ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO THỰC HÀNH

VI XỬ LÝ - VI ĐIỀU KHIỂN

LAB 2: TIMER INTERRUPT AND LED SCANNING

Lớp – Nhóm: L03 – L05
Giảng viên hướng dẫn: Lê Trọng Nhân
Cao Tiến Đạt
Sinh viên thực hiện: Ngô Quang Tùng 2213869

Thành phố Hồ Chí Minh, 10/2024

Mục lục

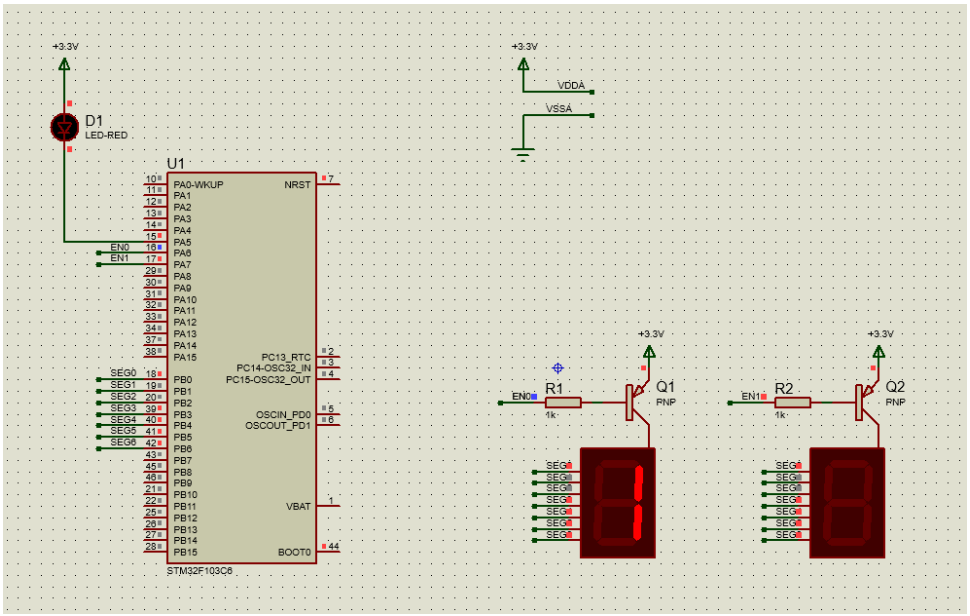
Chương 2

Timer Interrupt and LED Scanning

Link GitHub

2.1 Bài 1

2.1.1 Sơ đồ nguyên lý



Hình 2.1: Sơ đồ nguyên lý

2.1.2 Source code

```
int count = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    static uint8_t display_toggle = 0;
  if(count> 0){
      count--;
  if(count <= 0){
      count = 50;
```

```
8
9       if (htim->Instance == TIM2) {
10
11          switch (display_toggle) {
12              case 0:
13                  HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                        GPIO_PIN_RESET);
14                  HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                        GPIO_PIN_SET);
15                  display7SEG(1);
16                  break;
17
18              case 1:
19                  HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                        GPIO_PIN_SET);
20                  HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                        GPIO_PIN_RESET);
21                  display7SEG(2);
22                  break;
23          }
24          display_toggle = !display_toggle;
25       }
26      }
27    }
28 }
```

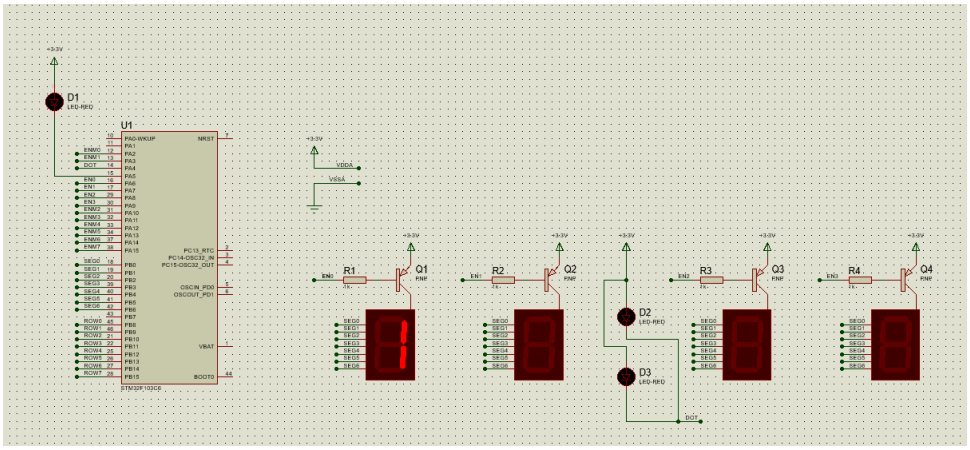
Chương trình 1.1: Hàm HAL_TIM_PeriodElapsedCallback

