức 1

Số 3: A1, B1, G1, C1, D1 được kích lên mức 1

Số 4: F1, G1, B1, C1 được kích lên mức 1

Số 5: A1, F1, G1, C1, D1 được kích lên mức 1

Số 6: A1, F1, G1, E1, D1, C1 được kích lên mức 1

Số 7: A1, B1, C1 được kích lên mức 1

Số 8: A1, B1, C1, D1, E1, F1, G1 được kích lên mức 1

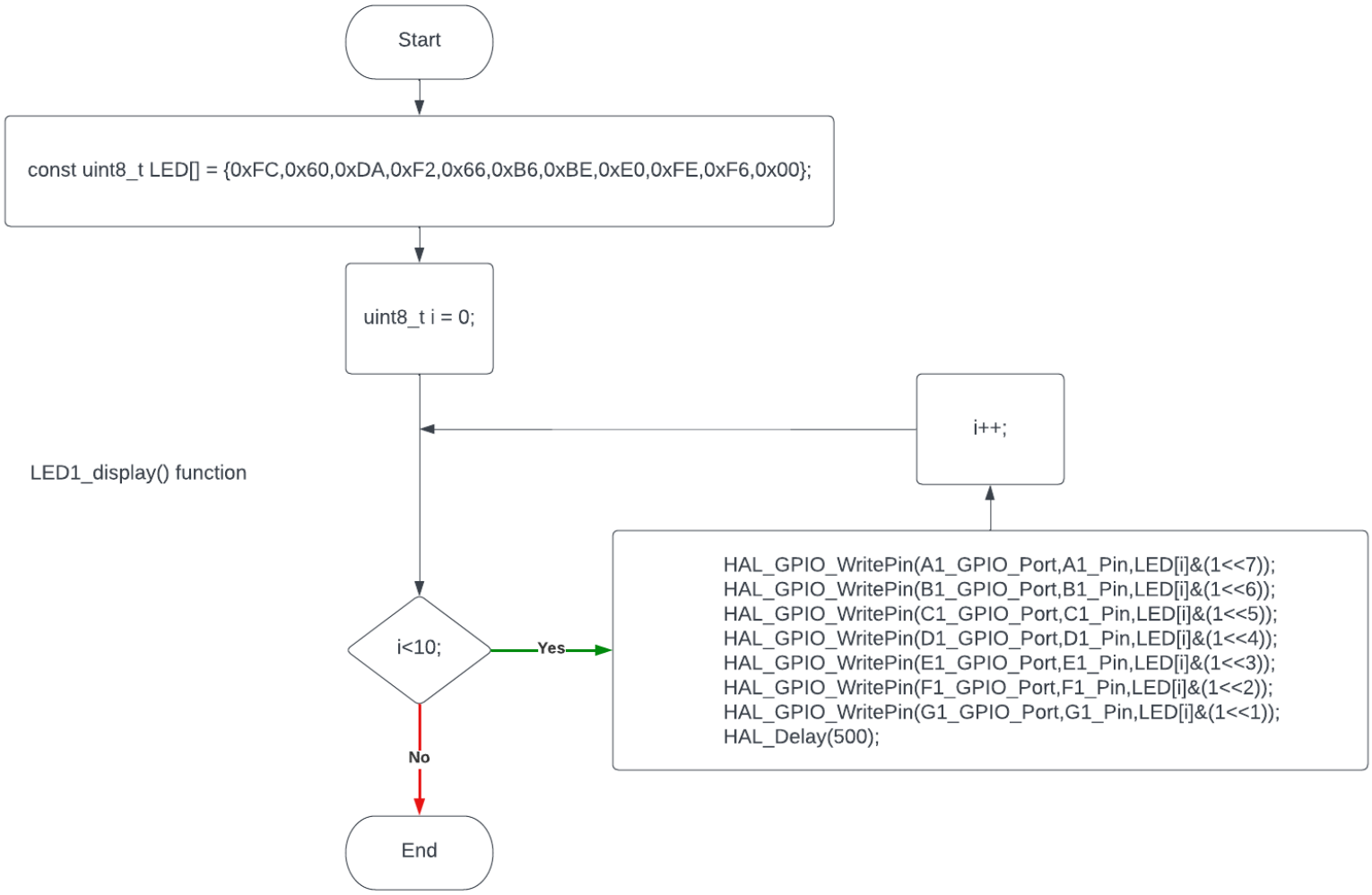
Số 9: A1, F1, G1, B1, C1, D1 được kích lên mức 1