Xác định giá trị DURATION_1
Thời gian sáng của mỗi LED: $t = 0.5$ **(s) = 500 (ms).**
DURATION_1 = $\frac{t}{T_{\text{InternalClock}}} = \frac{500}{10} = 50$**.**

Chương trình 1.1: Hàm chuyển đổi trạng thái cho 2 đèn LED

2.2 Bài 2

2.2.1 Sơ đồ nguyên lý



Hình 2.2: Sơ đồ nguyên lý

2.2.2 Source code

```
1  int count = 100;
2  int led_blink_count = 0;
3  uint8_t led_state = 0;
4  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
5      static uint8_t display_toggle = 0;
6      if (count > 0) {
7          count--;
8          if (count <= 0) {
9              count = 50;
10             if (htim->Instance == TIM2) {
11                 switch (display_toggle) {
12                     case 0:
13                         HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                               GPIO_PIN_RESET);
14                         HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                               GPIO_PIN_SET);
15                         HAL_GPIO_WritePin(GPIOA, EN2_Pin,
                               GPIO_PIN_SET);
16                         HAL_GPIO_WritePin(GPIOA, EN3_Pin,
                               GPIO_PIN_SET);
17                         display7SEG(1);
18                         break;
19
20                     case 1:
21                         HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                               GPIO_PIN_SET);
```

```
22                          HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                               GPIO_PIN_RESET);
23                          HAL_GPIO_WritePin(GPIOA, EN2_Pin,
                               GPIO_PIN_SET);
24                          HAL_GPIO_WritePin(GPIOA, EN3_Pin,
                               GPIO_PIN_SET);
25                          display7SEG(2);
26                          break;
27
28                      case 2:
29                          HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                               GPIO_PIN_SET);
30                          HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                               GPIO_PIN_SET);
31                          HAL_GPIO_WritePin(GPIOA, EN2_Pin,
                               GPIO_PIN_RESET);
32                          HAL_GPIO_WritePin(GPIOA, EN3_Pin,
                               GPIO_PIN_SET);
33                          display7SEG(3);
34                          break;
35
36                      case 3:
37                          HAL_GPIO_WritePin(GPIOA, EN0_Pin,
                               GPIO_PIN_SET);
38                          HAL_GPIO_WritePin(GPIOA, EN1_Pin,
                               GPIO_PIN_SET);
39                          HAL_GPIO_WritePin(GPIOA, EN2_Pin,
                               GPIO_PIN_SET);
40                          HAL_GPIO_WritePin(GPIOA, EN3_Pin,
                               GPIO_PIN_RESET);
41                          display7SEG(0);
42                          break;
43                  }
44                  display_toggle = (display_toggle + 1) % 4;
45              }
46          }
47      }
48      led_blink_count++;
49      if (led_blink_count >= 100) {
50          led_blink_count = 0;
51
52          led_state = !led_state;
53          HAL_GPIO_WritePin(GPIOA, DOT_Pin, led_state ?
               GPIO_PIN_SET : GPIO_PIN_RESET);
54      }
55  }
```

Chương trình 2.1: Hàm HAL_TIM_PeriodElapsedCallback

Xác định tần số của quá trình quét LED
Chu kỳ của quá trình quét LED là:

$$T = 0.5\,\text{s} + 0.5\,\text{s} = 1\,\text{s}.$$

Tần số của quá trình quét LED được tính bằng:

$$f = \frac{1}{T} = \frac{1}{1} = 1\,\text{Hz}.$$

2.3 Bài 3

2.3.1 Source code

```
const int MAX_LED = 4;
int led_buffer[4] = {1, 2, 3, 4};
void update7SEG(int index) {
    switch (index) {
        case 0:
            HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_RESET)
               ;
            HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_SET);
            display7SEG(led_buffer[0]);
            break;

        case 1:
            HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_RESET)
               ;
            HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_SET);
            display7SEG(led_buffer[1]);
            break;

        case 2:
            HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_RESET)
               ;
            HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_SET);
            display7SEG(led_buffer[2]);
            break;

        case 3:
            HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_SET);
```

```
33              HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_RESET)
                    ;
34              display7SEG(led_buffer[3]);
35              break;
36
37          default:
38              break;
39      }
40  }
```

Chương trình 3.1: Hàm update7SEG

2.3.2 Source code

```
1  int index_led = 0;
2  int count = 50;
3  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
4      if (htim->Instance == TIM2) {
5          if (count > 0) {
6              count--;
7          }
8          if (count <= 0) {
9              count = 50;
10             update7SEG(index_led);
11             index_led++;
12             if (index_led >= MAX_LED) {
13                 index_led = 0;
14             }
15         }
16     }
17 }
```

Chương trình 3.2: Hàm HAL_TIM_PeriodElapsedCallback

2.4 Bài 4

2.4.1 Source code

```
1  int index_led = 0;
2  int count = 25;
3  int dot_count = 100;
4  int dot_state = 0;
5
6  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
7      if (htim->Instance == TIM2) {
8          if (count > 0) {
```

```
9               count--;
10          }
11          if (count <= 0) {
12              count = 25;
13              update7SEG(index_led);
14
15              index_led++;
16              if (index_led >= MAX_LED) {
17                  index_led = 0;
18              }
19          }
20          if (dot_count > 0) {
21              dot_count--;
22          }
23          if (dot_count <= 0) {
24              dot_count = 100;
25              dot_state = !dot_state;
26
27              HAL_GPIO_WritePin(GPIOA, DOT_Pin, (dot_state) ?
                    GPIO_PIN_SET : GPIO_PIN_RESET);
28          }
29      }
30  }
```

Chương trình 4.1: Hàm HAL_TIM_PeriodElapsedCallback

2.5 Bài 5

2.5.1 Source code

```
1  int led_buffer[MAX_LED];
2  int led_buffer[4] = {1, 2, 3, 4};
3  void updateClockBuffer(int hour, int minute) {
4
5      led_buffer[0] = hour / 10;
6      led_buffer[1] = hour % 10;
7      led_buffer[2] = minute / 10;
8      led_buffer[3] = minute % 10;
9  }
```

Chương trình 5.1: Hàm updateClockBuffer

2.6 Bài 6

2.6.1 1.6.1 Trả lời câu hỏi

Nếu dòng 1 của đoạn code mẫu trên bị thiếu, điều gì sẽ xảy ra và tại sao?

Nếu dòng 1 của đoạn code mẫu trên bị thiếu, biến `timer_flag0` sẽ không được set. Khi đó, điều kiện trong vòng lặp `while(1)` sẽ không bao giờ đúng, và `LED_RED` sẽ không được chuyển đổi trạng thái.

2.6.2 1.6.2 Trả lời câu hỏi

Nếu dòng 1 của đoạn code mẫu trên đổi thành `setTimer0(1)`, điều gì sẽ xảy ra và tại sao?

Nếu dòng 1 của đoạn code mẫu trên đổi thành `setTimer0(1)`, biến `timer_flag0` sẽ được set sau 1ms. Khi đó, điều kiện trong vòng lặp `while(1)` sẽ trở thành đúng, `LED_RED` sẽ chuyển đổi trạng thái rất nhanh ở lần đầu tiên. Sau đó, hàm `setTimer0(2000)` sẽ lần lượt được gọi, và `LED_RED` sẽ chuyển đổi trạng thái sau một khoảng thời gian nhất định (2000ms).

2.6.3 1.6.3 Trả lời câu hỏi

Nếu dòng 1 của đoạn code mẫu trên đổi thành `setTimer0(10)`, những thay đổi gì sẽ xảy ra so với 2 câu hỏi trước và tại sao?

Nếu dòng 1 của đoạn code mẫu trên đổi thành `setTimer0(10)`, biến `timer_flag0` sẽ được set sau 10ms. Khi đó, điều kiện trong vòng lặp `while(1)` sẽ trở thành đúng, và `LED_RED` sẽ chuyển đổi trạng thái nhanh hơn bình thường ở lần đầu tiên. Sau đó, hàm `setTimer0(2000)` sẽ lần lượt được gọi, và `LED_RED` sẽ chuyển đổi trạng thái sau một khoảng thời gian nhất định (2000ms).
So với trường hợp 1, `LED_RED` có thể thay đổi trạng thái bình thường. So với trường hợp 2, `LED_RED` ở trường hợp 3 sẽ chuyển đổi trạng thái lần đầu tiên chậm hơn (10ms so với 1ms).