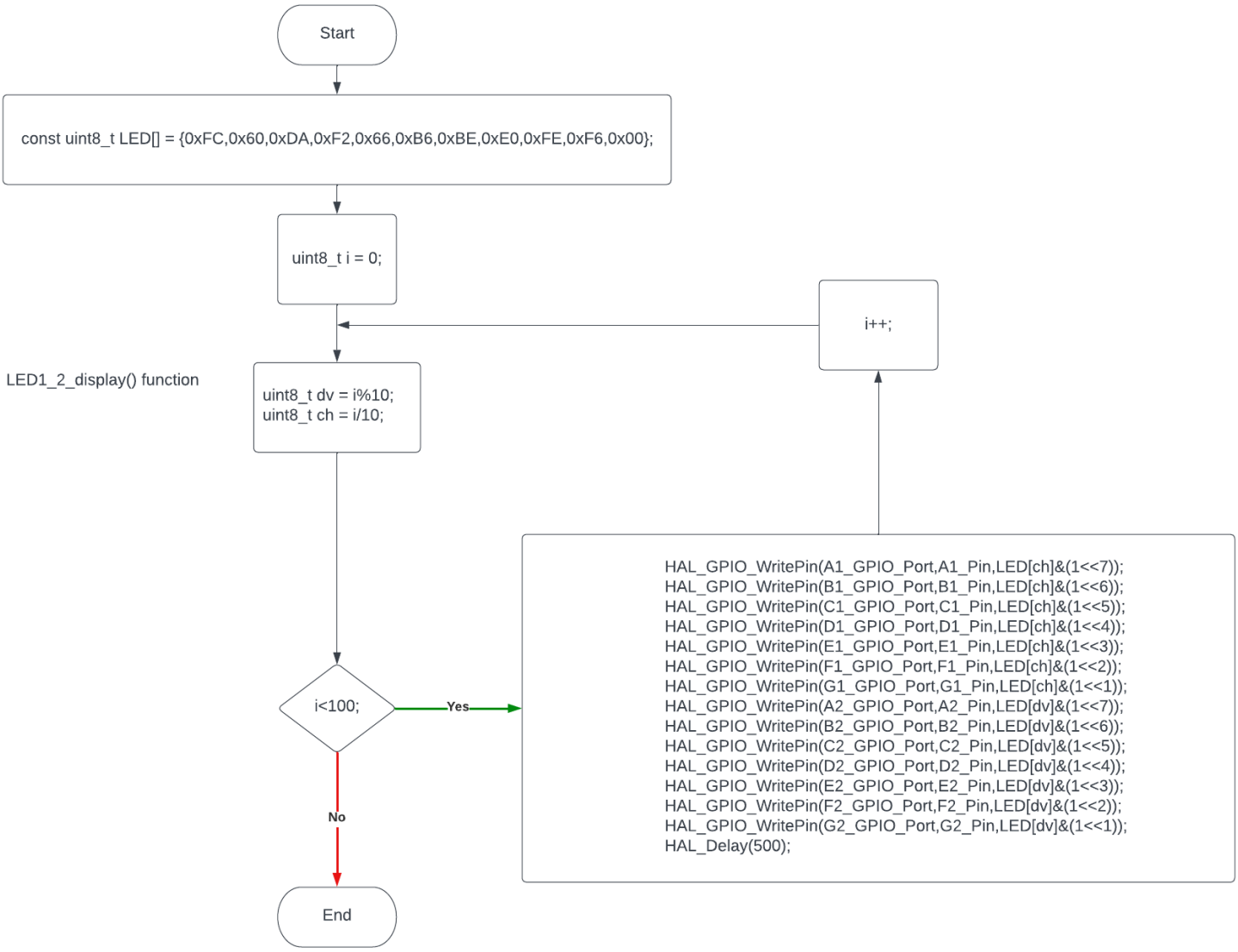
- Khai báo bảng tra và các mã hiển thị từ 0 đến 9 như thế nào?

```
const uint8_t LED[] = {0b11111100,0b01100000,0b11011010,
0b11110010,0b01100110,0b10110110,
0b10111110,0b11100000,0b11111110,0b11110110,0b00000000};
// binary .Chú ý: 0b00000000 tắt led
const uint8_t LED[] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6,0x00};
// hex .Chú ý: 0x00 tắt led
```