2.7 Bài 7

2.7.1 Source code

```
1   setTimer0(1000);
2   setTimer1(250);
3   setTimer2(1000);
4   int index_led = 0;
5   int dot_state = 0;
6   while (1)
7   {
8     /* USER CODE END WHILE */
9     if(timer0_flag == 1){
```

```
10          setTimer0(1000);
11      second++;
12      if(second >= 60){
13          second = 0;
14          minute++;
15      }
16      if(minute>=60){
17          minute = 0;
18          hour++;
19      }
20      if(hour>= 24){
21          hour = 0;
22      }
23      updateClockBuffer( hour,  minute);
24    }
25      if(timer1_flag == 1) {
26              setTimer1(250);
27              update7SEG(index_led);
28              index_led++;
29              if (index_led >= MAX_LED) {
30                  index_led = 0;
31              }
32          }
33      if(timer2_flag == 1) {
34          setTimer2(1000);
35          dot_state = !dot_state;
36          HAL_GPIO_WritePin(GPIOA, DOT_Pin, (dot_state) ?
               GPIO_PIN_SET : GPIO_PIN_RESET);
37      }
38  }
```

Chương trình 7.1: Hàm While trong main

2.8 Bài 8

2.8.1 Source code

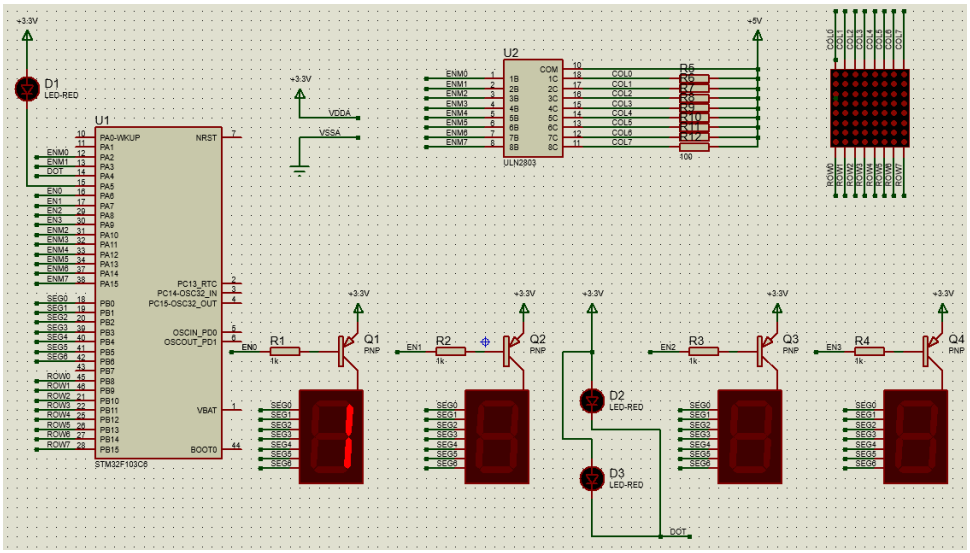
```
1   setTimer0(1000);
2   setTimer1(250);
3   setTimer2(1000);
4   int index_led = 0;
5   int dot_state = 0;
6   while (1)
7   {
8     /* USER CODE END WHILE */
9     if(timer0_flag == 1){
10          setTimer0(1000);
```

```
11      second++;
12      if(second >= 60){
13          second = 0;
14          minute++;
15      }
16      if(minute>=60){
17          minute = 0;
18          hour++;
19      }
20      if(hour>= 24){
21          hour = 0;
22      }
23      updateClockBuffer( hour,  minute);
24   }
25      if(timer1_flag == 1) {
26              setTimer1(250);
27              update7SEG(index_led);
28              index_led++;
29              if (index_led >= MAX_LED) {
30                  index_led = 0;
31              }
32          }
33      if(timer2_flag == 1) {
34          setTimer2(1000);
35
36          dot_state = !dot_state;
37          HAL_GPIO_WritePin(GPIOA, DOT_Pin, (dot_state) ?
               GPIO_PIN_SET : GPIO_PIN_RESET);
38      }
39   }
```