- Giải thích thêm các kỹ thuật lập trình tối ưu code nếu có.

3.2. Lưu đồ lập trình:


Start
const uint8_t LED[] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6,0x00};
uint8_t i = 0;
i++;
LED1_display() function
i<10;
Yes
No
HAL_GPIO_WritePin(A1_GPIO_Port,A1_Pin,LED[i]&(1<<7));
HAL_GPIO_WritePin(B1_GPIO_Port,B1_Pin,LED[i]&(1<<6));
HAL_GPIO_WritePin(C1_GPIO_Port,C1_Pin,LED[i]&(1<<5));
HAL_GPIO_WritePin(D1_GPIO_Port,D1_Pin,LED[i]&(1<<4));
HAL_GPIO_WritePin(E1_GPIO_Port,E1_Pin,LED[i]&(1<<3));
HAL_GPIO_WritePin(F1_GPIO_Port,F1_Pin,LED[i]&(1<<2));
HAL_GPIO_WritePin(G1_GPIO_Port,G1_Pin,LED[i]&(1<<1));
HAL_Delay(500);
End


Start
const uint8_t LED[] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6,0x00};
uint8_t i = 0;
i++;
LED1_2_display() function
uint8_t dv = i%10;
uint8_t ch = i/10;
HAL_GPIO_WritePin(A1_GPIO_Port,A1_Pin,LED[ch]&(1<<7));
HAL_GPIO_WritePin(B1_GPIO_Port,B1_Pin,LED[ch]&(1<<6));
HAL_GPIO_WritePin(C1_GPIO_Port,C1_Pin,LED[ch]&(1<<5));
HAL_GPIO_WritePin(D1_GPIO_Port,D1_Pin,LED[ch]&(1<<4));
HAL_GPIO_WritePin(E1_GPIO_Port,E1_Pin,LED[ch]&(1<<3));
HAL_GPIO_WritePin(F1_GPIO_Port,F1_Pin,LED[ch]&(1<<2));
HAL_GPIO_WritePin(G1_GPIO_Port,G1_Pin,LED[ch]&(1<<1));
HAL_GPIO_WritePin(A2_GPIO_Port,A2_Pin,LED[dv]&(1<<7));
HAL_GPIO_WritePin(B2_GPIO_Port,B2_Pin,LED[dv]&(1<<6));
HAL_GPIO_WritePin(C2_GPIO_Port,C2_Pin,LED[dv]&(1<<5));
HAL_GPIO_WritePin(D2_GPIO_Port,D2_Pin,LED[dv]&(1<<4));
HAL_GPIO_WritePin(E2_GPIO_Port,E2_Pin,LED[dv]&(1<<3));
HAL_GPIO_WritePin(F2_GPIO_Port,F2_Pin,LED[dv]&(1<<2));
HAL_GPIO_WritePin(G2_GPIO_Port,G2_Pin,LED[dv]&(1<<1));
HAL_Delay(500);
i<100;
Yes
No
End

3.3. Mã nguồn chương trình:

```
/* 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 */
const uint8_t LED[] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6,0x00};
/* 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 */
void LED1_display(){
	for(uint8_t i = 0; i<10; i++){
			HAL_GPIO_WritePin(A1_GPIO_Port,A1_Pin,LED[i]&(1<<7)); //
0B11110010
			HAL_GPIO_WritePin(B1_GPIO_Port,B1_Pin,LED[i]&(1<<6));
			HAL_GPIO_WritePin(C1_GPIO_Port,C1_Pin,LED[i]&(1<<5));
			HAL_GPIO_WritePin(D1_GPIO_Port,D1_Pin,LED[i]&(1<<4));
```

```
                        HAL_GPIO_WritePin(E1_GPIO_Port,E1_Pin,LED[i]&(1<<3));
                        HAL_GPIO_WritePin(F1_GPIO_Port,F1_Pin,LED[i]&(1<<2));
                        HAL_GPIO_WritePin(G1_GPIO_Port,G1_Pin,LED[i]&(1<<1));
                        HAL_Delay(500);
        }
}

void LED1_2_display(){
        for(uint8_t i = 0; i<100; i++){
                uint8_t dv = i%10;
                uint8_t ch = i/10;
                HAL_GPIO_WritePin(A1_GPIO_Port,A1_Pin,LED[ch]&(1<<7));
                HAL_GPIO_WritePin(B1_GPIO_Port,B1_Pin,LED[ch]&(1<<6));
                HAL_GPIO_WritePin(C1_GPIO_Port,C1_Pin,LED[ch]&(1<<5));
                HAL_GPIO_WritePin(D1_GPIO_Port,D1_Pin,LED[ch]&(1<<4));
                HAL_GPIO_WritePin(E1_GPIO_Port,E1_Pin,LED[ch]&(1<<3));
                HAL_GPIO_WritePin(F1_GPIO_Port,F1_Pin,LED[ch]&(1<<2));
                HAL_GPIO_WritePin(G1_GPIO_Port,G1_Pin,LED[ch]&(1<<1));

                HAL_GPIO_WritePin(A2_GPIO_Port,A2_Pin,LED[dv]&(1<<7));
                HAL_GPIO_WritePin(B2_GPIO_Port,B2_Pin,LED[dv]&(1<<6));
                HAL_GPIO_WritePin(C2_GPIO_Port,C2_Pin,LED[dv]&(1<<5));
                HAL_GPIO_WritePin(D2_GPIO_Port,D2_Pin,LED[dv]&(1<<4));
                HAL_GPIO_WritePin(E2_GPIO_Port,E2_Pin,LED[dv]&(1<<3));
                HAL_GPIO_WritePin(F2_GPIO_Port,F2_Pin,LED[dv]&(1<<2));
                HAL_GPIO_WritePin(G2_GPIO_Port,G2_Pin,LED[dv]&(1<<1));
                HAL_Delay(500);
        }
}
/* 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();
  /* USER CODE BEGIN 2 */
          LED1_display();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
```

```
                LED1_2_display();
    /* 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();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, A1_Pin|B1_Pin|C1_Pin|D2_Pin
                        |E2_Pin|F2_Pin|G2_Pin|D1_Pin
                        |E1_Pin|F1_Pin|G1_Pin|A2_Pin
                        |B2_Pin|C2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : A1_Pin B1_Pin C1_Pin D2_Pin
                         E2_Pin F2_Pin G2_Pin D1_Pin
                         E1_Pin F1_Pin G1_Pin A2_Pin
                         B2_Pin C2_Pin */
  GPIO_InitStruct.Pin = A1_Pin|B1_Pin|C1_Pin|D2_Pin
                        |E2_Pin|F2_Pin|G2_Pin|D1_Pin
                        |E1_Pin|F1_Pin|G1_Pin|A2_Pin
```

```
                      |B2_Pin|C2_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 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 */
```

4. Ghi chú khác (nếu có)

Bit cuối trong mã hiển thị binary không quan trọng do trong chương trình không truy cập vào bit đó nên để 0 hay 1 đều được