Chương trình 8.1: Hàm setNumberOnClock

2.9 Bài 9

2.9.1 Sơ đồ nguyên lý



Hình 2.3: Sơ đồ nguyên lý

2.9.2 Source code

```
1  void updateLEDMatrix(int index)
2  {
3      switch (index)
4      {
5          case 0:
6              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                   GPIO_PIN_SET);
7              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                   GPIO_PIN_SET);
8              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                   GPIO_PIN_SET);
9              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                   GPIO_PIN_SET);
10             HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                   GPIO_PIN_SET);
11             HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                   GPIO_PIN_SET);
12             HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                   GPIO_PIN_SET);
13             HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                   GPIO_PIN_RESET);
14             displayLEDMtrix(matrix_buffer[index]);
15             break;
16         case 1:
```

```
17              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
18              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
19              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
20              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
21              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
22              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
23              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
24              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_RESET);
25              displayLEDMtrix(matrix_buffer[index]);
26              break;
27          case 2:
28              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
29              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
30              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
31              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
32              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
33              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
34              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
35              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_RESET);
36              displayLEDMtrix(matrix_buffer[index]);
37              break;
38          case 3:
39              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
40              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
41              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
42              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
43              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
```

```
44              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
45              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
46              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_RESET);
47              displayLEDMtrix(matrix_buffer[index]);
48              break;
49          case 4:
50              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
51              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
52              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
53              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
54              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
55              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
56              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
57              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_RESET);
58              displayLEDMtrix(matrix_buffer[index]);
59              break;
60          case 5:
61              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
62              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
63              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
64              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
65              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
66              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
67              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
68              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_RESET);
69              displayLEDMtrix(matrix_buffer[index]);
70              break;
71          case 6:
```

```
72              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
73              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
74              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
75              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
76              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
77              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
78              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_SET);
79              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_RESET);
80              displayLEDMtrix(matrix_buffer[index]);
81              break;
82          case 7:
83              HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin,
                    GPIO_PIN_SET);
84              HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin,
                    GPIO_PIN_SET);
85              HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin,
                    GPIO_PIN_SET);
86              HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin,
                    GPIO_PIN_SET);
87              HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin,
                    GPIO_PIN_SET);
88              HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin,
                    GPIO_PIN_SET);
89              HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin,
                    GPIO_PIN_SET);
90              HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin,
                    GPIO_PIN_RESET);
91              displayLEDMtrix(matrix_buffer[index]);
92              break;
93          default:
94              break;
95      }
96  }
```

Chương trình 9.1: Hàm updateLEDMatrix

2.10 Bài 10

2.10.1 Source code

Trong bài tập này, sử dụng một biến `matrix_offset` để thay đổi giá trị index trong hàm `displayLEDMatrix(char c, int index)`. Sau một khoảng thời gian nhất định, giá trị của biến `matrix_offset` sẽ thay đổi, tạo ra hiệu ứng dịch chuyển trên LED ma trận.

Hiện thực bổ sung Timer 5 dùng để điều khiển quá trình dịch chuyển của LED ma trận. Giá trị của `DURATION_5` được đặt tùy ý.

Cuối cùng, cập nhật source file `main.c` để thực hiện dịch chuyển ký tự đang hiển thị sang trái.

```
/* USER CODE BEGIN PD */
#define DURATION_0 100
#define DURATION_1 25
#define DURATION_2 100
#define DURATION_3 100
#define DURATION_4 25
#define DURATION_5 400
/* USER CODE END PD */

/* USER CODE BEGIN PV */
int led_index = 0;
int led_buffer[LED_NUMBER] = {0, 0, 0, 0};
int matrix_index = 0;
int matrix_offset = 0;
/* USER CODE END PV */

/* USER CODE BEGIN 2 */
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(DOT_GPIO_Port, DOT_Pin, GPIO_PIN_RESET);
HAL_TIM_Base_Start_IT(&htim2);

setTimer0(DURATION_0);
setTimer1(DURATION_1);
setTimer2(DURATION_2);
setTimer3(DURATION_3);
setTimer4(DURATION_4);
setTimer5(DURATION_5);
/* USER CODE END 2 */

/* USER CODE BEGIN WHILE */
while (1) {
    updateClockBuffer();
    update7SEG(led_index);
    updateLEDMatrix(matrix_index);
    display7SEG(led_buffer[led_index]);
    displayLEDMatrix('A', (matrix_index + matrix_offset) %
       MATRIX_NUMBER);
```

```
37
38      if (timer0_flag == 1) {
39          setTimer0(DURATION_0);
40          HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
41      }
42
43      if (timer1_flag == 1) {
44          setTimer1(DURATION_1);
45          led_index = (led_index + 1) % LED_NUMBER;
46      }
47
48      if (timer2_flag == 1) {
49          setTimer2(DURATION_2);
50          HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
51      }
52
53      if (timer3_flag == 1) {
54          setTimer3(DURATION_3);
55          second++;
56          if (second >= 60) {
57              second = 0;
58              minute++;
59          }
60          if (minute >= 60) {
61              minute = 0;
62              hour++;
63          }
64          if (hour >= 24) {
65              hour = 0;
66          }
67      }
68
69      if (timer4_flag == 1) {
70          setTimer4(DURATION_4);
71          matrix_index = (matrix_index + 1) % MATRIX_NUMBER;
72      }
73
74      if (timer5_flag == 1) {
75          setTimer5(DURATION_5);
76          matrix_offset = (matrix_offset + 1) % MATRIX_NUMBER;
77      }
78  }
79  /* USER CODE END WHILE */
```